**Reprenez le contrôle à l’aide de Linux**

# Mais c’est quoi Linux ?

## Un système d’exploitation

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.

## La naissance de Linux

#### 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.

**Le projet Gnu**

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.

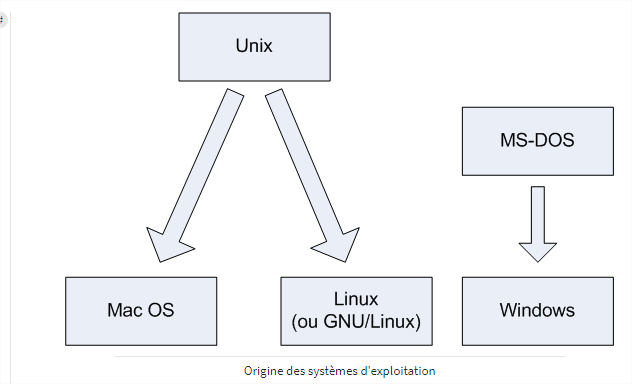
#### Pendant ce temps, Linus Torvalds s'amusait

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 distributions de Linux

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 mais voici 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.

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

* CentOs ;
* Fedora ;

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 Debian :



Logo de Ubuntu:



Quant à **Ubuntu** (figure suivante), c'est la distribution qui a créé la surprise. Elle est devenue très populaire en peu de temps. Pourquoi ? Il y a plusieurs raisons à cela.

* Elle est prévue pour le **grand public**. Le slogan est *« Linux for human beings »*, ce qui signifie « Linux pour des êtres humains ».
* **Les mises à jour sont fréquentes** : les développeurs travaillent d'arrache-pied et une nouvelle version de la distribution sort tous les six mois environ, ce qui vous permet de disposer des dernières nouveautés.
* Il y a beaucoup d'utilisateurs, donc **beaucoup de gens pour vous aider** si vous avez des questions (un point à ne pas négliger !).

# Téléchargez Linux, c’est gratuit

## Les deux visages de Linux

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.

Télécharger et graver le CD

**1/ Récupérer l'ISO**

Linux se télécharge sous la forme d'un gros fichier .iso d'environ 1,4 Go que vous pourrez graver sur un DVD.

**2/ Graver le DVD**

Il vous faut maintenant graver le gros fichier.isoque vous venez de télécharger.

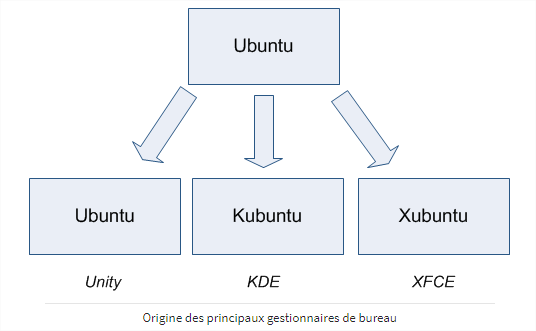
# Tester et installer Ubuntu

## Installation de Linux depuis Windows

Il suffit d’insérer le CD de Linux et redémarrer l’ordinateur

Cette méthode a toutefois des défauts. Ubuntu sera un peu moins performant (car il sera installé **dans** Windows) et nécessitera plus de mémoire vive (512 Mo).

Dans la mesure du possible, je vous conseille d'installer Ubuntu en utilisant la « vraie » méthode classique basée sur un CD d'installation.



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.

# Partitionner son disque

Bien que les risques soient minimes si vous suivez pas à pas mes instructions, je vous recommande quand même de **faire une sauvegarde de vos données importantes sous Windows** avant de commencer ce chapitre.

## Défragmentez votre disque

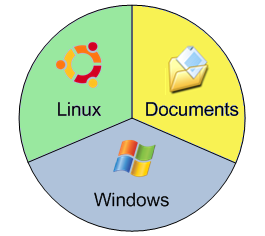
Attention, la défragmentation du disque **concerne uniquement les disques durs HDD** (mémoire de masse magnétique) dont je vais vous parler dans cette section. Si vous avez un disque SSD (mémoire flash), ne défragmentez surtout pas votre disque dur.

C'est une opération qui consiste en gros à mieux organiser les fichiers sur votre disque dur, à les rassembler pour éviter qu'ils ne soient éparpillés.

Pour lancer cet utilitaire, allez dans :Démarrer →Tous les programmes →Accessoires →Outils Système →Défragmenteur de disque.

Qu’est-ce que le partitionnement

Cela consiste à découper son disque en plusieurs parties (virtuellement, hein, pas pour de vrai ;) ).  
Si on allouait une partie à Linux, une autre à Windows et la dernière aux documents, le schéma ressemblerait alors à la figure suivante.



Ainsi, Linux et Windows pourraient s'échanger vos fichiers (photos, documents texte, etc.) sans interférer l'un avec l'autre. Autre intérêt de cette technique : si par hasard vous deviez réinstaller Linux ou Windows, vous ne perdriez pas vos documents car seules les partitions Linux ou Windows seraient formatées !

#### 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.  
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.

##### Lequel choisir ?

Je peux vous proposer ceci pour vos partitions :

* **Linux** : ext4 ;
* **Windows** : NTFS ;
* **Documents** : ext4.

Le but est que la partition « Documents » puisse être lue et écrite depuis Windows et Linux, afin que vous puissiez accéder à vos documents, que vous soyez sous Windows ou Linux.

L'ext4, comme vous le savez, fonctionne parfaitement sous Linux. Cependant, Windows ne le reconnaît pas… à moins d'installer un programme qui lui permettra de l'utiliser. Je peux vous recommander [**FS Driver**](http://www.fs-driver.org/), qui rend utilisables l'ext2, l'ext3 et l'ext4 sous Windows.

**Le partitionnement manuel**

##### 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 sempiternelsC:,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*…

**Étape 1 : réduire la taille de la partition Windows**

**Étape 2 : créer une partition pour installer Ubuntu**

**Étape 3 : créer une partition pour les documents**

**Étape 4 : créer une partition pour le *swap***

Il faut enfin créer une partition d'environ 1 Go appelée « swap ». C'est une partition un peu spéciale dont je ne vous ai pas parlé jusqu'ici pour ne pas vous embrouiller.

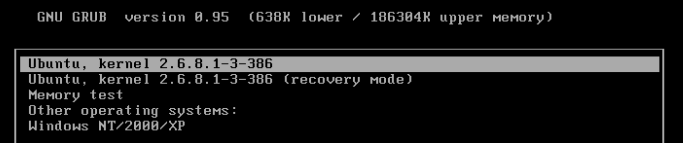
Pour faire simple, il s'agit d'une extension de la mémoire vive sur votre disque dur. Lorsque votre mémoire vive est pleine, Linux continue à fonctionner mais passe par le disque dur, grâce à la partition « swap ».

Laissez tout l'espace libre restant pour cette partition, mais sachez que ça ne sert à rien qu'elle fasse plus d'1 Go en général.

Ensuite on finalise l’installation

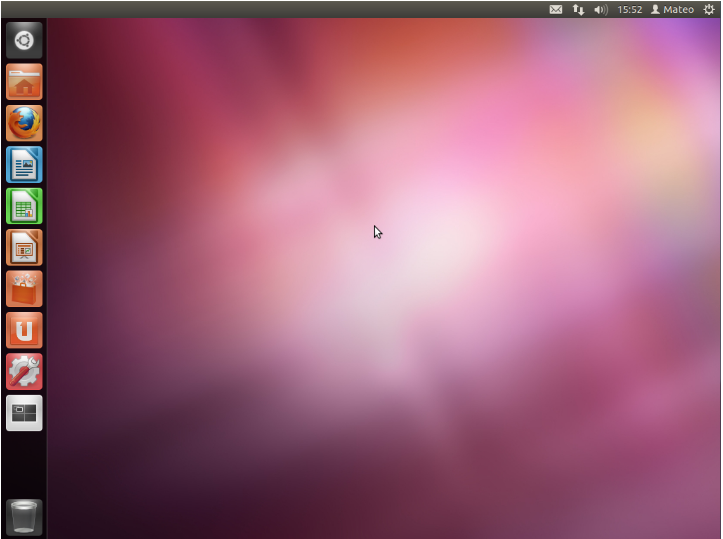
Grub au démarrage

Lors de chaque démarrage, si vous avez choisi d'installer Linux sur le même disque dur que Windows, on vous demandera quel OS vous voulez charger (figure suivante).

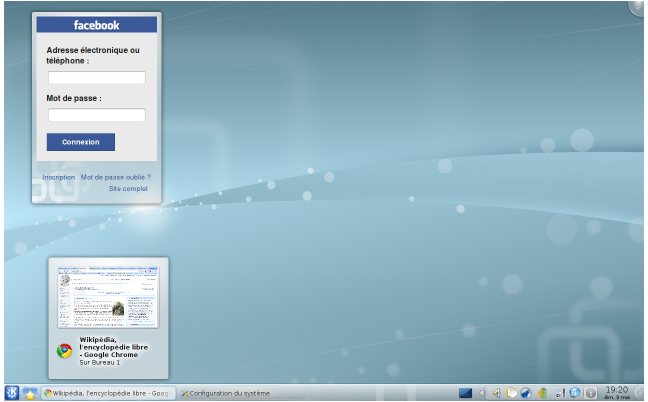


**Découverte du bureau Unity**

C'est le gestionnaire de bureau par défaut de Ubuntu



**Découverte du bureau KDE**



# Installez Linux dans une machine virtuelle

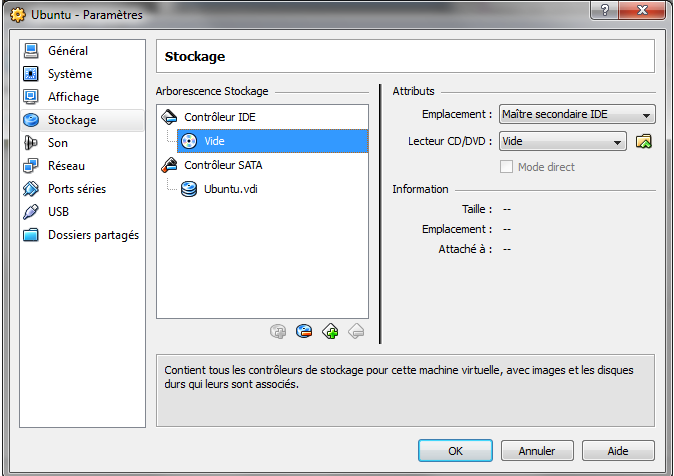
La virtualisation est une technique de plus en plus répandue en informatique : cela consiste à faire tourner un ordinateur « virtuel » dans votre ordinateur.

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

On lance l’installation tout en gardant les options par défaut, une fois l’installation effectuée il faut configurer le CD ou l’iso

Sur l’écran d’accueil d’Ubuntu, cliquez tout d’abord sur le nom de la machine virtuelle disponible dans la liste puis cliquez sur le bouton « Configuration ».

Dans la fenêtre qui s'ouvre, sélectionnez « Stockage » dans la liste de gauche puis sélectionnez la ligne « Vide » sous « Contrôleur IDE »



Au premier lancement, VirtualBox devrait vous afficher quelques messages d'information. Le plus important d'entre eux (figure suivante) vous indique qu'une touche spéciale (je l'appelle la « touche de secours ») vous permet de *sortir* de la machine virtuelle : il s'agit ici de Ctrl Droite, la touche Ctrl à droite du clavier.

Vous pouvez ensuite utiliser Ubuntu comme s'il était véritablement installé sur votre ordinateur ! Les performances sont légèrement moindres (car Windows tourne toujours en arrière-plan), mais cela est négligeable aujourd'hui, à moins que vous n'utilisiez des applications gourmandes comme les jeux.

Vous pouvez, si vous le désirez, afficher Ubuntu en plein écran. Il suffit d'effectuer la combinaison de touches Ctrl Droite + F (vous pouvez aussi aller dans le menu Machine → Passer en plein écran). Malheureusement, comme vous allez le constater, Ubuntu n’occupe pas tout votre écran. Pour y remédier, il va falloir installer les additions invité…

## Installation des additions invité

 Ce sont en fait des pilotes spéciaux que l'on installe dans la machine virtuelle pour améliorer ses performances.

Ubuntu n'a pour le moment pas « conscience » qu'il s'agit d'une machine virtuelle. Il s'exécute comme il le ferait sur une vraie machine. Le rôle des additions est de modifier légèrement Ubuntu pour qu'il prenne conscience qu'il est dans une machine virtuelle, ce qui aura pour effet d’améliorer son fonctionnement.

Parmi les améliorations apportées par cette modification, on note :

* de meilleures **performances graphiques** : le déplacement des fenêtres et les animations seront plus fluides ;
* un meilleur **suivi de la souris** : la souris réagira de façon plus naturelle ;
* un accès à de plus **grandes résolutions**, capables de suivre en temps réel la taille de la fenêtre de VirtualBox si vous la redimensionnez ou si vous l'affichez en plein écran (avec Ctrl Droite + F) ;
* le **partage du presse-papier** entre Windows et Linux. Vous pourrez copier du texte dans Linux et le coller dans Windows, et vice-versa !
* les **répertoires partagés** : vous pouvez faire en sorte qu'un répertoire de Windows apparaisse aussi sous Linux dans la machine virtuelle. Tous les changements dans ce dossier seront immédiatement répercutés sur les deux systèmes.

Pour installer les additions, rendez-vous dans le menu Périphériques → Installer les Additions invité. Vous pouvez aussi faire la combinaison de touches Ctrl Droite + D

Les additions sont installées ! Pour qu'elles soient prises en compte, vous devrez ensuite redémarrer Ubuntu.

Vous pouvez maintenant partager un dossier entre Windows et Ubuntu : rendez-vous dans le menu Périphériques → Dossiers partagés. Vous pourrez y indiquer un dossier existant de Windows et le nom du dossier équivalent dans Ubuntu.

# 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 sans avoir le mode graphique en arrière-plan :

* 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.

**Les différentes consoles**

Sous toute machine Linux, il y a donc non pas une mais six consoles qui fonctionnent en simultané (d'où les six raccourcis différents de Ctrl + Alt + F1 à Ctrl + Alt + F6).

Vous pouvez savoir dans quel terminal vous êtes lors du chargement : il est en effet marqué « tty1 » si vous êtes sur le terminal n°1.

## 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.

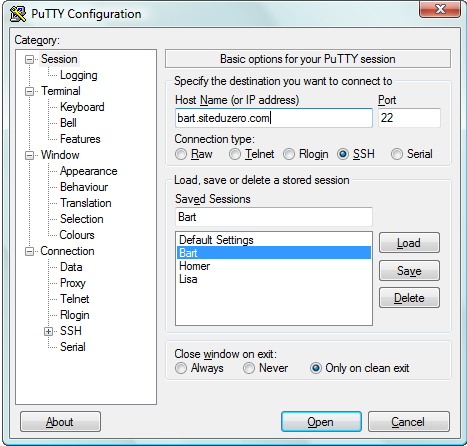
Il existe des tonnes et des tonnes de protocoles pour communiquer par Internet, mais pour ce qui est d'accéder à la ligne de commande à distance, c'est-à-dire à la console, il y en a deux principaux.

* **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



Si vous avez l'habitude de vous connecter à plusieurs serveurs différents, sachez qu'il est possible de sauvegarder les IP et configurations pour se connecter à chacun d'eux. Utilisez la section « Saved sessions » au centre pour enregistrer ou ouvrir des connexions pré-enregistrées.

# Entrer une commande

## L’invite de commandes

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 nombre 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.

**Shift + PgUp** : vous permet de « remonter » dans les messages envoyés par la console. En mode graphique, la molette de la souris accomplit aussi très bien cette action.

La touche Page Up est généralement représentée sur votre clavier par une flèche directionnelle Haut barrée de plusieurs petites lignes horizontales.

**Shift + PgDown** : pareil, mais pour redescendre.

**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 + U** : supprime tout ce qui se trouve à gauche du curseur. Si celui-ci est situé à la fin de la ligne, cette dernière sera donc supprimée.

**Ctrl + K** : supprime tout ce qui se trouve à droite du curseur. S'il est situé au début de la ligne, celle-ci sera donc totalement supprimée.

**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

## Organisation des dossiers

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, les sons , 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**

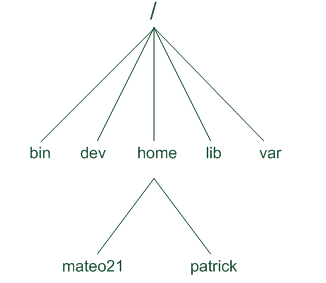
* **bin** : contient des programmes (exécutables) susceptibles d'être utilisés par tous les utilisateurs de la machine.
* **boot** : fichiers permettant le démarrage de Linux.
* **dev** : fichiers contenant les périphériques. En fait – on en reparlera plus tard – ce dossier contient des sous-dossiers qui « représentent » chacun un périphérique. On y retrouve ainsi par exemple le fichier qui représente le lecteur CD.
* **etc** : fichiers de configuration.
* **home** : répertoires personnels des utilisateurs. On en a déjà parlé un peu avant : c'est dans ce dossier que vous placerez vos fichiers personnels, à la manière du dossier Mes documents de Windows.

Chaque utilisateur de l'ordinateur possède son dossier personnel. Par exemple, dans mon cas mon dossier personnel se trouve dans/home/mateo21/. S'il y avait un autre utilisateur (appelons-le Patrick) sur mon ordinateur, il aurait eu droit lui aussi à son propre dossier :/home/patrick/.

* **lib** : dossier contenant les bibliothèques partagées (généralement des fichiers.so) utilisées par les programmes. C'est en fait là qu'on trouve l'équivalent des.dll de Windows.
* **media** : lorsqu'un périphérique amovible (comme une carte mémoire SD ou une clé USB) est inséré dans votre ordinateur, Linux vous permet d'y accéder à partir d'un sous-dossier de media. On parle de **montage**.
* **mnt** : c'est un peu pareil quemedia, mais pour un usage plus temporaire.
* **opt** : répertoire utilisé pour les *add-ons* de programmes.
* **proc** : contient des informations système.
* **root** : c'est le dossier personnel de l'utilisateur « root ». Normalement, les dossiers personnels sont placés danshome, mais celui de « root » fait exception. En effet, comme je vous l'ai dit dans le chapitre précédent, « root » est le superutilisateur, le « chef » de la machine en quelque sorte. Il a droit à un espace spécial.
* **sbin** : contient des programmes système importants.
* **tmp** : dossier temporaire utilisé par les programmes pour stocker des fichiers.
* **usr** : c'est un des plus gros dossiers, dans lequel vont s'installer la plupart des programmes demandés par l'utilisateur.
* **var** : ce dossier contient des données « variables », souvent des *logs* (traces écrites de ce qui s'est passé récemment sur l'ordinateur).

Cette liste de dossiers est en fait présente sur tous les OS de type Unix, et pas seulement sous Linux.

**Schéma résumé de l'architecture**



## 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, ex : which pwd qui affichera : /bin/pwd

Paramètres à la commande ls :

**-a**: afficher tous les fichiers et dossiers cachés, ex : ls -la

**-l**: liste détaillée , ex : ls -ll

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

**-t**: trier par date de dernière modification, ex: ls -lt

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.  
On verra comment créer des alias lorsqu'on saura se servir d'un éditeur de fichiers.

**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**

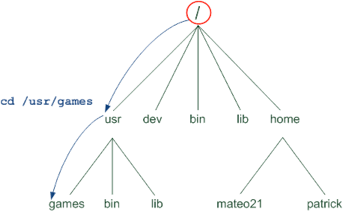
Un chemin relatif est un chemin qui dépend du dossier dans lequel vous vous trouvez.

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, ex : du -ah

-s : avoir juste le grand total, ex : du -sh

# Manipuler les fichiers

## cat & less : afficher un fichier

Cat : afficher tout le fichier

La commande cat permet d'afficher tout le contenu d'un fichier dans la console d'un coup.

le paramètre –n qui 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 & tail : afficher le début et la fin d'un fichier

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**.

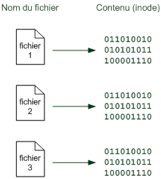
Le stockage des fichiers sur Linux.

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 :

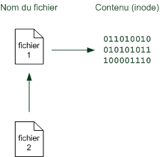
-rw-r--r-- 2 mateo21 mateo21 0 2008-07-31 13:55 one

-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

# La connexion sécurisée à distance avec SSH

## Se connecter à une console à distance

Une des grandes forces de Linux est que l'on peut s'en servir même si l'on est à des centaines de kilomètres de la machine. Ce fonctionnement date de l'époque d'Unix où il était nécessaire d'administrer des machines à distance.  
Aujourd'hui, si j'habite à Paris, je peux très bien contrôler un ordinateur sous Linux situé à Tokyo, au Japon, en même temps qu'un autre ordinateur situé au fin fond du Nevada, aux États-Unis. Je peux même ordonner à l'ordinateur de Tokyo d'envoyer un fichier à celui du Nevada.

## De Telnet à SSH

**Les protocoles**

Pour communiquer entre eux en réseau, deux ordinateurs doivent utiliser le même **protocole**.

Le protocole http est utilisé sur le web pour s'échanger des pages web.

Mais il existe bien d'autres protocoles ! Par exemple le FTP (*File Transfer Protocol*, protocole de transfert de fichiers), l'IMAP (*Internet Message Access Protocol*, utilisé pour s'échanger des e-mails), etc

**Le protocole Telnet : simple mais dangereux**

Un protocole très simple, très basique, a été créé dans les années 80 : c'est **Telnet**. Il sert juste à échanger des messages simples d'une machine à une autre.

En théorie donc, on peut communiquer avec un serveur à l'aide du protocole Telnet. Le problème de ce protocole… c'est justement qu'il est trop simple : les données sont transférées en clair sur le réseau. Il n'y a aucun chiffrement.

**Le protocole SSH : la solution pour sécuriser les données**

Comme on ne peut pas complètement empêcher quelqu'un d'intercepter les données qui transitent sur l'internet, il faut trouver un moyen pour que le client et le serveur

## Comment sont chiffrées les échanges avec SSH

**Quelles sont les différentes méthodes de chiffrement ?**

Il existe des tonnes d'algorithmes de chiffrement. Si l'on ne peut pas connaître tous les algorithmes de chiffrement, il faut par contre savoir que l'on peut les classer en deux catégories : les chiffrements *symétriques* et les chiffrements *asymétriques*.

**Le chiffrement symétrique**

C'est la méthode de chiffrement la plus simple. Cela ne veut pas dire qu'elle n'est pas robuste (il existe des chiffrements symétriques très sûrs). Cela veut plutôt dire que le fonctionnement est simple à comprendre. :-)

Avec cette méthode, on utilise une clé (un mot de passe secret) pour chiffrer un message.

Pour déchiffrer ensuite le message, on utilise cette même clé

Il faut donc que la personne qui chiffre et celle qui déchiffre connaissent toutes deux cette clé qui sert à chiffrer et déchiffrer.

Si le pirate intercepte un message chiffré, **il ne peut rien en faire s'il ne connaît pas la clé secrète**

Le chiffrement symétrique est donc puissant, mais il a un gros défaut : il faut communiquer d’abord en clair la clé de chiffrement…

Ce qu’il faudrait don c’est de chiffrer la clé.

**Le chiffrement asymétrique**

Le chiffrement symétrique utilise une seule et même clé pour chiffrer et déchiffrer.  
Le chiffrement asymétrique, lui, utilise une clé pour chiffrer, et une autre pour déchiffrer.

Il y a donc deux clés :

* une clé dite « **publique** » qui sert à **chiffrer** ;
* une clé dite « **privée** » qui sert à **déchiffrer**.

La clé publique ne sert qu'à chiffrer. Avec ce type d'algorithme, on ne peut déchiffrer un message que si l'on connaît la clé privée.

On demande à l'ordinateur de générer une paire de clés : une privée et une publique. Elles vont ensemble.

La clé publique peut être transmise en clair sur le réseau (elle est « publique »). Ce n'est pas grave si un pirate l'intercepte. En revanche, la clé privée — qui permet donc de déchiffrer — doit rester secrète.

**La création d'un tunnel sécurisé avec SSH**

**SSH combine chiffrement asymétrique et chiffrement symétrique**

SSH utilise les deux chiffrements : asymétrique et symétrique. Cela fonctionne dans cet ordre.

1. On utilise d'abord le chiffrement asymétrique pour s'échanger discrètement une clé secrète de chiffrement symétrique.
2. Ensuite, on utilise tout le temps la clé de chiffrement symétrique pour chiffrer les échanges.

Le chiffrement asymétrique est 100 à 1 000 fois plus lent que le chiffrement symétrique !  
Les ordinateurs s'échangent donc la clé de chiffrement symétrique de manière sécurisée (grâce au chiffrement asymétrique) et peuvent ensuite communiquer plus rapidement en utilisant en permanence le chiffrement symétrique.

**Les étapes de la création d'un canal sécurisé avec SSH**

Le serveur envoie la clé publique en clair au client pour qu'il puisse chiffrer

Le client génère une clé de chiffrement symétrique qu'il chiffre grâce à la clé publique qu'il a reçue.

Le client envoie la clé symétrique chiffrée au serveur. Le pirate peut l'intercepter, mais ne peut pas la déchiffrer car il faut pour cela la clé privée, connue seulement du serveur

Le serveur déchiffre la clé reçue grâce à sa clé privée qu'il a gardée bien au chaud chez lui.

Le client et le serveur connaissent maintenant tous les deux la clé symétriquetopsecret, et à aucun moment ils ne l'ont échangée en clair sur le réseau !  
Ils peuvent donc s'envoyer des messages chiffrés de manière symétrique en toute tranquillité. Ce chiffrement est plus rapide et tout aussi sûr que le chiffrement asymétrique car le pirate ne connaît pas la clé

Se connecter avec SSH et PuTTY

L’identification automatique par clé

# Transférer des fichiers

## **Wget** : téléchargement de fichiers

Cette commande permet de télécharger des fichiers directement depuis la console.

Il suffit d'indiquer l’adresse HTTP ou FTP d'un fichier à télécharger ex :

wget http://cdimage.debian.org/debian-cd/4.0\_r5/i386/iso-cd/ debian-40r5-i386-businesscard.iso

## Scp : copier des fichiers sur le réseau

Cette commande permet de copier des fichiers d'un ordinateur à un autre à travers le réseau ! Le tout de manière sécurisée.

On l'utilise comme ceci :

**scp** fichier\_origine copie\_destination

Le premier élément à indiquer est la position du fichier que l'on veut copier. Le second élément correspond au répertoire de destination où il doit être copié.

Chacun de ces éléments peut s'écrire sous la forme suivante : login@ip:nom\_fichier. Le login et l'IP sont facultatifs. Si vous n'écrivez ni login ni IP, scp considérera que le fichier se trouve sur votre ordinateur.

**Copier un fichier de votre ordinateur vers un autre**

Prenons un cas concret (figure suivante) pour que vous puissiez mieux voir comment ça s'utilise :



Notez que l'on peut utiliser le symbole ~ pour indiquer « mon répertoire personnel »



**Copier un fichier d'un autre ordinateur vers le vôtre**

On peut également faire le contraire (figure suivante), c'est-à-dire récupérer un fichier qui se trouve sur un autre ordinateur et le placer chez nous :



Si je mets un espace et un point à la fin. Il signifie « copier dans le répertoire dans lequel je me trouve ».

**Le piège du port**

Si le serveur SSH auquel vous essayez de vous connecter n'est pas sur le port standard (22), il faudra indiquer le numéro du port avec l'option -P :



## ftp & sftp : transférer des fichiers

Le FTP (*File Transfer Protocol*) est un protocole permettant d'échanger des fichiers sur le réseau. Il est assez ancien (1985) et toujours utilisé à l'heure actuelle pour transférer des fichiers.

On l'utilise en général dans deux cas.

* Pour télécharger un fichier depuis un serveur FTP public. En général, les navigateurs web font cela de manière automatique et transparente lorsque vous cliquez sur un lien de téléchargement.

La connexion se fait alors en **mode anonyme**.

* Pour transférer des fichiers vers un serveur FTP privé (et éventuellement en télécharger aussi). Lorsque l'on prend un hébergement pour son site web, l'hébergeur nous donne en général des accès FTP pour aller y déposer les fichiers du site.

La connexion se fait donc en **mode authentifié**.

Tout le monde n'a pas forcément accès à un serveur FTP privé, aussi je vous propose pour les exemples suivants de vous connecter à un serveur FTP public (rassurez-vous, si vous voulez vous connecter à un FTP privé, la méthode reste la même).

#### Connexion à un serveur FTP

Essayons de nous connecter au serveur FTP de Debian, accessible à l'adresse suivante :ftp://ftp.debian.org.

Le serveur FTP devrait répondre en vous demandant un login et un mot de passe. Pour les serveurs FTP publics, le login à utiliser est toujours anonymous

Pour le mot de passe, peu importe ce que vous mettez, vous serez acceptés. :-)

Il n’y a qu’un seul dossier debian

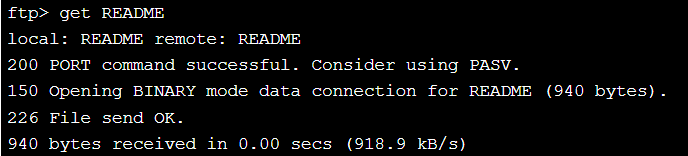
**Le transfert de fichiers**

Si vous souhaitez récupérer un fichier ou en envoyer un, il y a deux commandes à connaître (figure suivante) :

* put : envoie un fichier vers le serveur ;
* get : télécharge un fichier depuis le serveur.

Notez qu'il est impossible d'utiliser put sur les serveurs FTP publics comme celui auquel nous sommes connectés. Seul le téléchargement de fichiers est autorisé.  
D'autres commandes, comme chmod (qui permet de modifier les droits des fichiers), ne sont pas activées non plus.

Vous pouvez par exemple récupérer le fichier README en écrivant get README



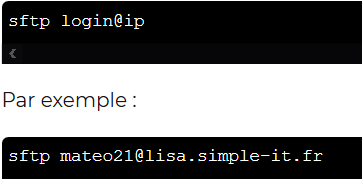
Il a été téléchargé dans le dossier dans lequel vous vous trouviez sur votre ordinateur.

Pour quitter le serveur et vous déconnecter, vous avez le choix entre la bonne vieille combinaison de touches Ctrl + D qui commande la fermeture de la session ou encore les commandes bye, exit et quit, qui sont analogues.

#### sftp : un FTP sécurisé

Le protocole FTP a un défaut : il n'est pas sécurisé ; les données ne sont pas cryptées. Quelqu'un ayant accès au réseau pourrait alors intercepter le contenu des fichiers que vous échangez ou encore votre mot de passe lors de la connexion.

Pour remédier à cela, on a inventé sftp, qui repose sur SSH pour sécuriser la connexion



Pour se connecter en SFTP, on utilise le même port que SSH (soit 22 par défaut). Si votre serveur SSH fonctionne sur un autre port, vous devrez le préciser comme ceci :sftp -oPort=27401 mateo21@serveur

À l'heure actuelle, le SFTP reste assez peu utilisé. Les hébergeurs web utilisent toujours le FTP classique alors que la plupart des logiciels graphiques comme FileZilla sont pourtant capables de se connecter en SFTP.

## rsync : synchroniser des fichiers pour une sauvegarde

# Vim: l'éditeur de texte du programmeur

Editeur de texte pour Linux.

Tape vi + nom du fichier à créer

Tap i pour pouvoir écrire

Tape échap pour te déplacer

Touche I permet une insertion en début de ligne tout en haut

Touche A permet une insertion à la fin de la ligne.

:q ! pour quitter sans sauver

:q = quitte si fichier non modifié.

ZZ quitte et sauve le fichier

:w enregistre le fichier

:r nomFichier = va copier ici le contenu du fichier en question.

Substitution :

:[1ere ligne, dernière ligne]s/Modèle/Remplacement/[gil]

Ex :

:1,5s/bin/sbin

:5,30s/bash/kash/g

Les options sont supprimées dès que tu fermes vim, sauf si tu créé un fichier caché vi .exrc avec les commandes des options souhaitées :

set number

set showmode

set autoindent

Créer une abréviation :

En mode commande tu dois faire : ab puis le nom de ton abréviation puis l'abréviation, ex :

:ab DV Devops

A chaque fois que tu taperas DV il affichera Devops.

Si tu tapes ls, alors ton entrée standard est ton dossier courrant

Et la sortie est l'écran

Si tu fais ls DOC, l'entrée standard est DOC.

cat > f1

Si tu tapes un truc puis tape cat > f1, tu renvoie le texte dans f1

cat f1>f2  // Va copier le contenu de f1 dans f2 tout en écrasant ce qu'il y a dans f2.

## Installer Vim

Sous Ubuntu, il faut savoir que ce n'est pas Vim qui est installé mais **Vim-tiny**, une version allégée. Personnellement, elle ne me convient pas ; de plus, elle est limitée en possibilités. Je vous invite donc à installer le vrai Vim complet en tapant:

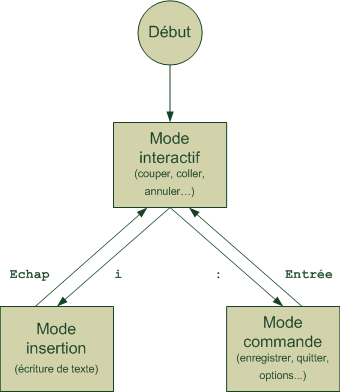
* sudo apt-get install vim

Vous pourrez alors lancer le logiciel en tapant la commande **vim**.  
La commande **vi** fonctionne aussi mais il est recommandé de taper plutôt vim.

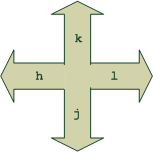
* + 1. **Les modes d'édition de Vim**

1. Vim possède trois modes de travail différents.

* **Mode interactif** : c'est le mode par défaut par lequel vous commencez. En lançant Vim, vous êtes donc en mode interactif.

1. Dans ce mode, vous ne pouvez pas écrire de texte. Le mode interactif est un mode puissant qui permet de se déplacer dans le texte, de supprimer une ligne, copier-coller du texte, rejoindre une ligne précise, annuler ses actions, etc.
2. **Mode insertion** : celui-là, c'est celui que vous connaissez ! Vous tapez du texte et ce dernier s'insère à l'endroit où se trouve le curseur.
3. Pour entrer dans ce mode, il existe plusieurs possibilités. L’une des plus courantes est d'appuyer sur la touchei(*insertion*). Pour en sortir, il faut appuyer sur la touche Echap.
4. **Mode commande** : ce mode permet de lancer des commandes telles que « quitter », « enregistrer », etc. Vous pouvez aussi l'utiliser pour activer des options de Vim (comme la coloration syntaxique, l'affichage du numéro des lignes…). Vous pouvez même envoyer des commandes au shell (la console) telles que ls,locate,cp, etc.
5. Pour activer ce mode, vous devez être en mode interactif et appuyer sur la touche deux points « : ». Vous validerez la commande avec la touche Entrée et reviendrez alors au mode interactif.
6. 
   * 1. **Opérations basiques (déplacement, écriture, enregistrement…)**
7. On ouvre vim en tapant simplement : vim
8. Vous pouvez aussi ouvrir un fichier en ajoutant son nom en paramètre:

* vim nomdufichier

1. Si le fichier n'existe pas, il sera créé.
   * + 1. **i: insérer du texte**
       2. **Le déplacement**
2. En mode interactif, il est possible de déplacer le curseur au sein du texte. Pour cela, on utilise les touches :
3. 
4. * + - 1. **0 et $: se déplacer en début et fin de ligne**
         2. **w : se déplacer de mot en mot**
5. Avec w, vous pouvez vous déplacer de mot en mot dans le fichier.
   * + 1. **:w : enregistrer le fichier**
6. Pour enregistrer votre fichier, vous devez être au préalable en mode interactif (appuyez surEchap pour vous en assurer).
7. Si ton fichier n'a pas de nom, tu tapes:w nomFichier et ça l'enregistrera au nom choisi.
   * + 1. **:q : quitter**
8. Maintenant que vous avez enregistré, vous pouvez quitter Vim en tapant:q

:q ! pour quitter en sauvegardant

:q = quitte si fichier non modifié.

ZZ quitte et sauve le fichier

* + - 1. **:wq: enregistrer puis quitter**

1. C'est la combinaison des deux commandes que nous venons de voir. Vous enregistrez et quittez immédiatement Vim lorsque vous tapez:wq.
   * 1. **Opérations standard (copier, coller, annuler…)**
2. Nous allons effectuer la majorité de ces actions en mode interactif : appuyez sur la touche Echap si vous n'y êtes pas déjà.
   * + 1. **x : effacer des lettres**
3. Placez le curseur sur une lettre en mode interactif puis appuyez sur **x** pour l'effacer.  
   Cela revient à appuyer sur **Suppr** en mode insertion.
4. On peut aller plus loin et effacer plusieurs lettres d'un coup. Pour cela, utilisez la formule suivante :
5. (nombre)x
   * + - 1. **dd : supprimer une ligne**
6. Appuyez deux fois sur d (**dd**) pour supprimer toute la ligne sur laquelle se trouve le curseur.
7. Mieux : vous pouvez faire précéder cette instruction d'un nombre de lignes à supprimer. Par exemple, si vous tapez 2dd, vous supprimerez deux lignes d'un coup.
   * + - 1. **dw : supprimer un mot**
8. Placez le curseur sur la première lettre d'un mot. Tapez ensuite **dw**(*delete word*) : cela supprime le mot complet !
9. Vous pouvez aussi supprimer les trois prochains mots en tapant3dw. Notez que le 3 peut être placé entre le d et le w ; cela revient au même :d3w(qui peut se lire « *delete 3 words* »).
   * + - 1. **d0 et d$ : supprimer le début ou la fin de la ligne**
10. En tapant **d0**, vous supprimez du curseur jusqu'au début de la ligne.
11. En tapant **d$**, vous supprimez du curseur jusqu'à la fin de la ligne.
12. **yy : copier**
13. **yy** copie la ligne actuelle en mémoire.
14. Vous pouvez aussi utiliser **yw** pour copier un mot, y**$** pour copier du curseur jusqu'à la fin de la ligne, etc.
    * + 1. **p : coller**
15. Si vous avez « coupé » du texte avec **dd** ou copié du texte avec **yy**(ou un de leurs équivalents) vous pouvez ensuite le coller avec la touche **p**.
16. Vous pouvez aussi coller plusieurs fois un texte en faisant précéder le **p** d'un nombre. Par exemple, **8p** collera huit fois le texte en mémoire.
17. Si je place mon curseur sur une ligne, que je tape yy puis 8p, je la collerai donc huit fois
    * + 1. **u : annuler les modifications**
18. Pour annuler vos dernières modifications, appuyez sur **u**(*undo*). Si vous souhaitez annuler vos quatre dernières modifications, tapez sur **4u**.
    * + 1. **G : sauter à la ligne n° X**
19. Toutes les lignes d'un fichier possèdent un numéro. La numérotation commence à 1.
20. Regardez bien en bas à droite de Vim, vous devriez voir quelque chose comme 4,3.   
    4 correspond au numéro de la ligne sur laquelle se trouve le curseur, et 3 au numéro de la colonne (3e lettre de la ligne).
21. Vous pouvez par exemple directement sauter à la ligne n° 7 en tapant **7G**(attention, c'est un « G » majuscule)
22. Pour sauter à la dernière ligne, tapez simplement G.  
    Pour revenir à la première ligne, tapez **gg**
    * 1. **Opérations avancées (split, fusion, recherche…)**
         1. **/ : rechercher un mot**
23. Si vous tapez **/** vous passez en mode recherche. Le curseur se place en bas de l'écran (vous indiquant que vous êtes passés en mode commande).  
    Écrivez ensuite le mot que vous recherchez, par exemple « remplir » : /remplir. Tapez ensuite sur Entrée pour valider
24. Le curseur se place alors sur la prochaine occurrence de « remplir » dans le fichier.  
    Pour passer à la prochaine occurrence du mot, plus bas dans le fichier (s'il apparaît plusieurs fois), appuyez sur **n**. Pour rechercher en arrière, appuyez sur **N**
25. Si vous souhaitez dès le départ lancer une recherche qui remonte vers le début du fichier, utilisez **?** au lieu de **/** pour lancer la recherche ; le fonctionnement reste le même.
    * + 1. **:s: rechercher et remplacer du texte**
26. **:s/ancien/nouveau** 🡪 remplace la première occurrence de la ligne où se trouve le curseur ;
27. **:s/ancien/nouveau/g** : remplace toutes les occurrences de la ligne où se trouve le curseur ;
28. **:#,#s/ancien/nouveau/g** : remplace toutes les occurrences dans les lignes n° # à # du fichier ;
29. **:%s/ancien/nouveau/g**: remplace toutes les occurrences dans tout le fichier.
    * + 1. **:r : fusion de fichiers**
30. Avec **:r** vous pouvez insérer un fichier à la position du curseur. Vous devez indiquer le nom du fichier à insérer, par exemple ::r autrefichier.
31. L'autocomplétion avec Tab fonctionne là aussi, donc pas besoin d'écrire le nom du fichier en entier !
    * + 1. **Le découpage d'écran (split)**
32. Vim possède une fonctionnalité pratique : il permet de découper l'écran et d'ouvrir plusieurs fichiers.
33. Le fichier est ouvert une seconde fois (ce qui vous permet de voir deux endroits différents du fichier à la fois) mais il est bien entendu possible d'ouvrir deux fichiers différents. Pour cela, ajoutez le nom du fichier à ouvrir à la suite de la commande ::sp autrefichier. Bonne nouvelle : l'autocomplétion à l’aide de la toucheTabfonctionne aussi dans Vim !
    * + - 1. **:vsp : découper l'écran verticalement**
34. Si le découpage horizontal par défaut ne vous convient pas, sachez que vous pouvez aussi effectuer un découpage vertical avec:vsp
    * + - 1. **Les principaux raccourcis en écran splitté**
35. Chaque morceau de l'écran (correspondant à un fichier) est appelé **viewport**.  
    Voici une liste de raccourcis pratiques que vous pouvez utiliser lorsque l'écran est splitté (scindé).

* Ctrl + wpuisCtrl + w : navigue de viewport en viewport. Répétez l'opération plusieurs fois pour accéder au viewport désiré.
* Ctrl + wpuisj : déplace le curseur pour aller au viewport juste en dessous. La même chose fonctionne avec les touchesh,ketlque l'on utilise traditionnellement pour se déplacer dans Vim.
  + - 1. **:! : lancer une commande externe**

1. Il est possible d'écrire des commandes traditionnelles du shell directement dans Vim. Pour cela, commencez par taper:!suivi du nom de la commande.
2. Essayez par exemple de taper:!ls. Vous afficherez alors le contenu du dossier dans lequel vous vous trouvez !  
   Cette fonctionnalité est bien pratique pour effectuer quelques actions sans avoir à quitter Vim.
   * 1. **Les options de Vim**
3. Vim peut être personnalisé de deux façons différentes :

* En activant ou désactivant des options. [La documentation complète des options](http://vimdoc.sourceforge.net/htmldoc/) est disponible en ligne.
* En installant des plugins. Voyez [la page officielle des plugins les plus téléchargés de Vim](https://www.vim.org/scripts/script_search_results.php?keywords=&script_type=&order_by=downloads&direction=descending&search=search).

1. Les options peuvent être activées après le démarrage de Vim en lançant des commandes. Cependant, ces options seront « oubliées » dès que vous quitterez le logiciel.  
   Si vous voulez que les options soient activées à chaque démarrage de Vim, il faut créer un fichier de configuration.vimrcdans votre répertoire personnel.
2. La première méthode consiste à activer l'option en mode commande. Une fois Vim ouvert, pour activer l'option nommée « option », tapez :
3. :set option
4. Pour la désactiver, tapez :
5. :set nooption
6. Certaines options doivent être précisées avec une valeur, comme ceci :
7. :set option=valeur
8. Pour connaître l'état d'une option :
9. :set option?
   * + - 1. **Activer des options dans un fichier de configuration**
10. C'est à mon avis la meilleure façon de procéder. Commencez par copier un fichier de configuration déjà commenté qui vous servira d'exemple : il y en a un dans/etc/vimqui s'appellevimrc.
11. Copiez-le dans votre répertoire personnel en le faisant précéder d'un point (pour que ce soit un fichier caché) :

* $ cp /etc/vim/vimrc  ~/.vimrc

1. Ouvrez maintenant ce fichier… avec Vim, bien sûr.

* $ vim .vimrc

1. Passons maintenant à l'activation de quelques commandes bien utiles
   * + 1. **syntax : activer la coloration syntaxique**
2. Activez donc l’option :

* syntax on

1. Notez qu'il faut enregistrer, quitter et relancer Vim pour que le changement soit pris en compte… sauf bien sûr si vous activez l'option à la volée en tapant dans Vim:set syntax=ON
   * + 1. **background : coloration sur un fond sombre**
2. Par défaut, la coloration de Vim est plus adaptée aux fonds clairs. Les commentaires, par exemple, sont écrits en bleu foncé sur noir… ce qui n’est pas très lisible.
3. Si votre console est sur fond noir (comme chez moi), je vous recommande d'activer la prochaine optionbackgroundet de la mettre à dark.

* set background=dark
  + - 1. **number : afficher les numéros de ligne**

1. Il est possible d'afficher le numéro de chaque ligne à gauche (figure suivante) :

* set number
  + - 1. **showcmd : afficher la commande en cours**

1. Lorsque vous écrivez une commande comme2ddpour supprimer deux lignes, vous écrivez à l'aveugle. Vous ne voyez pas ce que vous avez écrit.
2. Contrairement à Vi, Vim permet de pallier ce problème… encore faut-il activer l'option :

* set showcmd
  + - 1. **ignorecase : ignorer la casse lors de la recherche**

1. Lors d'une recherche, si vous souhaitez que Vim ne fasse pas la différence entre les majuscules et les minuscules, activez cette option :

* set ignorecase
  + - 1. **mouse : activer le support de la souris**

1. Eh oui ! Même en mode console, il est possible d'utiliser la souris.  
   Commencez par activer le support de cette dernière :

* set mouse=a

# 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.

Le Shell c’est l’environnement console où l’on tape les commandes Linux, il en existe plusieurs:

* **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.

Par convention on donne l’extension .sh aux scripts shells ex: vim test.sh

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

**Exécuter le script bash**

Pour cela il faut donner le droit d’exécution au script avec un chmod +x NomDuFichier

**Exécution du script**

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

Le . signifie regarde dans le dossier où je suis actuellement.

Actuellement, le script doit être lancé via ./test.sh et vous devez être dans le bon répertoire.  
Ou alors vous devez taper le chemin en entier, comme /home/qeyss/scripts/test.sh.

Comment font les autres programmes pour pouvoir être exécutés depuis n'importe quel répertoire sans « ./ » devant ?

Ils sont placés 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, /usr/bin ou /usr/local/bin (ou encore un autre répertoire du PATH).

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

**Ajouter un dossier au PATH :**

J’ai créé un dossier scripts dans mon home cloud\_user, pour l’ajouter au PATH je dois taper la commande suivante : export PATH=$PATH:/home/cloud\_user/scripts.

# Afficher et manipuler des variables

## Déclarer une variable

message='With almost 78,000 cases of Covid-19 now confirmed across the globe, experts say the situation will soon reach a critical threshold'

Ne mettez pas d'espaces autour du symbole égal « = » ! Le bash est très pointilleux

## Echo : Afficher une variable

echo $message

echo "I love this game"

message='Bonjour tout le monde'

**Les doubles quotes " "**

echo "Le message est : $message"

Va afficher : Le message est : Bonjour tout le monde

**Les back quotes ` `**

Les back quotes demandent à bash d'**exécuter** ce qui se trouve à l'intérieur.

message=`pwd`

echo "Vous êtes dans le dossier $message"

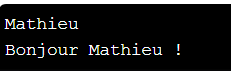
Vous êtes dans le dossier /home/qeyss/bin

## Read : Demander une saisie

Vous pouvez demander à l'utilisateur de saisir du texte avec la commande read. Ce texte sera immédiatement stocké dans une variable.

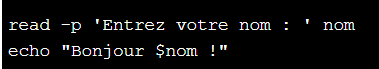


Lorsque vous lancez ce script, rien ne s'affiche, mais vous pouvez taper du texte (votre nom, par exemple) :



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

Bon : notre programme n'est pas très clair et nous devrions afficher un message pour que l'utilisateur sache quoi faire. Avec l'option -p de read, vous pouvez faire cela :



**-n : limiter le nombre de caractères**

Avec -n, vous pouvez au besoin couper au bout de X caractères si vous ne voulez pas que l'utilisateur insère un message trop long.

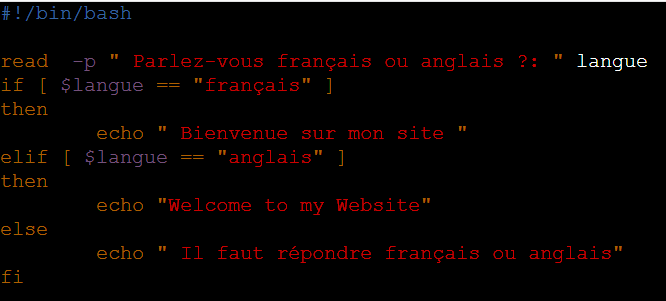


**-s : ne pas afficher le texte saisi**

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



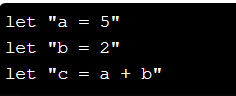
Si vous voulez insérer des retours à la ligne, il faudra activer le paramètre -e et utiliser le symbole \n



## Effectuer des opérations mathématiques

En bash, les variables sont toutes des chaînes de caractères. En soi, le bash n'est pas vraiment capable de manipuler des nombres ; il n'est donc pas capable d'effectuer des opérations.

Heureusement, il est possible de passer par des commandes (eh oui, encore). Ici, la commande à connaître est let.



Ici c vaudra 7.

Les opérations utilisables sont :

* l'addition : + ;
* la soustraction : - ;
* la multiplication : \* ;
* la division : / ;
* la puissance : \*\* ;
* le modulo (renvoie le reste de la division entière) : %.

## Les variables d’environnement

Actuellement, les variables que vous créez dans vos scripts bash n'existent que dans ces scripts. En clair, une variable définie dans un programme A ne sera pas utilisable dans un programme B.

Les variables d'environnement sont des variables que l'on peut utiliser dans n'importe quel programme. On parle aussi parfois de **variables globales**. Vous pouvez afficher toutes celles que vous avez actuellement en mémoire avec la commande env :

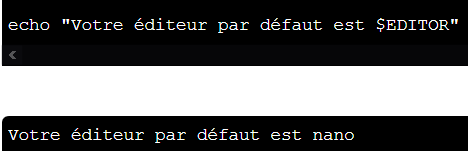
Parmi celles que je peux vous commenter et qui peuvent s'avérer utiles, on trouve :

* SHELL : indique quel type de shell est en cours d'utilisation (sh, bash, ksh…) ;
* PATH : une liste des répertoires qui contiennent des exécutables que vous souhaitez pouvoir lancer sans indiquer leur répertoire. Nous en avons parlé un peu plus tôt. Si un programme se trouve dans un de ces dossiers, vous pourrez l'invoquer quel que soit le dossier dans lequel vous vous trouvez ;
* EDITOR : l'éditeur de texte par défaut qui s'ouvre lorsque cela est nécessaire ;
* HOME : la position de votre dossier home ;
* PWD : le dossier dans lequel vous vous trouvez ;
* OLDPWD : le dossier dans lequel vous vous trouviez auparavant.

Notez que les noms de ces variables sont, par convention, écrits en majuscules.

Comment utiliser ces variables dans vos scripts ? C'est très simple, il suffit de les appeler par leur nom !

Exemple :



Plus rarement, vous pourriez avoir besoin de définir votre propre variable d'environnement. Pour cela, on utilise la commande **export**

Les variables de paramètres

Comme toutes les commandes, vos scripts bash peuvent eux aussi accepter des paramètres. Ainsi, on pourrait appeler notre script comme ceci :

./variables.sh param1 param2 param3

Pour récupérer ces paramètres on fait ainsi :

 $# : contient le nombre de paramètres ;

 $0 : contient le nom du script exécuté (ici ./variables.sh) ;

 $1 : contient le premier paramètre ;

 $2 : contient le second paramètre ; etc..

## Les tableaux

Le bash gère également les variables « tableaux ». Ce sont des variables qui contiennent plusieurs cases, comme un tableau. Vous en aurez probablement besoin un jour ; voyons comment cela fonctionne.

Pour définir un tableau, on peut faire comme ceci :



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

${tableau[2]}

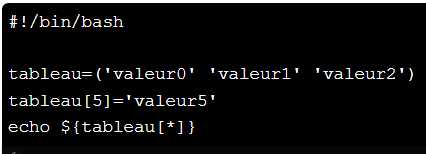
… ceci affichera le contenu de la case n° 2 (donc valeur2).

Les cases sont numérotées à partir de 0 ! La première case a donc le numéro 0.

Vous pouvez aussi définir manuellement le contenu d'une case :



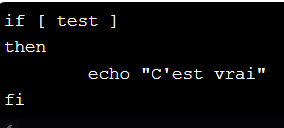
Vous pouvez afficher l'ensemble du contenu du tableau d'un seul coup en utilisant ${tableau[\*]}



## Scripting conditionals

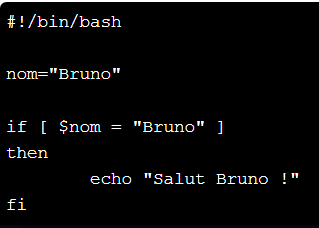
**if : la condition la plus simple**

Syntaxe :



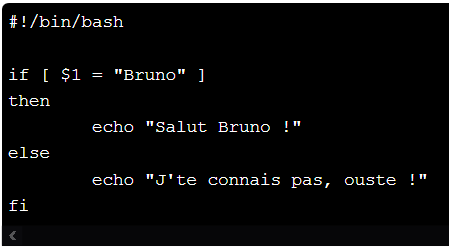
Le mot fi (if à l'envers !) à la fin indique que le if s'arrête là. Tout ce qui est entre le then et le fi sera exécuté uniquement si le test est vérifié.

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 ] !



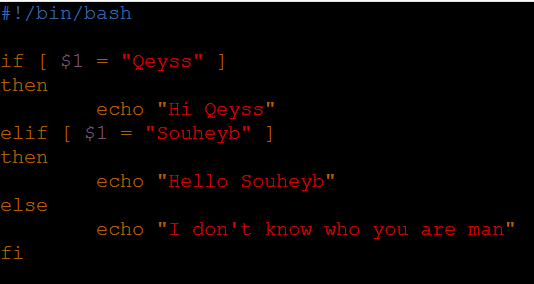
Ensuite il faut faire un chmod +x au fichier puis le lancer avec ./NomDuScript

Autre script avec le 1er paramètre $1 envoyé au script



Tu le lanceras ainsi ./NomDuScript Paramètre ex : ./name.sh Bruno

**If et Elif**



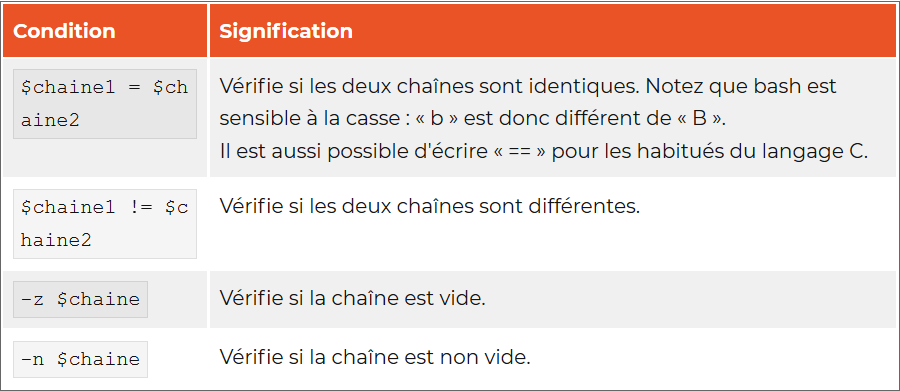
Si tu n’envoies aucun paramètre le script plante car il attend un paramètre.

**Les différents types de tests**

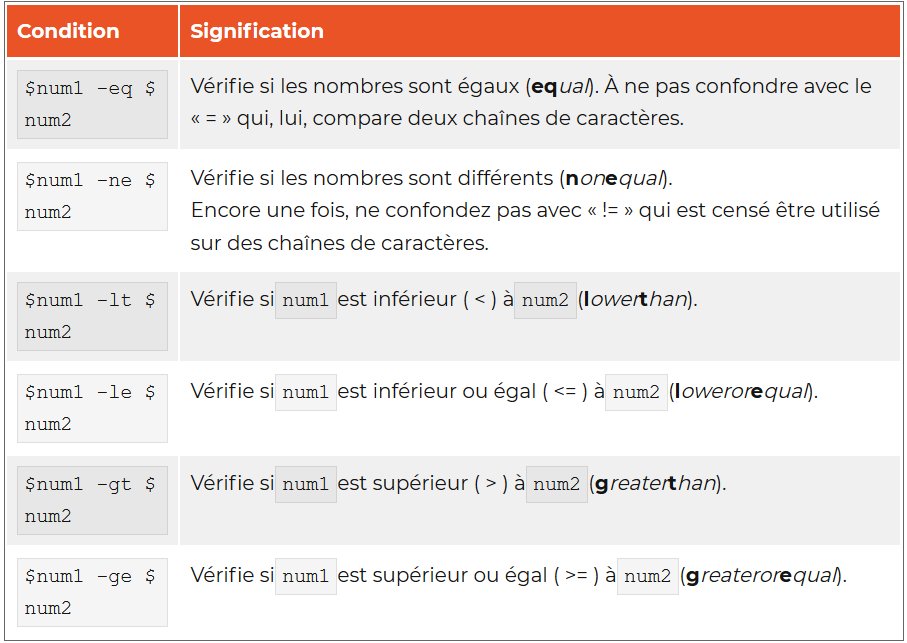
Il est possible d’effectuer trois types de tests différents en bash :

* des tests sur des chaînes de caractères ;
* des tests sur des nombres ;
* des tests sur des fichiers.

**Tests sur des chaînes de caractères**



**Tests sur des nombres**



**Tests sur des fichiers**

Un des avantages de bash sur d'autres langages est que l'on peut très facilement faire des tests sur des fichiers : savoir s'ils existent, si on peut écrire dedans, s'ils sont plus vieux, plus récents, etc.



**Effectuer plusieurs tests à la fois**

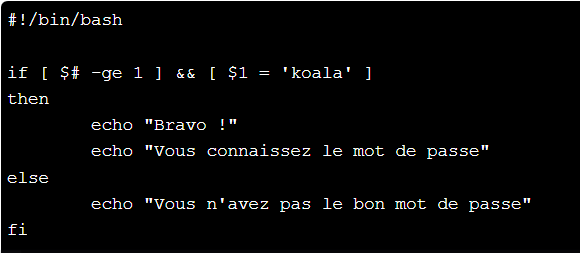
Dans un if, il est possible de faire plusieurs tests à la fois. En général, on vérifie :

* 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 ».

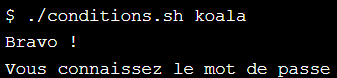
Il faut encadrer chaque condition par des crochets. Prenons un exemple :



Le test vérifie deux choses :

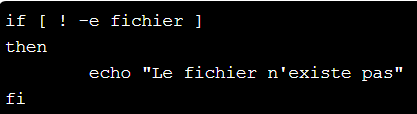
* qu'il y a au moins un paramètre (« si$#est supérieur ou égal à 1 ») ;
* que le premier paramètre est bienkoala(« si$1est égal àkoala »).

Si ces deux conditions sont remplies, alors le message indiquant que l'on a trouvé le bon mot de passe s'affichera.



**Inverser un test**

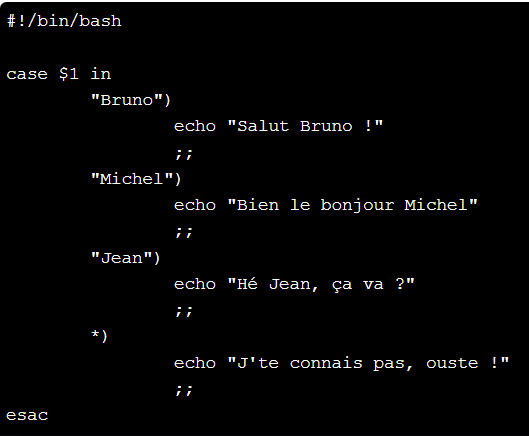
Il est possible d'inverser un test en utilisant la négation. En bash, celle-ci est exprimée par le point d'exclamation « ! ».



**case : tester plusieurs conditions à la fois**

À la place de saisir plusieurs elif, il est possible d’utiliser l'instruction **case**, c’est le switch en bash.

Exemple :



case $1 in

Tout d'abord, on indique que l'on veut tester la valeur de la variable$1. Bien entendu, vous pouvez remplacer$1par n'importe quelle variable que vous désirez tester.

"Bruno")

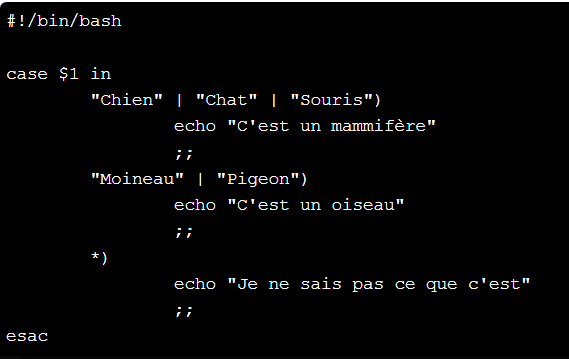
Là, on teste une valeur. Cela signifie « Si$1est égal à Bruno ». Notez que l'on peut aussi utiliser une étoile comme joker : « B\* » acceptera tous les mots qui commencent par un B majuscule.

\*)

C'est en fait le « else » ducase. Si aucun des tests précédents n'a été vérifié, c'est alors cette section qui sera lue.

esac marque la fin du case(esac, c'est « case » à l'envers !).

Nous pouvons aussi faire des « ou » dans uncase. Dans ce cas, petit piège, il ne faut pas mettre deux||mais un seul ! Exemple :



## Scripting loops

Tu peux créer une variable ainsi dans un script:

DIRECTORY = "/home/qeyss/test"

**while : boucler « tant que »**

Syntaxe :

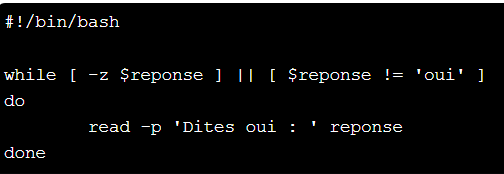
while [ test ]

do

        echo 'Action en boucle'

done

Exemple:



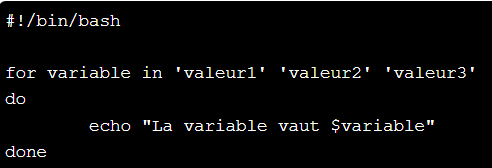
« Tant que la réponse est vide ou que la réponse est différente deoui ».  
Nous sommes obligés de vérifier d'abord si la variable n'est pas vide, car si elle l'est, le second test plante.

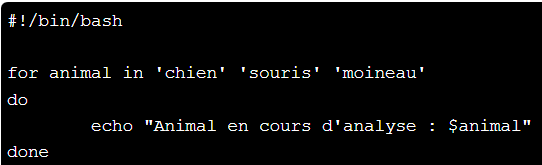
**for : boucler sur une liste de valeurs**

**Parcourir une liste de valeurs**

La boucleforpermet de parcourir une liste de valeurs et de boucler autant de fois qu'il y a de valeurs.

Syntaxe :





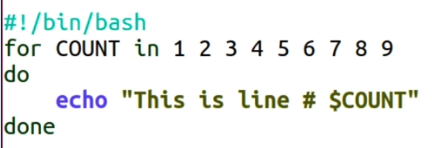
Va afficher :

Animal en cours d'analyse : chien

Animal en cours d'analyse : souris

Animal en cours d'analyse : moineau

Un autre exemple :



Va afficher :

This is line # 1

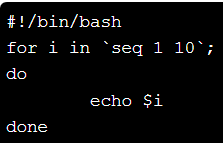
This is line # 2

This is line # 3

Etc..

**Un for plus classique**

Toutefois, en trichant un peu à l'aide de la commande seq, il est possible de simuler un for classique :



Explication: seq génère tous les nombres allant du premier paramètre au dernier paramètre, donc 1 2 3 4 5 6 7 8 9 10.

## Les fonctions

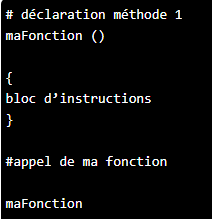
**Qu’est-ce qu’une fonction ?**

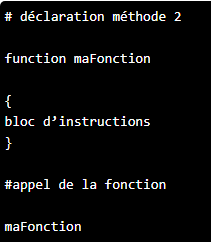
Une fonction, c’est un ensemble d’instructions, permettant d'effectuer plusieurs tâches avec des **paramètres d'entrée différents**.

Son utilisation vous permettra de rendre votre programme plus lisible et structuré. Ainsi, il facilitera le développement de votre programme.

Comment déclarer une fonction ?

Il y a deux manières :





On peut lui passer des paramètres comme les paramètres que vous passez à votre script.

Souvenez-vous  **$1 $2 $3 $n** (**$0** pour le nom du script) et  (**$#**  le nombre de paramètre passé au script) ...

o