**Linux Foundation Certified SysAdmin (LFCS)**

Length: 24:06:49

# Essential commands(25% of the Exam)

## Log into Local and Remote Graphical and Text Mode Consoles – Debian Version

Ctrl + Alt + t afin d'ouvrir le terminal.

Cliquer sur icône show applications and tape terminal

For a remote access :

ssh user@ip

Exemple :  ssh [pi@192.168.0.12](mailto:pi@192.168.0.12)

On peut aussi prendre la main avec VNC viewer, il faut créer un compte sur Real VNC puis cela te permettra de prendre la main sur un ordinateur en indiquant son nom ou IP.

## Search for files

Linux case is sensitive

find / -name "test.txt"

Everything in Linux is a file.

Pour ignorer la case:

find / -iname "test.txt"

Pour chercher tout ce qui est n'est pas "test.txt" par exemple :

find / -not -name "test.txt"

Pour chercher tous les dossiers :

find / -type d

Pour chercher les dossiers intitulés "log"

find / -type d -name "log"

Pour chercher tous les files qui se terminent par log :

find / -type f -name "\*.log"

Chercher un fichier de plus de 27 k :

find /usr/bin -size +27000c

find /sur/bin -size +27k

Chercher les fichiers créés il y a plus d'un jour:

find / -type f -mtime 1

Chercher les fichiers créés il y a moins d'un jour:

find / -type f -mtime -1

Chercher tous les fichiers appartenant à Qeyss:

find / -user qeyss

Chercher tous les fichiers du dossier etc appartenant à l'utilisateur root:

find /etc -user root

Chercher tous les fichiers par des permissions:

find /usr/bin -perm 755

Chercher tous les fichiers et modifier la permission:

find / -name "test.txt" -exec chmod 700 {} \ ; ( ici on donne full permission au propriétaire et rien aux autres.)

ls -la permet d’avoir la liste des fichiers, leurs permissions, tailles, etc.

Which command

Afin de savoir où se trouve l'exécutable d'une commande, ex :

which ifconfig

which python

 locate command

La commande locate permet de trouver très rapidement un fichier. Contrairement à ce que l’on pourrait penser **locate** ne vas pas chercher le fichier au sein de l’arborescence, mais au sein d’une base de données contenant la liste des fichiers existants. De ce fait, il se peut que lorsque vous venez de créer un fichier ou un répertoire que ce dernier ne soit pas dans cette base de données, il faut pour cela réactualiser la base de données à l’aide de la commande **updatedb**.

## Evaluate and compare the Basic file system features and options.

A file system is where computer system persists general data for users and/or applications

Les **systèmes de fichiers** (il en existe de multiples) gèrent l'organisation des informations mémorisées sur les périphériques de stockage de l'ordinateur

Dans le cœur d'un ordinateur, tout est constitué de 1 et de 0, mais l'organisation de ces données n'est pas aussi simple.

Un **bit** est un 1 ou un 0 ; un **octet** (byte en anglais) est composé de huit bits ; un kilo-octet binaire (**kibioctet**, ou kibibyte) est un groupe de 1024 octets ; un méga-octet binaire(**mébioctet**, ou mebibyte) se constitue de 1024 kilo-octets binaires ; et ainsi de suite.

Un disque dur (hard drive ou hard disk) stocke vos données : à chaque fois que vous enregistrez un fichier, vous écrivez des milliers de 1 et de 0 sur un disque métallique, le piquant littéralement et créant une indentation qui pourra plus tard être relue en tant que 1 ou 0 par votre ordinateur. En réalité, vous magnétisez des milliers de petits bâtons (que nous appellerons clusters) qui seront polarisés dans un sens ou dans un autre (N-S ou S-N) ; cette polarisation différente sera ré-interprétée plus tard par l'ordinateur en tant que 1 ou 0.

Il y a tellement de données sur un disque dur qu'il doit obligatoirement y avoir un moyen de les organiser.

Les systèmes de fichiers organisent les fichiers de votre ordinateur sur votre disque dur de façon à pouvoir les retrouver lorsque vous en aurez besoin. Les systèmes de fichiers les plus répandus à l'heure actuelle sont sûrement le FAT32 et le NTFS, qui sont les deux seuls systèmes de fichiers que Microsoft® Windows® peut nativement lire. Mais, tout comme il existe d'autres systèmes pour classer des livres dans une bibliothèque, il existe de nombreux autres systèmes de fichiers : ext2, ext3, ext4, ReiserFS, JFS, XFS, Btrfs, ZFS,….

Les qualifications d’un système de fichiers :

La taille maximale d’un fichier

La taille maximale d’une partition

La gestions des droits d’accès aux fichiers et répertoires

La journalisation :

Parlons maintenant de la journalisation. Un système de fichiers journalisé est **plus fiable** lorsqu'on entre dans le domaine du stockage des données. Il a été expliqué plus haut ce qui se produit réellement lorsqu'un fichier est enregistré sur un disque dur (une suite de 1 et de 0 est inscrite sur le disque) ; mais que se produit-il si l'écriture de la chaîne est interrompue avant son terme (ce qui se produit, par exemple, lors d'une coupure de courant) ? Votre fichier devient « corrompu », incomplet.

Un système de fichiers journalisé travaille de façon à prévenir une telle corruption : lors de la sauvegarde d'un fichier, au lieu d'écrire immédiatement sur le disque dur les données à l'endroit exact où elles devraient être enregistrées, le système de fichiers écrit les données dans une autre partie du disque dur et note les changements nécessaires dans un journal, et ensuite, en arrière-plan, il repasse chacune des entrées du journal et termine le travail commencé ; lorsque la tâche est accomplie, il raye la tâche de la liste.

EXT ( Extended File System)

 BtrFS, XFS, HPFS, SWAP, FAT(16, 32, exFAT)

## Compare and manipulate File Content and Use Input Output redirection

**Commande Cat**

Permet de voir le contenu d’un fichier sans l’ouvrir

cat ShoppingList.txt

**Commande More**

Permet de voir le contenu d’un fichier page par page sans l’ouvrir.

**Commande Less**

Permet de voir le contenu d’un fichier page par page ( s’il est long) sans l’ouvrir tout en pouvant revenir en arrière

less ShoppingList.txt

**Commande Sort**

Permet de trier l’ensemble des lignes du fichier par ordre croissant.

sort ShoppingList.txt

On peut combiner deux commandes :

sort ShoppingList.txt | more

**Commande Man**

Te donne des infos sur une commande, ex : man sort

Redirection

>

Permet de rediriger le résultat d’une commande dans un autre fichier ou une autre commande.

Exemple : ls -la > test.txt

>> : redirige à la fin d'un fichier et le crée s'il n'existe pas

**Commande cut**

Permet de couper un fichier selon le nombre de caractères ou un délimiteur

Exemple de fichier intitulé notes.txt :

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

Coupure selon le nombre de caractères :

cut -c -3 notes.txt

Va afficher :

Fab

Mat

Sop

Mél

Cor

Alb

Ben

Coupure selon un délimiteur :

cut -d , -f 1 notes.txt

Là on demande à couper en prenant pour délimiteur la virgule et à n’afficher que les premiers termes avant la virgule soit :

Fabrice

Mathieu

Sophie

Mélanie

Corentin

Albert

Benoît

Si je veux afficher que les commentaires donc situés après la troisième virgule on tape :

cut -d , -f 3 notes.txt

Excellent travail

Nul comme d'hab

En nette progression

Allez presque la moyenne !

Pas mal mais peut mieux faire

Toujours parfait

En grave chute

Si je veux afficher les deux premiers éléments je tape :

cut -d , -f 1,2 notes.txt

Fabrice,18 / 20

Mathieu,3 / 20

Sophie,14 / 20

Mélanie,9 / 20

Corentin,11 / 20

Albert,20 / 20

Benoît,5 / 20

## Analyse text using basic regular expressions

Nous allons avoir besoin des deux symboles suivants, retenez-les :

* ^(accent circonflexe) : indique le début d'une chaîne ;
* $(dollar) : indique la fin d'une chaîne.

Ainsi, si vous voulez qu'une chaîne commence par « Bonjour », il faudra utiliser la regex ( expression régulière) :

'^Bonjour '

Si vous placez le symbole « ^ » devant le mot, alors ce mot devra obligatoirement se trouver au début de la chaîne, sinon on vous répondra FAUX.

De même, si on veut vérifier que la chaîne se termine par « zéro », on écrira cette regex :

'zéro$ '

Si je cherche toutes les lignes commençant par The dans le texte Alice.txt, je taperais :

grep '^The' Alice.txt

Les classes de caractères

Si je veux chercher les lignes commençant par un a, b ou c il faut mettre cela entre parenthèse [abc] ou [a-c], exemple :

grep '^[a-c]' Alice.txt

grep '^T[a-z][aeiouy]' Alice.txt

Là je cherche les lignes commençant par un T, suivi d’une lettre, suivie d’une voyelle.

#### Et pour dire que je n'en veux pas ?

Si vous ne voulez PAS des caractères que vous énumérez dans votre classe, vous devrez placer le symbole « ^ » au début de la classe.

Ainsi, la regex suivante :

'[^0-9]'

… signifie que vous voulez que votre chaîne comporte au moins un caractère qui ne soit pas un chiffre.

grep '^T[a-z][^e]' Alice.txt

Cela signifie toutes les lignes commençant par un T, suivi d’une lettre et dont le troisième caractère n’est pas un e.

Les quantificateurs

* + ?(point d'interrogation) : ce symbole indique que la lettre est facultative. **Elle peut y être 0 ou 1 fois**.

Ainsi,#a?#reconnaît 0 ou 1 « a » ;

* + +(signe plus) : la lettre est obligatoire. **Elle peut apparaître 1 ou plusieurs fois**.

Ainsi,#a+#reconnaît « a », « aa », « aaa », « aaaa », etc. ;

* + \*(étoile) : la lettre est facultative. **Elle peut apparaître 0, 1 ou plusieurs fois**.

Ainsi,#a\*#reconnaît « a », « aa », « aaa », « aaaa », etc. Mais s'il n'y a pas de « a », ça fonctionne aussi !

Les quantificateurs sont des symboles qui permettent de dire combien de fois peuvent se répéter un caractère ou une suite de caractères.

La regex#^Yaho+$#signifie que la chaîne doit commencer et finir par le mot « Yahoo ». Il peut y avoir un ou plusieurs « o ». Ainsi « Yaho », « Yahoo », « Yahooo », etc. marchent…

Les accolades

Parfois on aimerait indiquer que la lettre peut être répétée quatre fois, ou de quatre à six fois… bref, on aimerait être plus précis sur le nombre de répétitions.  
C'est là qu'entrent en jeu les accolades.

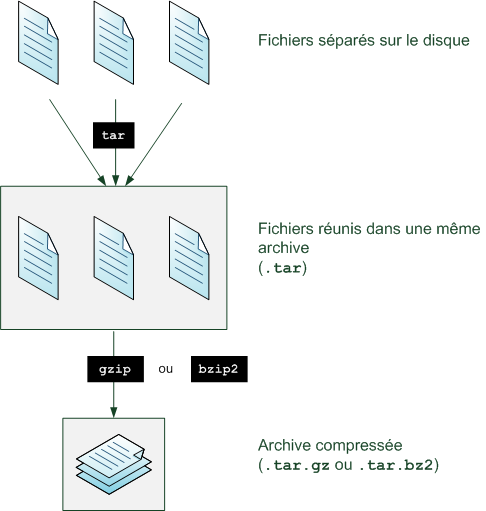
Il y a trois façons d'utiliser les accolades.

* {3} : si on met juste un nombre, cela veut dire que la lettre (ou le groupe de lettres s'il est entre parenthèses) doit être répétée **3 fois exactement**.  
  #a{3}#fonctionne donc pour la chaîne « aaa ».
* {3,5} : ici, on a plusieurs possibilités. On peut avoir la lettre de **3 à 5 fois**.  
  #a{3,5}#fonctionne pour « aaa », « aaaa », « aaaaa ».
* {3,} : si vous mettez une virgule, mais pas de 2e nombre, ça veut dire qu'il peut y en avoir jusqu'à l'infini. Ici, cela signifie « **3 fois ou plus** ».  
  #a{3,}#fonctionne pour « aaa », « aaaa », « aaaaa », « aaaaaa », etc. Je ne vais pas tous les écrire, ça serait un peu long.

|  |  |  |
| --- | --- | --- |
| eeeee | #e{2,}# | VRAI |
| Blablablabla | #^Bla(bla){4}$# | FAUX |
| 546781 | #^[0-9]{6}$# | VRAI |

## Archive,backup, Compress,Unpack, and Decompress Files

Tar permet d'assembler des fichiers avant de les compresser.



1° On rassemble tous les fichiers à compresser dans un même dossier.

2° On crée une archive avec tar, exemple sur le dossier intitulé tuto

tar -cvf tuto.tar tuto/

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

3° On compresse avec gzip :

gzip tuto.tar

Tu auras donc un dossier tuto.tar.gz

Il y a aussi la commande bzip2 pour compresser :

bzip2 tuto.tar

4° Pour décompresser on utilise la commande gunzip :

gunzip tuto.tar.gz

5° Pour extraire les fichiers d’une archive on utilise la commande tar avec l’option -xvf

tar -xvf tuto.tar

Il est possible d’archiver et compresser en même temps avec la commande tar et l’option -zcvf :

tar -zcvf tuto.tar.gz tuto/

Pour décompresser et extraire directement on utilise la commande tar avec l’option -zxvf :

tar -zxvf tuto.tar.gz

On peut afficher le contenu d’une archive sans l’extraire :

On doit taper : tar -tf nomdelarchive.tar

Pour voir le contenu d’un fichier compressé on utilise la commande zcat

Ex : zcat tuto.tar.gz

## Create,Delete,Copy and move files and directories

Create with touch or vim and the name of the file

Copy with cp :

cp test test2 🡪 Va copier test dans un fichier test2

cp test text/ 🡪 va copier le fichier test dans le dossier text

Remove with mv

mv test2 text/ 🡪 Va déplacer le fichier test2 dans le dossier text

mv test2 text/test3 🡪 Va déplacer le fichier test2 dans le dossier text et le renommer en test3

Delete a file with rm

rm test2

Delete a directory with rm -r

rm -r text/

rm -rf text/ 🡪 va supprimer le dossier et tout ce qu’il contient.

## Create and Manage Hard and Soft Links

La commande ln permet de créer des liens entre fichiers.

Ces liens sont des raccourcis

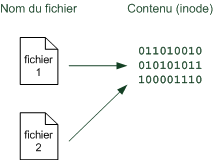
On peut créer des liens physiques ou symboliques

**Les liens physiques**

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).

Tu commences par créer un fichier1

Pour le créer tu fais : ln fichier1 fichier2, ça va automatiquement créer le fichier2



Ainsi, que vous passiez par fichier1 ou par fichier2, vous modifiez exactement le même contenu. En quelque sorte, **le fichier est le même**. On peut juste y accéder via deux noms de fichiers différents.

Si tu modifies n’importe lequel de ces deux fichiers, ça modifiera aussi l’autre.

En faisant un ls -i tu verras que les fichiers partagent le même inode.

**Les liens symboliques**

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 en créer un tu tapes : ln -s fichier1 fichier2

En faisant un ls -l tu verras que le fichier 2 au niveau des droits commence par l qui signifie link, ex :

-rw-r--r-- 1 mateo21 mateo21 0 2008-07-31 13:55 fichier1

lrwxrwxrwx 1 mateo21 mateo21 8 2008-07-31 14:15 fichier2 -> fichier1

Sinon ils sont similaires en tout aux liens physiques sauf que les liens symboliques fonctionnent aussi sur des répertoires, contrairement aux liens physiques.

## Lecture: List,set and Change Standard File Permissions

In Linux, everything is a file, so we can restrict based on some basic file permissions

On utilise la commande ls -la afin de voir les permissions des files and directories.

Sous Linux on ne se connecte en root que très rarement, **lorsque c'est nécessaire**. Certaines commandes de Linux que nous allons voir dans ce chapitre ne sont accessibles qu'à root.  
Le reste du temps, on utilise le compte « limité » que l'on a créé.

Cette simple protection permet de largement limiter les dégâts en cas de fausse manipulation, de virus sur votre PC, etc. En effet, un virus ne peut rien faire de plus que vous quand vous êtes connectés avec des droits limités. En revanche, si vous êtes en root il pourra tout faire, même détruire votre ordinateur.

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

Avec Ubuntu on ne peut pas se connecter en root au démarrage.

On peut s’y connecter temporairement à l’aide de la commande **sudo** en tapent sudo puis la commande souhaitée ou bien définitivement en tapant sudo su

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

Sous les autres distributions qu'Ubuntu, écrire « su » suffit à passer root.

**Gestion des users**

**adduser** pour ajouter un user

**deluser** pour supprimer un user

**passwd** pour changer le password, ex : passwd qeyss

Gestion des groupes

addgroup pour ajouter un groupe

delgroup pour supprimer un groupe

usermod pour modifier un user

**Les droits d’accès**

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.

Pour afficher les droits on tape **ls –l** et pour les modifier on utilise la commande **chmod**

mateo21@mateo21-desktop:~$ ls -l

total 40

drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 21:53 Desktop

drwxr-xr-x 2 mateo21 mateo21 4096 2007-11-13 13:46 Documents

lrwxrwxrwx 1 mateo21 mateo21   26 2007-09-19 18:31 Examples -> /usr/share/example-content

drwxr-xr-x 2 mateo21 mateo21 4096 2007-09-25 20:28 images

Vous voyez tous ces d, r, w et x au début ? Ce sont ce qu'on appelle les droits d'accès du fichier ou dossier.

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).

Les droits sont découpés en fonction des utilisateurs (figure suivante).



* Le premier triplet rwx indique les droits que possède le **propriétaire** du fichier sur ce dernier ;
* Le second triplet rwx indique les droits que possèdent les autres membres du **groupe** sur ce fichier ;
* Enfin, le dernier triplet rwx indique les droits que possèdent tous les **autres** utilisateurs de la machine sur le fichier.

| **Droit** | **Chiffre** |
| --- | --- |
| r | 4 |
| w | 2 |
| x | 1 |

Voici la liste des droits possibles et la valeur correspondante :

| **Droits** | **Chiffre** | **Calcul** |
| --- | --- | --- |
| --- | 0 | 0 + 0 + 0 |
| r-- | 4 | 4 + 0 + 0 |
| -w- | 2 | 0 + 2 + 0 |
| --x | 1 | 0 + 0 + 1 |
| rw- | 6 | 4 + 2 + 0 |
| -wx | 3 | 0 + 2 + 1 |
| r-x | 5 | 4 + 0 + 1 |
| rwx | 7 | 4 + 2 + 1 |

Par exemple, « 640 » indique les droits du propriétaire, du groupe et des autres (dans l'ordre).

* 6 : droit de lecture et d'écriture pour le propriétaire.
* 4 : droit de lecture pour le groupe.
* 0 : aucun droit pour les autres.

Pour changer les droits sur le fichier rapport.txt, et être le seul autorisé à le lire et l'éditer, je dois exécuter cette commande :

chmod 600 rapport.txt

##### Attribuer des droits avec des lettres (chmod relatif)

Il existe un autre moyen de modifier les droits d'un fichier. Il revient un peu au même mais permet parfois de paramétrer plus finement, droit par droit.  
Dans ce mode, il faut savoir que :

* **u** = user (propriétaire) ;
* **g** = group (groupe) ;
* **o** = other (autres).
* **a = all**

… et que :

* **+** signifie : « Ajouter le droit » ;
* **-** signifie : « Supprimer le droit » ;
* **=** signifie : « Affecter le droit ».

Maintenant que vous savez cela, vous pouvez écrire :

chmod g+w rapport.txt

Signification : « Ajouter le droit d'écriture au groupe ».

chmod g+w,o-w rapport.txt

Signification : « Ajouter le droit d'écriture au groupe et l'enlever aux autres ».

chmod go-r rapport.txt

Signification : « Enlever le droit de lecture au groupe et aux autres ».

chmod +x rapport.txt

Signification : « Ajouter le droit d'exécution à tout le monde ».

chmod a+rwx rapport.txt

Signification : « Ajouter tous les droits à tout le monde ».

#### Récursivement

Pour chacune de ces commandes, on peut les lancer récursivement sur un répertoire. C'est à dire que l'action sera effectuée sur le répertoire désigné et sur tous les fichiers ou répertoires qu'il contient. Ceci se fait en ajoutant l'option **-R** .

Gestion des propriétés d’un fichier

**chown** : Pour 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.

## Read, and Use System Documentation

Linux is an amazing operating system and the best part? The documentation is built right in!

We use the command man, like: man ls or info ls

/usr/share/doc contient aussi de la documentation.

## Manage Access to the Root Account

Root access on Linux machines means all the power to do whatever you need and whatever you want, and it's really important to limit who has access.

Pour supprimer tous les dossiers et fichiers d’un dossier je tape : rm -rf \*

Si la commande sudo ne fonctionne pas avec un user car il n’a pas les droits pour l’utiliser, il faut taper sudo visudo et le mettre dans la section où il y a root avec les mêmes droits que root.

La commande sudo !! permet de retaper la précédente commande mais avec sudo.

# Operation of Running Systems (20% of the Exam)

## Lecture: Boot, reboot, and shut down a system safely

We type **shutdown**

sudo shutdown –r : Pour faire un reboot

## Boot of changes system into different operating modes

Les runlevel sont les différents niveaux de démarrage/arrêt sous Unix/Linux.  
De petites différences existent suivant les différentes distributions. Je m'intéresse ici aux runlevel sous Debian GNU/Linux.

Il existe donc 6 runlevel sous Debian GNU/Linux:  
Le runlevel 0 correspond à l'arrêt du système  
Le runlevel 1 correspond au démarrage single-user  
Le runlevel 2 correspond mode de démarrage normal  
Les runlevels 3-5 correspondent à des modes de démarrage que l'on peut se configurer  
Le runlevel 6 correspond au redémarrage.

Les runlevel sont des niveaux constitués d'un ensemble de liens situés dans les répertoires /etc/rcX.d/ qui pointent vers les scripts situés dans /etc/init.d/

Au démarrage de Linux, le noyau est chargé puis il passe la main au processus INIT.  
Ce processus démarre tout ce qui se trouve dans /etc/rcS.d/ puis il lance par défaut le runlevel 2 (on peut changer cela dans le fichier /etc/inittab). Par exemple, si l'on veut démarrer en mode single-user (runlevel 1) (souvent en cas de problème du système), on tape **Linux 1** au prompt de LILO.

En tapant runlevel tu auras le level sur lequel tu es actuellement

## Install, configure and troubleshoot bootloaders

There's a lot that needs to happen from the moment when the power is turned on until the system decides which operating system to boot. That's where bootloaders come in! In this lesson, we'll take a close look at the bootloader that most Linux distributions use today -- Grub 2. We'll look at the files that make it up and discuss how you can install, configure, and troubleshoot it. At the end of this lesson, you'll understand how to configure grub2 and boot one or more operating systems on a machine!

***GR****and* ***U****nified* ***B****ootloader, version 2* (ou **GRUB 2**) est la seconde version de GNU GRUB. C'est un logiciel permettant de [charger un système d'exploitation](https://doc.ubuntu-fr.org/amorcage). Il pourra amorcer lui-même des systèmes compatibles avec la norme POSIX (GNU/Linux, \*BSD, Mac OS, etc.) ou enchaîner vers un autre chargeur d’amorçage pour les autres systèmes.

Il est dans /etc/grub.d

Pour changer l’ordre de changement du boot faut aller dans /etc/default/grub

GRUB 2 est le chargeur d'amorçage installé par défaut, ce document concerne la version de GRUB 2 pour les ordinateurs Compatibles PC (la plupart des ordinateurs actuels)

Pour connaitre sa version d’installée on tape : grub-install –V

Utilisation du menu

**Si vous n'avez qu'un seul système d'installé, Grub ne s'affiche pas !** Pour le voir, il suffit de maintenir la **touche Majuscule (shift)** ( ou la touche[**ECHAP**](https://forum.ubuntu-fr.org/viewtopic.php?pid=21778237#p21778237)) enfoncée **au démarrage** de l'ordinateur.

**Fichier grub.cfg**

* /boot/grub/**grub.cfg** : est un fichier de configuration généré automatiquement par *update-grub* (il est inutile d’éditer ce fichier) et **ne doit** donc **pas être modifié manuellement**.

**Fichier /etc/default/grub**

Dans ce fichier, on peut **activer ou désactiver un paramètre** en le commentant / décommentant avec le caractère **dièse « # »**([croisillon](https://fr.wikipedia.org/wiki/wiki)).

**paramètres présents par défaut**

* **GRUB\_DEFAULT=0** correspond au menu qui sera sélectionné par défaut. (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#menu_par_defaut))
* **#GRUB\_HIDDEN\_TIMEOUT=0** : avec le « # », le menu de grub sera visible. Sans le « # », le menu de grub sera invisible. (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#affichermasquer_le_menu))
* **GRUB\_HIDDEN\_TIMEOUT\_QUIET=false** De pair avec ‘GRUB\_HIDDEN\_TIMEOUT’. (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#affichermasquer_le_menu))
* **GRUB\_TIMEOUT\_STYLE=hidden** Paramètre apparu avec la version 18.04.1 en remplacement des deux qui précèdent.
* **GRUB\_TIMEOUT=10** est la durée en secondes de l’affichage du menu avant de se lancer sur le menu sélectionné par défaut. (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#menu_par_defaut))
* **GRUB\_DISTRIBUTOR=``** est la ligne qui définit la syntaxe des titres du menu (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#intitule_dans_le_menu))
* **GRUB\_CMDLINE\_LINUX\_DEFAULT=""** et **GRUB\_CMDLINE\_LINUX=""** on y met les paramètres à ajouter lors de la détection automatique des systèmes (ex: radeon.modeset=1, logo.nologo,..).
* **#GRUB\_GFXMODE=640x480** (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#grub_gfxmode)).
* **#GRUB\_INIT\_TUNE="480 440 1"** : sans le « # », on aura un bip à l’affichage du menu Grub.

**Paramètres qu'on peut ajouter**

* **GRUB\_GFXPAYLOAD=1024x768** : définit la résolution d’affichage entre Grub et celle définie dans la distribution (pour le splash screen par exemple). Défini à 'keep' par défaut.
* **GRUB\_GFXPAYLOAD\_LINUX=auto** : définit la résolution d’affichage par défaut du linux lancé FIXME
* **GRUB\_BACKGROUND="/boot/grub/images/fjord.jpg"** : pour mettre une image en fond d'écran (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#fond_d_ecran))
* **GRUB\_DISABLE\_OS\_PROBER="true"** : si on souhaite désactiver la recherche d'autres systèmes à chaque update-grub (inutile si on les lance autrement)
* **GRUB\_OS\_PROBER\_SKIP\_LIST="…"** : liste de partitions à ne pas inclure dans la recherche automatique de systèmes. (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#os-prober_partiel))
* **GRUB\_DISABLE\_LINUX\_RECOVERY="true"** : on mettra ce paramètre si on ne veut plus avoir la possibilité de lancer les systèmes en mode 'maintenance'
* **GRUB\_DISABLE\_SUBMENU=y** : si on souhaite voir directement une entrée de menu par noyau linux disponible (On peut supprimer les [noyaux](https://doc.ubuntu-fr.org/kernel) les plus anciens → [Nettoyer Ubuntu](https://doc.ubuntu-fr.org/kernel#suppression_des_anciens_noyaux))
* **LANG=fr\_FR** : pour définir la langue utilisée pour l'édition du menu depuis Grub ou le mode 'ligne de commande'
* **GRUB\_TERMINAL\_INPUT=at\_keyboard** : nécessaire pour paramétrer le clavier français (pour ne plus avoir à chercher les ()[]= .. du clavier Qwerty). (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#clavier_francais))
* **GRUB\_THEME="/boot/grub/themes/ubuntu-mate/theme.txt"** : chemin vers le fichier de configuration du thème (mode graphique). (→ [**Détails**](https://doc.ubuntu-fr.org/tutoriel/grub2_parametrage_manuel#theme))

Top of Form

Bottom of Form

**Dossier /etc/grub.d/**

Ce dossier contient tous les scripts qui seront utilisés (en respectant l'ordre de numérotation) par update-grub pour créer le fichier grub.cfg.

* **00\_header** : script gérant les paramètres définis dans /etc/default/grub ;
* **05\_debian\_theme** : script pour gérer le thème en mode texte (fonds d'écran et couleurs) ;
* **10\_linux** : contient le script de lancement du système sur lequel on est ;
* **20\_memtest86+** : script permettant de générer les entrées memtest ; semble absent de certaines installations EFI
* **20\_linux\_xen** : script pour Xen Linux et Xen Hypervisor ;
* **30\_os-prober** : contient le script de recherche des autres systèmes installés ;
* **30\_uefi-firmware** FIXME
* **40\_custom** : configuration personnelle (systèmes à lancer en configuration manuelle, paramètres qui n'existent pas pour /etc/default/grub,..)
* **41\_custom** : identique à 40\_custom (si on le renomme par exemple 07\_custom, on pourra y mettre le lancement du mediacenter ou de Windows en premier sur la liste)

**Nous n'utiliserons** pour nos paramétrages **que les fichiers 'custom'**.

Paramétrage manuel

Avant de commencer, il est souhaitable :

* de lancer dans un [terminal](https://doc.ubuntu-fr.org/terminal) une [commande](https://doc.ubuntu-fr.org/commande_shell) de mise à jour : *sudo update-grub* ;
* de créer une [sauvegarde](https://doc.ubuntu-fr.org/tutoriel/comment_sauver_et_restaurer_un_fichier) (ex. : *sudo cp /boot/grub/grub.cfg /boot/grub/grub.cfg.autogénéré*) ;

De même, aucun script de configuration présent dans **/etc/grub.d/** ne doit être effacé.

**Menu par défaut**

On utilisera ici les paramètres **GRUB\_DEFAULT** et **GRUB\_TIMEOUT** du fichier **/etc/default/grub**

Pour changer le système d'exploitation sur lequel grub démarre par défaut on paramètrera **GRUB\_DEFAULT=** au choix avec :

* un **chiffre** (0 = le premier, 1 = le deuxième, etc.) ;
* **saved** (sans guillemet) pour ce qui a été sélectionné lors du précédent lancement (nécessite de rajouter GRUB\_SAVEDEFAULT=true) ;
* ou le **titre** de menu exact d’un menu mis **entre guillemets** (par exemple : "Microsoft Windows 10 (on /dev/sda1)"). Pour obtenir les titres exacts des menus : grep menuentry /boot/grub/grub.cfg

La **temporisation** avant un lancement automatique se définit avec le paramètre **GRUB\_TIMEOUT** :

* **GRUB\_TIMEOUT=10**, lancera le système par défaut au bout de **10 secondes**. Si on ne veut **pas de temporisation**, on mettra : **GRUB\_TIMEOUT=-1**

## Diagnose and manage processes

Knowing what is running on your Linux system, as well as what's very busy and what is not so much, is a critical aspect of system administration.

**w** – Show who is logged on and what they are doing

**top** – Display linux processes

Pour faire simple, dites-vous qu'un **processus** est un programme qui tourne en mémoire. La plupart des programmes ne font tourner qu'un processus en mémoire (une seule version d'eux-mêmes). C'est le cas d'OpenOffice par exemple. D'autres lancent des copies d'eux-mêmes, c'est le cas du navigateur Google Chrome qui crée autant de processus en mémoire que d'onglets ouverts

Pour lister les processus qui tournent sous Windows, on utilise Ctrl + Alt + Suppr et on va dans l'onglet « Processus ».  
Sous Linux, on peut utiliser deux commandes différentes : ps et top

**ps** vous permet d'obtenir la liste des processus qui tournent au moment où vous lancez la commande. Cette liste n'est pas actualisée en temps réel, contrairement à ce que fait top

On distingue quatre colonnes.

* PID : c'est le numéro d'identification du processus. Chaque processus a un numéro unique qui permet de l'identifier. Ce numéro nous sera utile plus tard lorsque nous voudrons arrêter le processus.
* TTY : c'est le nom de la console depuis laquelle a été lancé le processus.
* TIME : la durée d'exécution du processus. Plus exactement, cela correspond à la durée pendant laquelle le processus a occupé le processeur depuis son lancement.
* CMD : le programme qui a généré ce processus. Si vous voyez plusieurs fois le même programme, c'est que celui-ci s'est dupliqué en plusieurs processus (c'est le cas de MySQL, par exemple)

Quand on utilise ps sans argument comme on vient de le faire, il affiche seulement les processus lancés par le même utilisateur.

**ps -ef : lister tous les processus**

Avec ps -ef, vous pouvez avoir la liste de tous les processus lancés par tous les utilisateurs sur toutes les consoles

**ps -u UTILISATEUR : lister les processus lancés par un utilisateur**

Pour filtrer un peu cette longue liste, on peut utiliser -u afin d'obtenir par exemple uniquement les processus que l'on a lancés nous-mêmes.

**top : liste dynamique des processus**

La liste donnée par ps a un défaut : elle est **statique** (elle ne bouge pas). Or, votre ordinateur, lui, est en perpétuel mouvement. De nombreux processus apparaissent et disparaissent régulièrement.

**Ctrl +C & kill** : arrêter un processus

**Ctrl + C : arrêter un processus lancé en console**

Taper **Ctrl + C** ne coupe pas le programme brutalement, cela lui demande gentiment de s'arrêter, comme si vous aviez cliqué sur la croix pour fermer une fenêtre.

**kill :** tuer un processus

Ctrl + C ne fonctionne que sur un programme actuellement ouvert dans la console. De nombreux programmes tournent pourtant en arrière-plan, et Ctrl + C n'aura aucun effet sur eux.

C'est là que vous devez utiliser kill si vous voulez les arrêter

Pour vous en servir, il faudra auparavant récupérer le PID du ou des processus que vous voulez tuer. Pour cela, deux solutions :

* ps ;
* top.

Exemple si je veux tuer le process firefox lance par l’utilisateur qeyss, je recherché son PID :

root@ubuntu:~# ps -u qeyss | grep firefox

14692 tty2 00:00:21 firefox

root@ubuntu:~# kill 14692

Vous pouvez aussi tuer plusieurs processus d'un seul coup en indiquant plusieurs PID à la suite :

kill 32678 2768 33071

Vous voulez tuer un processus sans lui laisser le choix ?  
C'est tout à fait possible, mais à n'utiliser que dans le cas d'un programme complètement planté que vous voulez vraiment arrêter !

Avec kill -9 (comme le chiffre 9, oui, oui), vous demandez à Linux de tuer le processus sans lui laisser le temps de s'arrêter proprement. Cela peut faire le ménage quand rien ne va plus.

kill -9 32678

**killall :** tuer plusieurs processus

Souvenez-vous : je vous ai dit que certains programmes se dupliquaient en plusieurs processus. Si vous voulez arrêter l'ensemble de ces processus, comment faire ? Heureusement, vous avez des armes pour éradiquer cette vermine.

Vous pourriez, certes, tuer tous les processus en récupérant un à un leur PID. Mais il y a plus rapide : killall (« tuez-les tous ! »).

Contrairement à kill, killall attend le nom du processus à tuer et non son PID.

Supposons que nous ayons trois processus find en cours d'exécution que nous souhaitions arrêter.

$ ps -u mateo21 | grep find

  675 pts/1    00:00:01 find

  678 pts/2    00:00:00 find

  679 pts/3    00:00:01 find

Pour tous les tuer, il faudra donc taper :

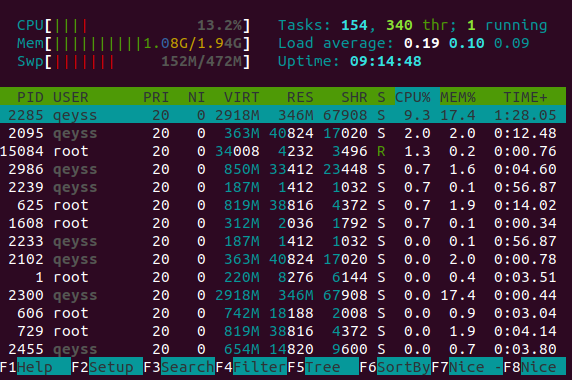
$ killall find

**htop** – Display linux processes

htop permet de visualiser, je dirais même de gérer les processus de manière interactive.

Grâce à lui vous pourrez rechercher, tuer les processus et mêmes les trier selon un critère (utilisation de mémoire, de CPU,...), et d'[autres](https://www.it-connect.fr/cours-tutoriels/administration-systemes/serveur-web/autres-serveur-web/) fonctions disponible toute via l'interface de htop.

Il faut l’installer avec : apt-get install htop



Dans la partie haute de l'interface on peut voir l'utilisation du **CPU** en pourcentage, l'utilisation de la mémoire en Mo par rapport à la totalité de mémoire disponible ainsi que la mémoire swap utilisée. Le temps depuis lequel le [système](https://www.it-connect.fr/cours-tutoriels/administration-systemes/windows-server/systeme/) est démarré est indiqué également. Ensuite, un "**tableau**" listant les processus est présenté où vous pouvez retrouver une multitude d'information comme l'utilisateur qu'il l'a exécuté, le pourcentage de mémoire et de **CPU** qu'il utilise ainsi que la commande qui sert à l'exécuter. Pour finir, dans la partie basse un menu explique les différentes actions qui sont disponibles.

**Trier les processus selon un critère**

Pour cela appuyez sur "**F6**" ce qui correspond à "**SortBy**" soit "**Trier par**". Un listing de critères apparaît sur la gauche, choisissez celui que vous souhaitez



**Rechercher un processus**

Pour recherche un processus appuyez sur **F3** puis saisissez le nom ou une partie du nom de celui-ci :

**Tuer un processus**

Positionnez-vous sur le processus que vous désirez "killer" puis appuyez sur F9, ensuite choisissez une méthode de **KILL** dans la liste puis faites Entrée.

**Voir processus d'un utilisateur en particulier**

Plutôt que de voir l 'ensemble des processus, si vous préférez voir les processus exécutés par un utilisateur en particulier, utilisez la commande :

htop -u nom\_utilisateur

## Locate and analyze system log files

When something goes wrong, you need to know where to look to fix it! In this lesson, we'll explore the main system log files on Centos/Redhat systems and Debian/Ubuntu systems and take a look at a few tips and tricks to making reviewing the log files a bit easier. At the end of this lesson, you'll know where the main system logs are and have a good idea for where to find application-specific logs.

cd /var/log

Any time you have any kind of problems in your Linux system, the log files are generally a good place to start, pour les trouver :

Sur Debian/Ubuntu tu tapes : sudo less syslog

Sur Rhel/Centos tu tapes : sudo less messages

Si je cherche une erreur spécifique je peux taper ceci : grep "Error" syslog

root@ubuntu:/var/log# grep "Error" syslog

Dans Debian/Ubuntu il y a plusieurs fichiers de logs.

## Schedule tasks to run at a set date and time

Rappel :

Par défaut, un processus est lancé dans l'état running à l'avant-plan. On peut l'arrêter avec la combinaison Ctrl + C, auquel cas il sera détruit.

Mais on peut aussi l'envoyer en arrière-plan. Si on l'exécute dès le départ avec un&, il sera à l'étatrunningà l'arrière-plan. Si on choisit de faire Ctrl + Z, il passera à l'état Stoppedà l'arrière-plan. Il faudra taperbgpour le faire passer à nouveau à l'état running en arrière-plan.

Enfin, la commande fg renvoie un processus de l'arrière-plan vers l'avant-plan.

**Exécuter une seule tâche à une heure precise**

On utilise la commande : **at** 🡪 at 15:12

touch fichier.txt

Autre exemple:

* + at 15:13 tomorrow
  + at 15h15 23/05/21

Chaque fois qu’une commande est enregistrée “at” nous indique un numéro de job

Pour supprimer un job on tape: atrm NuméroJob

Pour liser la liste des jobs on tape: atq

**Exécuter une tâche régulièrement**

On utilise la commande **crontab** pour programmer l’exécution régulière d’un programme.

crontab –l permet de lister ce qui aurait été programmé.

crontab est en fait une commande qui permet de lire et de modifier un fichier appelé la « crontab ».  
Ce fichier contient la liste des programmes que vous souhaitez exécuter régulièrement, et à quelle heure vous souhaitez qu'ils soient exécutés.

Il y a une crontab par user.

Pour modifier la crontab on, tape: **crontab -e**

Les champs:

* m : minutes (0 - 59) ;
* h : heures (0 - 23) ;
* dom (*day of month*) : jour du mois (1 - 31) ;
* mon (*month*) : mois (1 - 12) ;
* dow (*day of week*) : jour de la semaine (0 à 6, 0 étant le dimanche) ;
* command : c'est la commande à exécuter.

Imaginons que je veuille exécuter une commande tous les jours à 15 h 47 qui va créer un fichier. Je vais écrire ceci :

47 15 \* \* \* touch/home/qeyss/fichier.txt

0 5 \*\* 1 = Tous les lundis à 5h du matin

Exemple je créé un dossier scripts dans lequel je vais faire un script bash intitule line.sh où j’écrirais ceci :

#!/bin/bash

echo "A dark vehicle was seen leaving the scene" >> /home/cloud\_user/crime

Ensuite il faut render ce fichier executable avec chmod +x line.sh

Puis je vais ouvrir crontab avec crontab –e

Dans lequel j’ajouterais cette ligne :

35 15 \* \* \* /home/cloud\_user/scripts/line.sh

Ainsi mon script se lancer tous les jours à 15h35 et son role sera d’ajouter une ligne au fichier crime

## Verify completion of scheduled jobs

Sur Debian/Ubuntu tu iras dans cd /var/logs où tu feras un cat syslog | grep CRON

## Lecture: Update software to provide required functionality and security & manage software, Part 1 - Ubuntu/Debian

We'll take a close look at dpkg, apt-get, and aptitude. At the end of this lesson, you'll feel comfortable updating, installing, and uninstalling packages on Ubuntu and other Debian systems.

Aptitude

**Aptitude** est un gestionnaire de paquets basé sur l'infrastructure [APT](https://doc.ubuntu-fr.org/apt), c'est-à-dire que vous pourrez installer, supprimer et mettre à jour les applications ([paquets](https://doc.ubuntu-fr.org/apt#le_paquet)) avec Aptitude. Il présente des fonctionnalités équivalentes à *dselect* ou [*apt-get*](https://doc.ubuntu-fr.org/apt-get). Il y a deux façons d'utiliser Aptitude :

* d'une façon semblable à [apt-get](https://doc.ubuntu-fr.org/apt-get) ;
* avec une interface interactive.

**Aptitude** n'est pas installé par défaut dans Ubuntu, il faut donc faire un : sudo apt-get install aptitude

**Installation**

Pour installer un paquet ainsi que ses dépendances "automatiques" tapez dans un [terminal](https://doc.ubuntu-fr.org/terminal) la ligne suivante à l'invite de commande:

sudo aptitude install <paquet-a-installer>

**Suppression**

Pour supprimer un paquet, ainsi que ses dépendances "automatiques" devenues inutiles :

sudo aptitude remove <paquet-a-supprimer>

**Suppression + Purge**

Pour désinstaller un paquet et supprimer ses fichiers de configuration :

sudo aptitude purge <paquet-a-supprimer>

**Mise-à-jour dépôts**

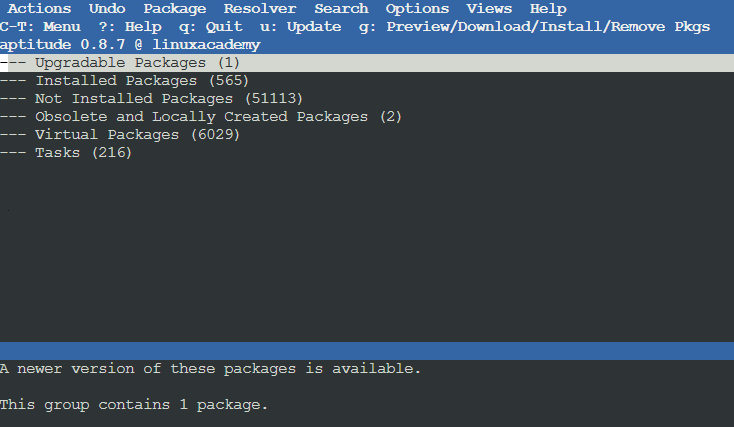
Pour avoir une version à jour de la liste des paquets disponibles, à faire avant toute installation :

sudo aptitude update

**Interface interactive**

Aptitude est intéressant parce qu'on peut l'utiliser de façon interactive. Pour commencer, lancez Aptitude :

sudo aptitude



**dpkg : le gestionnaire de paquets bas niveau de Ubuntu**

**dpkg** (pour **D**ebian **p**ac**k**a**g**e) est l'outil de bas niveau gérant les paquets des distributions basées sur Debian. Indispensable au bon fonctionnement d'Ubuntu, cet outil est intégré de base dans toutes ses [versions](https://doc.ubuntu-fr.org/versions) et [variantes](https://doc.ubuntu-fr.org/variantes).  
Utilisable en [ligne de commande](https://doc.ubuntu-fr.org/commande_shell), il est chargé de l'installation, la suppression et la gestion des paquets Debian (*.deb*), le type de paquets traités par Ubuntu. Il permet aussi la gestion de paquets Debian en provenance de sources extérieures aux [dépôts APT](https://doc.ubuntu-fr.org/depots).

À la différence de la commande [apt-get](https://doc.ubuntu-fr.org/apt-get), de la [Logithèque](https://doc.ubuntu-fr.org/software-center), ou de [GDebi](https://doc.ubuntu-fr.org/gdebi), **dpkg** est un outil qui ne gère pas les dépendances. Ainsi en cas de conflit ou bien lorsque seuls certains paquets impliquant trop de dépendances font défaut, l'utilisation de cet outil devient presque indispensable

**Installation de paquets**

Quelques [commandes](https://doc.ubuntu-fr.org/commande_shell) utilisées pour installer les paquets:

* L'option -i, ou --install, installe le ou les paquets indiqués, au format .deb.

sudo dpkg -i <paquet.deb>

* L'option -R, ou --recursive (associée à -i), installe tous les paquets se trouvant dans le répertoire de travail et ses sous-répertoires:

sudo dpkg -i -R \*.deb

**dpkg** ne gère pas les dépendances, il faut ensuite, pour compléter l'installation d'une application possédant des dépendances, exécuter dans un [terminal](https://doc.ubuntu-fr.org/terminal) la [commande](https://doc.ubuntu-fr.org/commande_shell):

sudo apt-get -f install

**Suppression**

* sudo dpkg -r nom\_du\_paquet

**Vérifier la présence d'un paquet**

utiliser dpkg -l comme ci-dessus conduit à une liste très longue. Si vous souhaitez simplement vérifier la présence d'un paquet ajouter tout ou partie du nom du paquet éventuellement suivi (voir précédé) de \* :

dpkg -l nom\_du\_paquet

exemple :

dpkg -l \*office\*

liste tous les paquets liés à libreoffice et openoffice

## Lecture: Update software to provide required functionality and security & manage software, Part 2 - Redhat/Centos

**Centos** utilise les fichiers dits « [RPM](http://www.rpm.org/) » (RPM Package Manager) qui sont des archives contenant des programmes pré-compilés prêts à l'emploi. Évidemment, il est possible comme sur certains systèmes non communautaires de les chercher un par un sur internet et de les installer manuellement. Mais la gestion des dépendances peut quelquefois devenir un véritable casse-tête (un logiciel qui en nécessite un autre, et un autre...).

Pour gérer les paquets du système, Centos utilise [YUM](http://linux.duke.edu/projects/yum/), un outil permettant de gérer les installations, les désinstallations et les mises à jour de paquets au format **RPM**. Il gére les dépendances en téléchargeant ce qui est nécessaire. Il trouve les paquets sur différentes sources (sites internet) que l'on appelle des **dépôts**.

**YUM** est fourni en standard dans toutes les versions de **Centos et Fedora**.

**RPM** est l'outil de base permettant d'installer des logiciels préparés au format « RPM ». **Yum** est d'une certaine façon une surcouche à rpm facilitant la vie de l'utilisateur. Téléchargement de paquets depuis des dépôts, recherche des dépendances, etc. **Yum** n'est pas le seul logiciel permettant ce genre de choses,

yum update : Will just updates the lists and the packages without software

Yum utilise des dépôts, ces dépôts regroupent un ensemble de logiciels. Chaque dépôt doit être ajouté à la configuration de YUM pour être utilisé. Pour cela il faut créer un fichier pour chaque dépôt dans le dossier **/etc/yum.repos.d**.

Pour connaître la liste des paquets à mettre à jour sans effectuer celle-ci, vous pouvez utiliser la commande : yum check-update

Pour rechercher un paquet on utilise la commande : yum list NomPaquet

Pour une recherche plus large : yum search MotCle

Pour installer un paquet : yum install NomPaquet

Pour supprimer un paquet : yum remove NomPaquet

Pour voir la liste des dépôts : yum repolist all

Le fichier de log de YUM est situé dans **/var/log/yum.log** et permet de suivre toutes les modifications.

For clean all packages not nedd anymore : yum clean all

Pour lister l’historique de yum: yum history

## Verifiy the integrity and availability of resources

**lsblk**

La commande **lsblk** permet d'obtenir la liste et les caractéristiques des disques et de leurs partitions.   
La commande ne nécessite pas les droits administrateurs pour être exécutée.

**Les commandes "mount" et "umount"**

**Description et intérêt de ces commandes**

Pour accéder à un périphérique sous Linux (disque dur externe, clé usb, CD-Rom) on doit utiliser la commande "mount". À l'origine, cette commande a été crée pour laisser aux administrateurs la plus grande liberté.  
Par exemple, il fallait que plusieurs utilisateurs puissent accéder à un même périphérique en même temps, aussi bien un disque dur qu'un CD-Rom. Mais il fallait empêcher ces utilisateurs de pouvoir "enlever" ce périphérique (imaginez que plusieurs utilisateurs accèdent aux données d'un CD-Rom, et qu'il y en ait un qui décide d'éjecter ce CD-Rom pendant que d'autres personnes sont en train de consulter des données sur celui-ci).  
  
Cela permettait également aux administrateurs de contrôler en détail les logiciels à installer car ils étaient les seuls à pouvoir accéder aux périphériques "amovibles" (notamment aux CD-Rom).  
  
Cela avait un intérêt particulier pour des ordinateurs branchés en réseaux. Mais pour l'utilisateur "lambda" utilisant un ordinateur personnel, ce type de manipulation était plutôt rebutant et même incompréhensible. Depuis la version "2.6" du noyau Linux, il existe un nouveau système de reconnaissance automatique (appelé "udev") qui permet de s'affranchir de ces manipulations.

**Exemples**

**Les Clés usb - disques durs externes - appareils photos numériques**

Si vous avez une clé usb, celle-ci est probablement formatée en "Fat32" (cela se caractérise par l'option **-t vfat**). Suivant le port usb sur laquelle vous l'avez connectée, son fichier "spécial de périphérique" est soit "/dev/sda" ou "/dev/sdb". Dans la plupart des cas, les clés usb sont formatées en primaire (donc "/dev/sda1" ou "/dev/sdb1"). Les manipulations pour "monter" une clé usb se résument donc aux étapes suivantes :  
  
Se loguer en root en tapant su puis le mot de passe root   
  
Créer un répertoire d'accès à la clé :   
  
mkdir /media/usb  
*(On pourrait choisir n'importe quel autre répertoire)*   
  
Ensuite, taper :   
mount -t vfat /dev/sda1 /media/usb   
*(Si cela ne fonctionne pas, remplacer "/dev/sda1" par "/dev/sdb1"*  
  
Certaines cartes "Secure Digital" d'appareil photo sont formatées en "dos". Cela correspond à l'option **-t umsdos**. Si les opérations ci-dessus ne fonctionne pas essayez la commande suivante :  
  
mount -t umsdos /dev/sda1 /media/usb

**Les lecteurs CD-Rom et les Graveurs**

Il faut savoir exactement sur quel nappe est connecté votre lecteur. Vous pouvez vous reporter à la [dénomination des périphériques](http://alex.nouvel.free.fr/tutoriels/peripheriques.php) pour en savoir plus. Dans la plupart des cas, ceux-ci sont connectés sur la deuxième nappe, soit en maître (/dev/hdc) soit en esclave (/dev/hdd). L'option de "montage" des CD-Rom est **-t iso9660**. Si vous avez un répertoire nommé "/media/cdrom", la commande est la suivante :   
  
mount -t iso9660 /dev/hdc /media/cdrom   
*(Si cela ne fonctionne pas, remplacer "/dev/hdc" par "/dev/hdd")*

**La commande "umount"**

Une fois que l'on a fini d'accéder au matériel, il faut "démonter" celui-ci. Par exemple, si on on "monté" le périphérique "/dev/sda1", il faudra le "démonter" avec la commande suivante :   
  
umount /dev/sda1

**FSCK**

C’est l'outil de contrôle d'intégrité et de réparation pour les systèmes de fichiers Linux.

La commande **fsck** (*File System Check* ou *File System Consistency Check*) permet le contrôle et la réparation des partitions.   
Ne pas confondre avec la défragmentation, fsck fait un check, une vérification de l'état du système de fichier, savoir si y'a pas d'erreur, de fichier/dossier partiellement écrit, des inodes agonisants et tout l'toutim, etc.  
Bref, il fait la révision, mais ne refait pas la carrosserie

Il faut impérativement que votre partition soit démontée, c'est-à-dire non accessible, ce qui est le cas avec un live CD/DVD, c'est pourquoi il est plus simple d'effectuer la vérification fsck depuis un [live CD/DVD](https://doc.ubuntu-fr.org/live_cd) ou un [live USB](https://doc.ubuntu-fr.org/live_usb).

Pour pouvoir utiliser fsck, vous devez au préalable connaître le [partitionnement](https://doc.ubuntu-fr.org/partitions) de votre disque (avec [gparted](https://doc.ubuntu-fr.org/gparted) par exemple).

Une fois choisie la partition à examiner, il faudra la démonter, par exemple si la partition est **dev/sda1** :

sudo umount /dev/sda1

On peut ensuite lancer la vérification :

sudo fsck /dev/sda1

## Verifiy the integrity and availability of key processes

With ps, top and htop

## Change kernel runtime parameters, persistent and non-persistent

In a Linux system the kernel controls almost every aspect of the operating system. Often, you will need to change parameters without taking the system down. Sometimes those changes are temporary, and sometimes they should survive a reboot.

**Un noyau**

Linux désigne au sens strict un *noyau de système d'exploitation*.

Le noyau est la couche de base d'un système d'exploitation. C'est le noyau qui gère la mémoire, l'accès aux périphériques (disque dur, carte son, carte réseau, etc...), la circulation des données sur le bus, les droits d'accès, les multiples processus qui correspondent aux multiples tâches que l'ordinateur doit exécuter en même temps, etc...

Par contre, le noyau ne gère pas le mail, l'affichage des pages Web, ou encore le traitement du texte. Ce sont des *programmes* ou *applications* qui s'en chargent. Ces programmes viennent se greffer sur le noyau, et ils doivent être adaptés à celui-ci.

Ce noyau de système d'exploitation a l'originalité d'être multi-utilisateurs et multi-tâches et de fonctionner sur de nombreuses plates-formes (Intel, Apple, Sparc, etc...). Il est conforme à la norme POSIX et est distribué sous Licence GPL. Il a la réputation d'être fiable, stable et sécurisé. Son appartenance au monde du libre garantit une correction rapide des erreurs qui pourraient être découvertes.

**Une distribution Linux**

Comme nous venons de le voir, Linux ne se suffit pas à lui-même. Avec un simple noyau, on ne peut rien faire ! Le noyau Linux vient donc à l'intérieur de *distributions*.

Une distribution Linux, c'est un ensemble cohérent de plusieurs choses :

* un noyau Linux,
* des programmes, en majorité libres (un navigateur Web, un lecteur de Mail, un serveur FTP, etc...),
* une méthode pour installer et désinstaller facilement ces programmes,
* un programme d'installation du système d'exploitation.

Le noyau Linux ne se suffit donc pas à lui même, mais on fait souvent un abus de langage en désignant par le terme *Linux* ce qui est en fait une *distribution Linux*.

**sysctl** est une commande utilisée pour modifier les paramètres du noyau en cours d'exécution. Les paramètres utilisables sont ceux listés dans le répertoire */proc/sys*. Procfs est nécessaire pour utiliser [**sysctl**](http://manpagesfr.free.fr/man/man8/sysctl.8.html)(8) sous Linux. Vous pouvez utiliser [**sysctl**](http://manpagesfr.free.fr/man/man8/sysctl.8.html)(8) aussi bien pour lire que pour écrire des paramètres.

Avec cette commande : cat /proc/sys/net/ipv4/ip\_forward

Tu verras que le service est off car sur 0.

Si je veux le mettre sur on je tape : sudo sysctl –w net.ipv4.ip\_forward=1

Those changes are going to take effect until the next reboot.

So if we want those changes to be persistent we need to go at cd /etc/sysctl.d

Tu verras les fichiers de conf à modifier pour que ces infos soient persistents après avoir fait un :

service procps start

## Use scripting to automate system maintenance tasks

Scripting is what separates the champions from the runners-up in the game of Linux! In this lesson, we'll go over the basics of shell scripting and create a very basic script to perform a series of commands in a set sequence. At the end of this lesson, you'll be able to craft sequential scripts that combine a series of commands.

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"

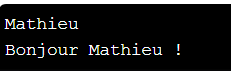
Vous êtes dans le dossier /home/qeyss/bin

## 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.

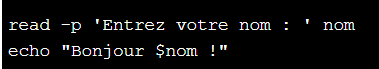


Lorsque vous lancez ce script, rien ne s'affiche, mais vous pouvez taper du texte (votre nom, par exemple) :



**-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.

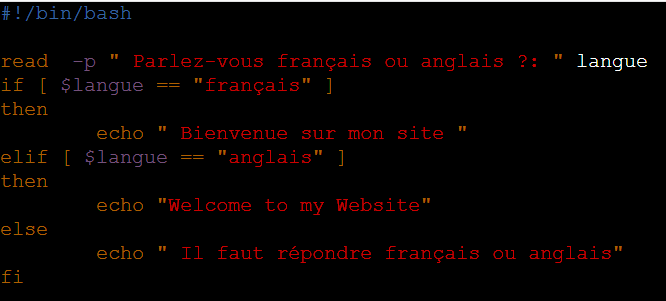


**-s : ne pas afficher le texte saisi**

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 :



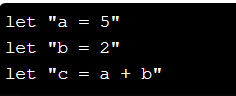
Si vous voulez insérer des retours à la ligne, il faudra activer le paramètre -e et utiliser le symbole \n



## 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 :

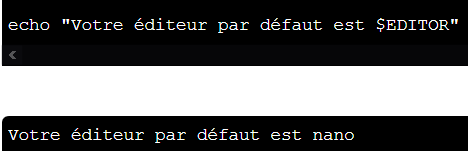
Parmi celles que je peux vous commenter et qui peuvent s'avérer utiles, on trouve :

* 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]}

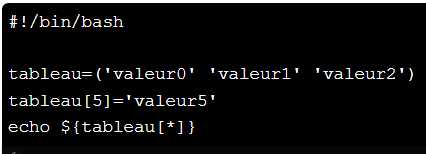
… ceci affichera le contenu de la case n° 2 (donc valeur2).

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 :



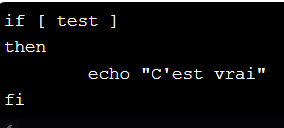
Vous pouvez afficher l'ensemble du contenu du tableau d'un seul coup en utilisant ${tableau[\*]}



## Scripting conditionals

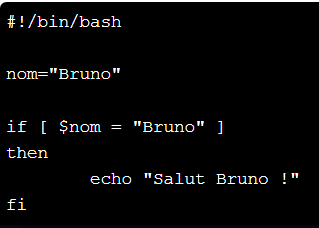
**if : la condition la plus simple**

Syntaxe :



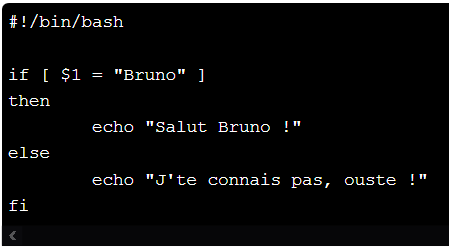
Le mot fi (if à l'envers !) à la fin indique que le if s'arrête là. Tout ce qui est entre le then et le fi sera exécuté uniquement si le test est vérifié.

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 ] !



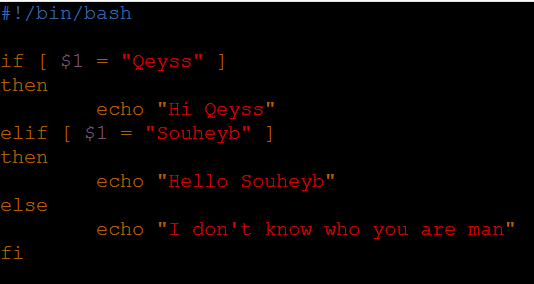
Ensuite il faut faire un chmod +x au fichier puis le lancer avec ./NomDuScript

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



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

**If et Elif**



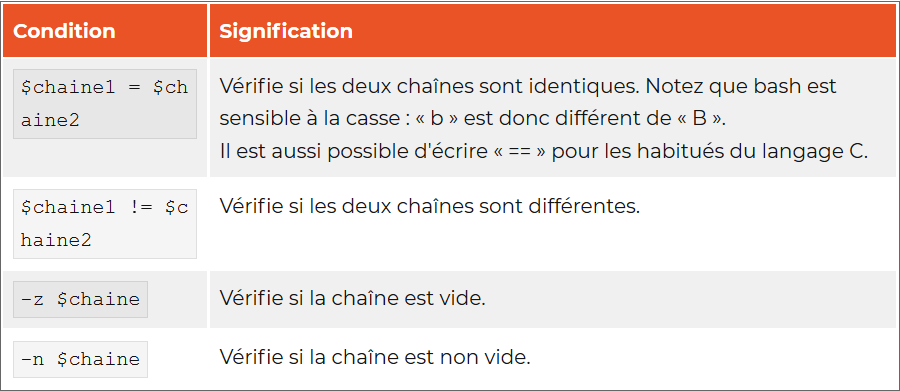
Si tu n’envoies aucun paramètre le script plante car il attend un paramètre.

**Les différents types de tests**

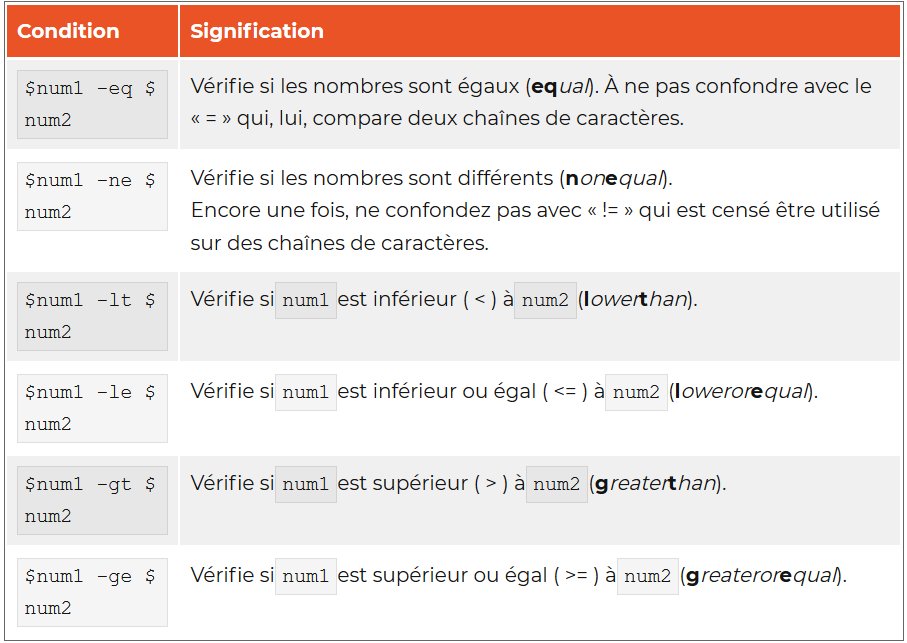
Il est possible d’effectuer trois types de tests différents en bash :

* 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**

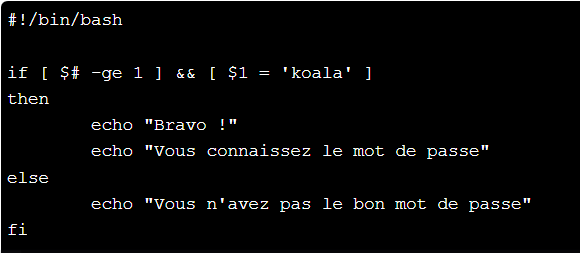
Dans un if, il est possible de faire plusieurs tests à la fois. En général, on vérifie :

* 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 ».

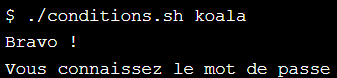
Il faut encadrer chaque condition par des crochets. Prenons un exemple :



Le test vérifie deux choses :

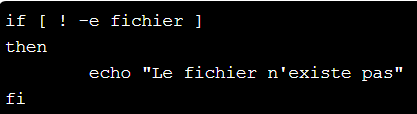
* qu'il y a au moins un paramètre (« si$#est supérieur ou égal à 1 ») ;
* que le premier paramètre est bienkoala(« si$1est égal àkoala »).

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**

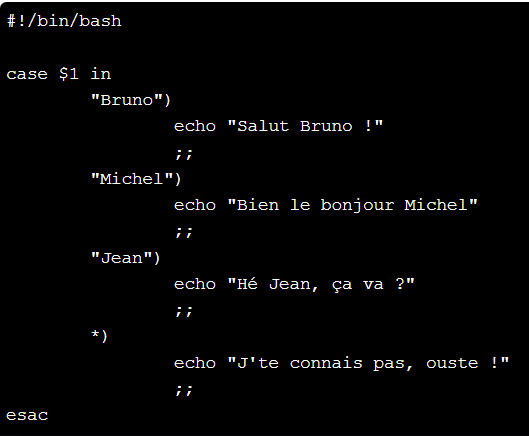
Il est possible d'inverser un test en utilisant la négation. En bash, celle-ci est exprimée par le point d'exclamation « ! ».



**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")

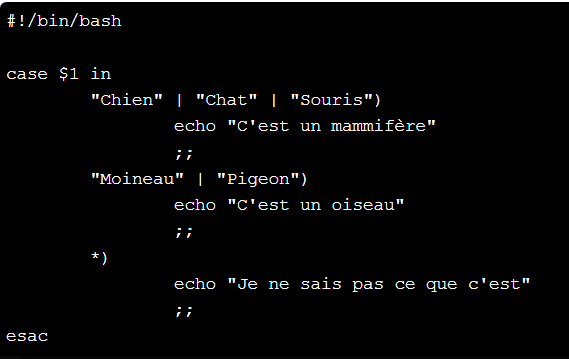
Là, on teste une valeur. Cela signifie « Si$1est égal à Bruno ». Notez que l'on peut aussi utiliser une étoile comme joker : « B\* » acceptera tous les mots qui commencent par un B majuscule.

\*)

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 :

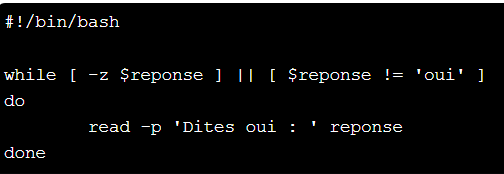
while [ test ]

do

        echo 'Action en boucle'

done

Exemple:



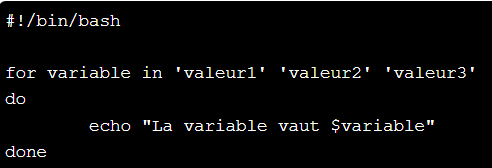
« Tant que la réponse est vide ou que la réponse est différente deoui ».  
Nous sommes obligés de vérifier d'abord si la variable n'est pas vide, car si elle l'est, le second test plante.

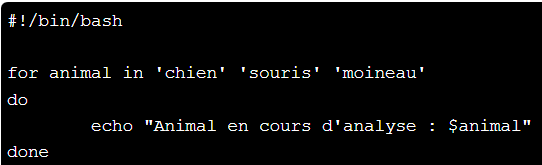
**for : boucler sur une liste de valeurs**

**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 :





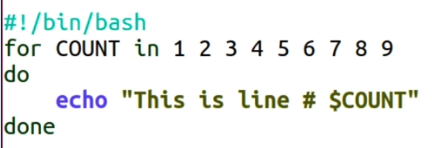
Va afficher :

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

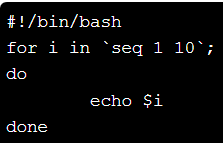
This is line # 2

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

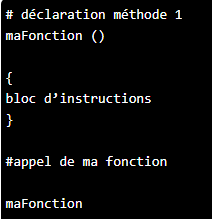
**Qu’est-ce qu’une fonction ?**

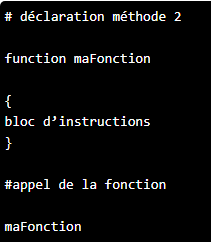
Une fonction, c’est un ensemble d’instructions, permettant d'effectuer plusieurs tâches avec des **paramètres d'entrée différents**.

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) ...

## Manage the startup process and services (In Services Configuration)

Starting applications on Linux is fairly easy, depending on which service manager your system uses. In this lesson, we'll go over the basics of both! Upstart and systemctl manage startup processes and services on Linux, and we'll demonstrate how to use each of them.

# User and Groups Management

## Create, Delete, and Modify Local User Accounts

Maintaining user accounts is a staple activity in the life of a Linux Administrator

Le fichier contenant les users est **group** situé dans etc, on peut le visualiser avec un cat /etc/group

When you create a brand new user you are also creating a single user group of the same name.

The old method was to use the command **useradd** like this : useradd NameUser

And then you need to give him a password with passwd NameUser

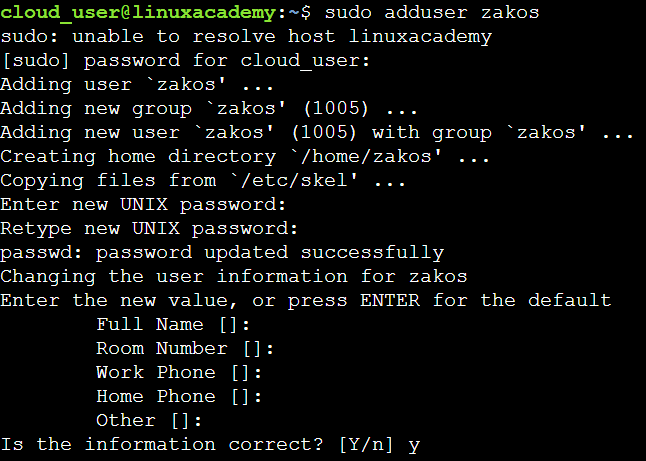
But in this way your new user won’t have an home directory so you need to create one like this : mkdir NameUser then give him all permissions and this new group with chown

Mais ton user n’aura aucun fichier source dans son dossier like bashcr etc… Il faut les ajouter, ils sont dans /etc/skel

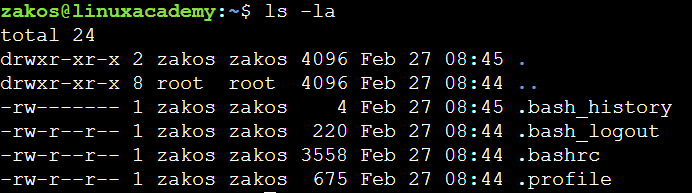
The new method

We use the command **adduser**

Et là il te demandera diverses infos sur le new user :



Et là tu auras automatiquement tous les fichiers bash etc…



Remove a user

On utilise la commande **userdel** + NameUser ( sous Centos) et **deluser** sous Debian

En faisant ainsi ça ne va pas supprimer le dossier du user dans home, il faudra donc attribuer ce dossier de manière récursive à un autre user puis le supprimer ou passer en root et le supprimer ainsi :

rmdir NomDossier 🡪 Si le dossier est vide

rm –r NomDossier  🡪 Tu devras confirmer chaque fichier à supprimer

rm –rf NomDossier  🡪 Supprime tout directement

## Create, Delete, and Modify Local Groups and Group Memberships

wheel:x:10:cloud\_user

wheel : nom du groupe

10 : guid

cloud\_user : nom de l’utilisateur

**Ajouter un user à un groupe**

Pour ajouter un user à un groupe tu ouvres le fichier /etc/group et ajoute le nom de l’utilisateur à la fin de la ligne souhaitée ex :

wheel:x:10:qeyss

**Savoir à quels groupes j’appartiens**

On tape la commande **groups**

By default groups don’t have passwords already set for them

**Changer le password d’un groupe**

sudo **gpasswd** NomDuGroupe

**Créer un groupe**

On utilise la commande **groupadd et addgroup (Pour Debian)**

**Modifier un utilisateur**

On utilise la commande **usermod**

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

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

**Supprimer un groupe**

Avec la commande **groupdel** et **delgroup (Debian)**

## Manage System-Wide Environment Profiles

Les variables d'environnement constituent un moyen d'influencer le comportement des logiciels sur votre système. Par exemple, la variable d'environnement « LANG » détermine la langue que les logiciels utilisent pour communiquer avec l'utilisateur.

Les variables sont constituées de **noms** auxquels on assigne des **valeurs**.

En tapant **env** tu verras toutes les variables d’environnement.

Créer une variable d’env

Pour créer la variable d'environnement « EDITOR » et lui assigner la valeur « nano » :

export EDITOR=nano

2ème méthode :

EDITOR=nano

export EDITOR

**Affecter une valeur à une variable d'environnement existante**

Par exemple pour assigner la valeur « fr\_FR.UTF-8 » à la variable « LANG », nous utilisons la commande suivante : LANG=fr\_FR.UTF-8

**Effacer des variables d'environnement**

Dans la plupart des cas, affecter une valeur vide à une variable d'environnement suffit à annuler son effet, comme dans l'exemple ci-dessous. Cependant certaines variables comme « POSIXLY\_CORRECT » ont une influence sur les programmes du fait même de leur existence, et ce, même si leur valeur est vide.

export LC\_ALL=

La commande **unset** peut être utilisée pour supprimer complètement une variable d'environnement :

unset LC\_ALL

**Le fichier bashcr**

.bashrc is a [shell script](http://en.wikipedia.org/wiki/Shell_script) that Bash runs whenever it is started interactively. It initializes an interactive shell session. You can put any command in that file that you could type at the command prompt.

You put commands here to set up the shell for use in your particular environment, or to customize things to your preferences. A common thing to put in .bashrc are [aliases](http://en.wikipedia.org/wiki/Alias_(command)) that you want to always be available.

The purpose of a .bashrc file is to provide a place where you can set up variables, functions and aliases, define your (PS1) prompt and define other settings that you want to use every time you open a new terminal window.

Le fichier .bashrc vous permet entre autres choses de personnaliser l'invite de commandes. Vous savez, ce petit message qui s'affiche devant votre curseur dans la console :

mateo21@mateo21-desktop:~$

**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.

Ce fichier propose un peu moins d'exemples commentés que celui présent dans votre home. Vous pouvez y copier vos alias et la ligne définissant l'invite de commandes (commençant parPS1).

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.

**Variables d'environnement au niveau session utilisateur**

Pour définir une variable d'environnement de manière à ce qu'elle affecte l'ensemble de la session d'un utilisateur, il suffit de placer une commande la définissant dans l'un des fichiers cachés de script présent dans le répertoire personnel de l'utilisateur. Voici les fichiers les plus courants qui peuvent être utilisés :

* **~/.profile** - C'est probablement le meilleur endroit pour placer une définition de variable d'environnement. En effet, il est exécuté automatiquement par le gestionnaire de connexion lors du démarrage d'une session graphique, mais aussi lors du démarrage d'une session en mode console texte.
* **~/.bash\_profile** ou **~/.bash\_login** - Si l'un de ces fichiers existe, il sera exécuté par **Bash** préférentiellement à **~/.profile** lors d'une connexion sur une console. (Bash utilisera ~/.bash\_profile de préférence à ~/.bash\_login). Cependant, ces fichiers n'auront par défaut aucune influence sur une session en mode graphique.
* **~/.bashrc** - Du fait de la manière dont Ubuntu configure par défaut les divers fichiers de scripts, c'est sans doute l'endroit le plus facile pour définir des variables. La configuration par défaut garantit a peu près que ce fichier sera exécuté à chaque invocation de \*bash\* ainsi que lors de la connexion à l'environnement graphique. Cependant du point de vue des performances, ce n'est pas l'idéal car les variables seront inutilement redéfinies à chaque fois. (NdT : à chaque fois que vous ouvrez un terminal par exemple?)

Haut du formulaire

Bas du formulaire

**Variables d'environnement au niveau système**

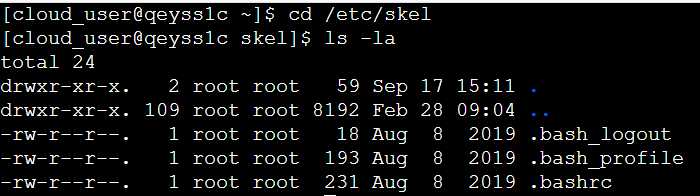
Les variables d'environnement qui affectent l'ensemble du système (plutôt qu'une session utilisateur particulière) peuvent être définies dans l'un des nombreux scripts au niveau système, qui s'exécutent lors du chargement du système ou de l'environnement graphique. Ces définitions peuvent être placées dans plusieurs fichiers sur Ubuntu :

* **/etc/profile** - Ce fichier est exécuté quelle que soit la méthode de connexion utilisée : une console, une connexion distante ssh, ou une connexion en mode graphique. C'est probablement l'emplacement qui vous sera conseillé par les vieux routiers d'UNIX pour gérer vos variables d'environnement. Cependant sur Ubuntu ce script fait quelques vérifications puis invoque **/etc/bash.bashrc**.
* **/etc/bash.bashrc** - C'est la version au niveau système du fichier utilisateur **~/.bashrc**. Par défaut Ubuntu est configuré pour exécuter ce fichier quelle que soit la méthode de connexion, sur une console ou en environnement graphique.
* **/etc/environment** - Ce fichier est spécialement conçu pour recevoir les définitions de variables d'environnement au niveau système. Ce n'est pas un fichier de script, mais plutôt un fichier de déclarations de variables, ligne par ligne. En particulier, il contient les définitions des variables de langue et de la variable **PATH** au niveau système.

**Remarque :** sur des systèmes destinés à un usage personnel, il est sans doute préférable de définir les variables au niveau utilisateur tel que décrit plus haut, plutôt qu'au niveau système. En effet les fichiers utilisateurs peuvent être modifiés sans nécessiter les privilèges d'administration contrairement à ceux-ci.

## Manage Template User Environment

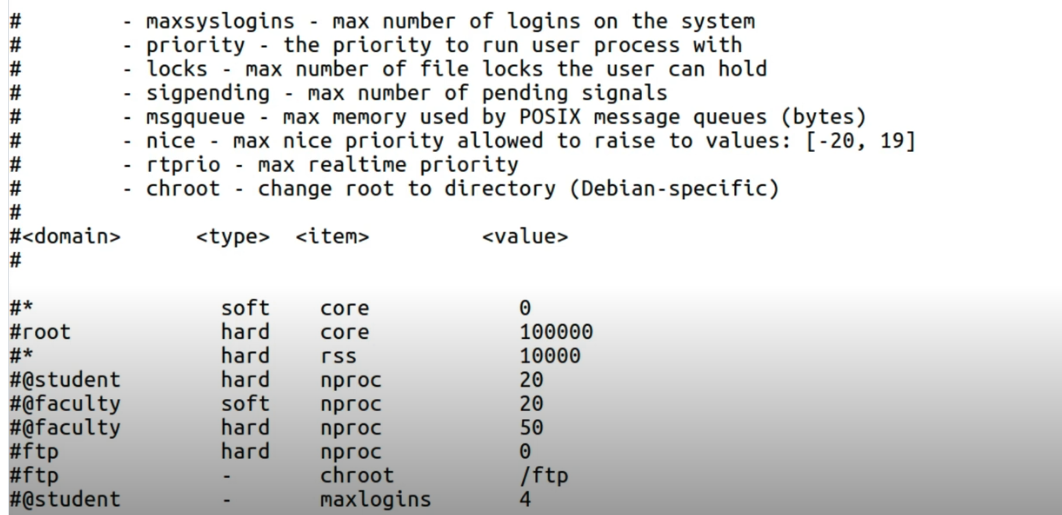
Quand un nouvel user est créé, les fichiers suivants dans etc/skel seront copies dans son home directory



Tu peux ajouter tout fichier ou dossier que tu veux à cet endroit ainsi ils seront de base créés pour tout nouveau user.

## Configure User Resource Limits

In order to preserve the availability of your server, sometimes you have to put limits on users. In this file you can limit what individual users have access to into your system. In order to limit user resource usage you have to edit the **/etc/security/limits.conf** file.



**[domain] can be:**

* an user name
* a group name, with @group syntax
* the wildcard **\***, for default entry
* the wildcard **%**, can be also used with %group syntax, for maxlogin limit

**[type] can have the two values:**

* “**soft**” for enforcing the soft limits
* “**hard**” for enforcing hard limits

**[item] can be one of the following:**

* **core** – limits the core file size (KB)
* **data** – max data size (KB)
* **fsize** – maximum filesize (KB)
* **memlock** – max locked-in-memory address space (KB)
* **nofile** – max number of open files
* **rss** – max resident set size (KB)
* **stack** – max stack size (KB)
* **cpu** – max CPU time (MIN)
* **nproc** – max number of processes
* **as** – address space limit (KB)
* **maxlogins** – max number of logins for this user
* **maxsyslogins** – max number of logins on the system
* **priority** – the priority to run user process with
* **locks** – max number of file locks the user can hold
* **sigpending** – max number of pending signals
* **msgqueue** – max memory used by POSIX message queues (bytes)
* **nice** – max nice priority allowed to raise to values: [-20, 19]
* **rtprio** – max realtime priority

## Manage User Privileges

I want some users to be able to run a command, but not others. What's an admin to do?! In this lesson, we'll take a look at how file permissions, which limit which commands a user may execute depending on group memberships, as well as how to limit whether and from where a user can log in to the system. After this lesson, you will be able to restrict command execution to particular groups.

**access.conf** - the login access control table file

**Description**

The **/etc/security/access.conf** file specifies (*user/group*, *host*), (*user/group*, *network/netmask*) or (*user/group*, *tty*) combinations for which a login will be either accepted or refused.

When someone logs in, the file access.conf is scanned for the first entry that matches the (*user/group*, *host*) or (*user/group*, *network/netmask*) combination, or, in case of non-networked logins, the first entry that matches the (*user/group*, *tty*) combination. The permissions field of that table entry determines whether the login will be accepted or refused.

Each line of the login access control table has three fields separated by a ":" character (colon):

*permission*:*users/groups*:*origins*

The first field, the *permission* field, can be either a "*+*" character (plus) for access granted or a "*-*" character (minus) for access denied.

The second field, the *users*/*group* field, should be a list of one or more login names, group names, or *ALL* (which always matches). To differentiate user entries from group entries, group entries should be written with brackets, e.g. *(group)*.

The third field, the *origins* field, should be a list of one or more tty names (for non-networked logins), host names, domain names (begin with "."), host addresses, internet network numbers (end with "."), internet network addresses with network mask (where network mask can be a decimal number or an internet address also), *ALL* (which always matches) or *LOCAL*. *LOCAL* keyword matches if and only if the *PAM\_RHOST* is not set and <origin> field is thus set from *PAM\_TTY* or *PAM\_SERVICE*". If supported by the system you can use *@netgroupname* in host or user patterns. The *@@netgroupname* syntax is supported in the user pattern only and it makes the local system hostname to be passed to the netgroup match call in addition to the user name. This might not work correctly on some libc implementations causing the match to always fail.

## Configure PAM

Pluggable Authentication Modules in Linux allows you to configure and reconfigure how authentication takes place in PAM-aware applications. In this lesson, we'll discuss PAM and what it is for, and how systems administrators can use it to configure PAM-aware applications to use a particular authentication scheme. After this lesson, you will be able to configure PAM on Linux systems.

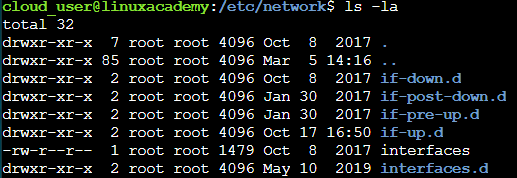
# Networking

## Configure networking and hostname resolution statically or dynamically

Linux systems live on networks with hardware devices, Windows systems, Apple systems, and more. As such, they need to support both static IP addresses and host name resolution and standard DHCP (Dynamic Host Configuration Protocol). In this lesson, we'll look at the configuration files necessary to configure networking and hostname resolution statically or dynamically on both Debian/Ubuntu systems and CentOS/RedHat systems. After this lesson, you'll be able to statically configure your Linux system or allow it to receive configuration information from a DHCP server.

**Configuration with an older Ubuntu**

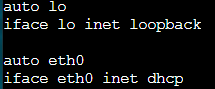
Les fichiers liés au network sont dans **cat /etc/network**



**The file interfaces**

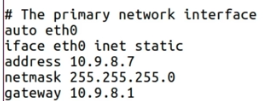
This files describes the network interfaces available on your system.

Dans le dossier interfaces.d tu verras un fichier eth0 avec ta configuration



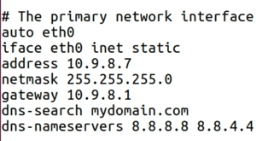
Eth0 est ta carte réseau, si tu veux la mettre en statique faut supprimer dhcp et mettre **static** à la place.

Et ajouter une ligne avec une adresse ip, puis le netmask et le gateway, ex :



Ensuite il faut faire un sudo ifup eth0 for restart that interface and change it’s IP address

Je peux aussi changer le dns en les ajoutant à la fin :



**Configuration with a new Ubuntu**

Dans le dossier interfaces.d tu ne verras pas forcément le fichier eth0 mais un autre comme :

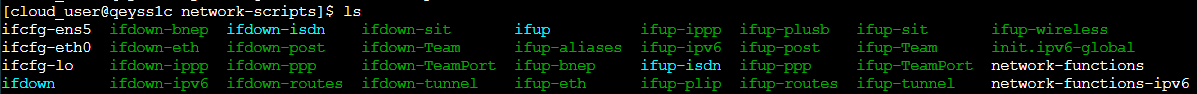


Ensuite on ouvrant le fichier ça sera identique à l’ancien et on pourra y faire les mêmes changements

**Configuration sur CentOS**

Il faut aller dans **/etc/sysconfig/network-scripts**

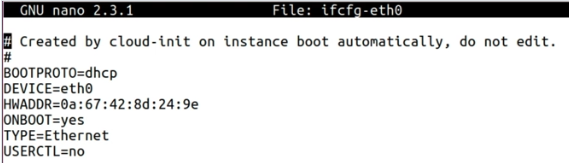
This directory has a lot of files in it.



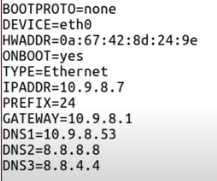
Ces premiers fichiers sont les plus importants



Dans ifcfg-eth0 on verra :



Je peux faire des changements et fixer l’IP, mettre un netmask, gateway, DNS, etc..



Ensuite il faudra taper : **systemctl restart network**

## Configure Network Services to Start Automatically at Boot

Linux systems often serve as file servers, web servers, FTP servers, network times servers, and other network entities whose services are relied upon by many other machines, applications, and systems on the network. In this lesson, we'll review configuring services to start automatically at boot on both SystemD systems and SysV systems, with a focus on networking services. After this lesson, you will be able to configure services to start at boot on both SystemD systems and SysV systems.

Implement Packet Filtering

Start, Stop, and Check the Status of Network Services

1

# Learning by doing

Archiving and Compressing Files and Directories

Managing File Attributes and Permissions

## Finding User Files and reassigning Permissions

Your development has recently downsized several of the contractors that were working on the custom application being prepared for sale. One particular developer named Jason Todd (whose username was *jtodd*) was working on a key piece of the application and served as the build coordinator of the team.

Since his departure, none of the other developers have been able to access the directory where all the files for the project are stored, and they are unsure if there are other files or directories owned by Jason that they need. You have been asked to find *all* of his files and directories on the development server and reassign their ownership to the user 'cloud\_user'.

Using the appropriate command(s) and the username and credentials for the system provided, find all files and directories on the system currently owned by *jtodd* and reassign their ownership to the user called *cloud\_user* and turn the system back over to your development team so that they can continue their work.

1° Find all files owned by jtodd

sudo find / -user jtodd

2° The owner of the */opt/myapp* directory and all files and subdirectories within it should be *cloud\_user*

sudo chown cloud\_user:cloud\_user -R /opt/myapp

3° The owner of /home/jtodd should be cloud\_user

sudo chown cloud\_user:cloud\_user /home/jtodd

4° S’assurer que plus aucun file appartient à jtodd

sudo find / -user jtodd

## Determining Which Distribution Is Running on a Host

On tape : hostnamectl ou cat /etc/\*release\*

## Create a new user with the old method

1° useradd Souheyb

2° passwd Souheyb

3° Dans/home faire un mkdir Souheyb

4° chown Souheyb:Souheyb Souheyb/

## Create a new user with the new method

A

Extras:

### The Sudoers File

This file is the seedy underbelly of sudo.  It controls who can use the sudo command to gain elevated privileges.  It is usually located at /etc/sudoers.  The best and safest way to edit this file is by using the visudo command.

Si un user n’est pas dans ce fichier, il ne pourra pas executer la commande sudo.

To do :

Se connecter à distance en ssh à ton serveur Linux raspberry

dd