

The LPIC-1 logo is a large, light gray circular emblem. Inside the circle is a stylized 'L' and a dot, resembling a smiley face. A registered trademark symbol (®) is located at the top right of the circle.

# Administration Linux niveau 1

# LPIC-1

Trainer:Younes Kamal Elidrissi



## About me

# Younes Kamal Elidrissi : Trainer Consultant

[kamal\\_elidrissi@hotmail.com](mailto:kamal_elidrissi@hotmail.com)

[kamal\\_elidrissi@befinfo.com](mailto:kamal_elidrissi@befinfo.com)

**RHCE-CSM:** Redhat Certificate of Expertise in Clustering and Storage Management

**RHCSE:** Redhat Certified System Engineer

**RHCSA :** Redhat Certified System Administrator

**VITAI :** VMware IT Academy Instructor

**VCP :** VMware Certified Professional

**VCP-Cloud :** VMware Certified Professional Cloud

**SCSA:** *Sun certified system Administrator solaris10*

**SCNA:** *Sun certified Network Administrator solaris10*

**LPI:** *linux Professional institut administrateur niveau 1*

**NCLA:** *Novel Certified Linux Administrator Suse Enterprise 11*



# Linux LPI niveau 1 (lpi-101)

- Module 00 : Historique
- Module 01 : Introduction au système linux
- Module 02 : Installation De Linux (Redhat, Centos,...)
- Module 03 : Commandes de base partie1
- Module 04 : Commandes de base partie2
- Module 05 : Les scripts Shell
- Module 06 : Les utilisateurs et les droits
- Module 07 : Gestion du système de fichier
- Module 08 : Gestion et installation des packages

# Linux LPI niveau 1 (lpi-102)

- Module 09 : **Noyau linux**
- Module 10 : **Démarrage, initialisation, arrêt système**
- Module 11 : **Taches Administratives ( crontab, cpio,..)**
- Module 12 : **L'impression**
- Module 13: **Configuration Réseaux ( tcp/ip,.....)**
- Module 14: **Services Réseaux ( web, Samba, NFS,...)**
- Module 15 : **Les utilitaires du client linux (ftp,ssh..)**
- Module 16 : **Élément de sécurité (TCPwrappers, accounts,..)**
- Module 17 : **L'environnement Graphique (X11, GDM..)**



LPI-101

LPIC-1

Trainer: Younes Kamal Elidrissi



# Historique suite

- 1969 : Ken Thompson et Dennis Ritchie écrivent une première version du noyau d'un système d'exploitation pour les laboratoires BELL.
- Ils inventent en 1973 un langage de programmation nommé le langage C. Ils ré-écrivent le noyau d'UNIX en C, L'avantage est celui de la portabilité de l'OS.
- L'entreprise ATT qui fournit les premières versions commerciales de cet OS. Diverses entreprises s'intéressent à ce marché, et plusieurs versions apparaissent (Sun et son SunOS, qui deviendra Solaris, IBM et son AIX, HP et son HPUNIX...).
- L'université californienne de Berkeley travaille également sur Unix, et lui apporte des atouts en réseau .
- Les Unix d'aujourd'hui sont les héritiers des versions BSD (berkeley) ou ATT (System V), soit un peu des deux. Sun est de nos jours un acteur majeur d'Unix.

# historique

- le système GNU/Linux a vu le jour en 1991 par un étudiant finlandais, Linus Torvalds.
- Linux a su garder, au fil de ses évolutions, l'héritage des tous premiers systèmes Unix.
- Parallèlement à cela, une philosophie nouvelle est apparue concernant le partage des connaissances, protégée par une licence qui garantirait la transparence des fichiers sources et la possibilité de les modifier.

# Naissance de linux

1991 : Linus Torvalds (Finlandais) développe un noyau s'inspirant d'unix : linux. Il le met très vite sous licence GPL, rejoint par de nombreux développeurs.



Succès : qualité technique du noyau + nombreuses distributions qui facilitent l'installation du système et des programmes





# **Module 01:**

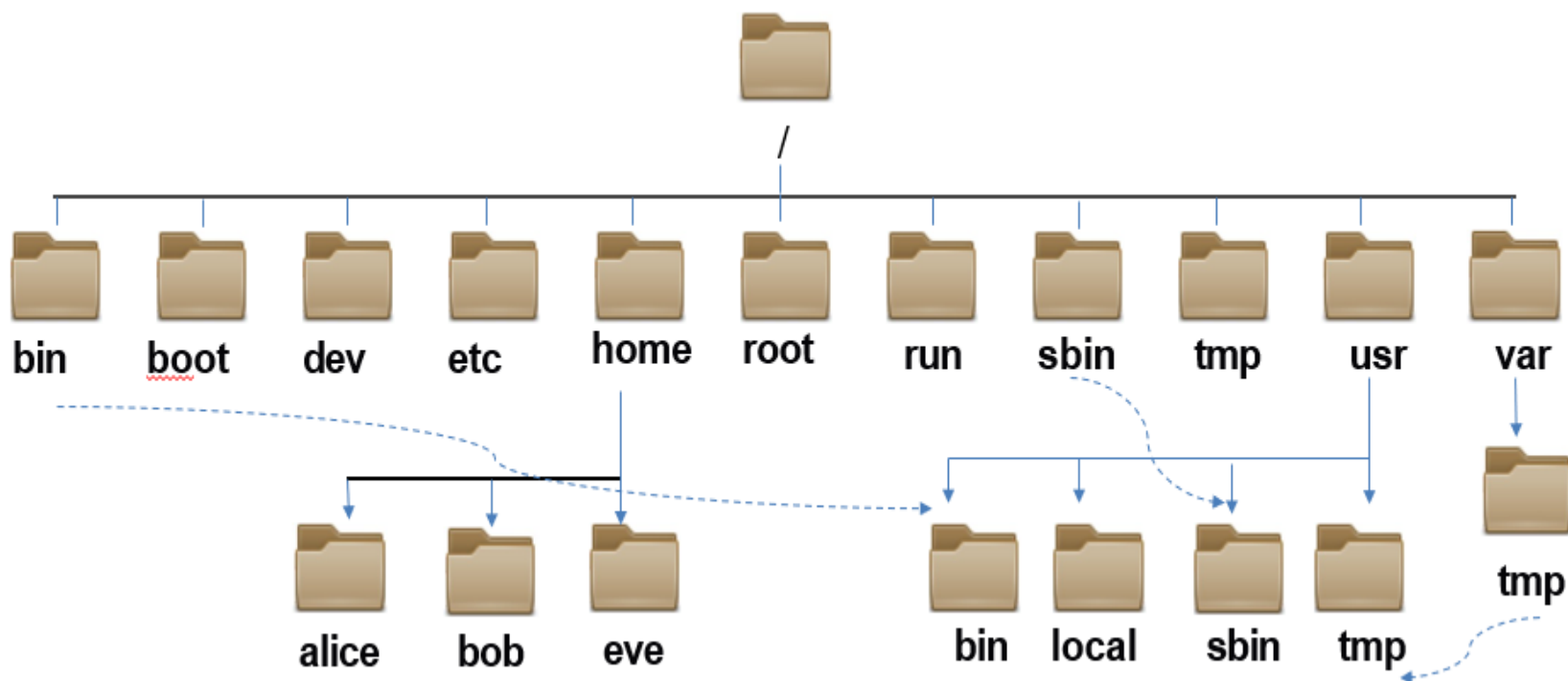
## **Introduction au système linux**

# Module 01 : Introduction au système linux

# Noyau linux

- Il s'agit de la partie fondamentale d'un système d'exploitation. Le noyau permet de simplifier et sécuriser l'utilisation des différents composants et périphériques de l'ordinateur.
- Il détermine également quel programme doit s'exécuter et pendant combien de temps grâce à une méthode appelée l'ordonnancement.
- Résident en mémoire, se charge au démarrage (boot).
- Le noyau est « minimal », les fonctionnalités plus évoluées sont exécutées en mode « utilisateur »

# Le système de fichiers



# Le système de fichiers

/	Racine du système, contient les répertoires principaux
/bin	Commandes essentielles communes à tous les utilisateurs
/boot	Fichiers de démarrage du système, contient le noyau
/dev	Points d'entrée des périphériques
/etc	Fichiers de configuration
/home	Contient les répertoires personnels des différents utilisateurs
/root	Répertoire personnel de l'administrateur
/usr	Hiérarchie secondaire, applications, bibliothèques partagées
/proc	Système de fichier virtuel, informations en temps réel

# Le Shell : intérêt ?

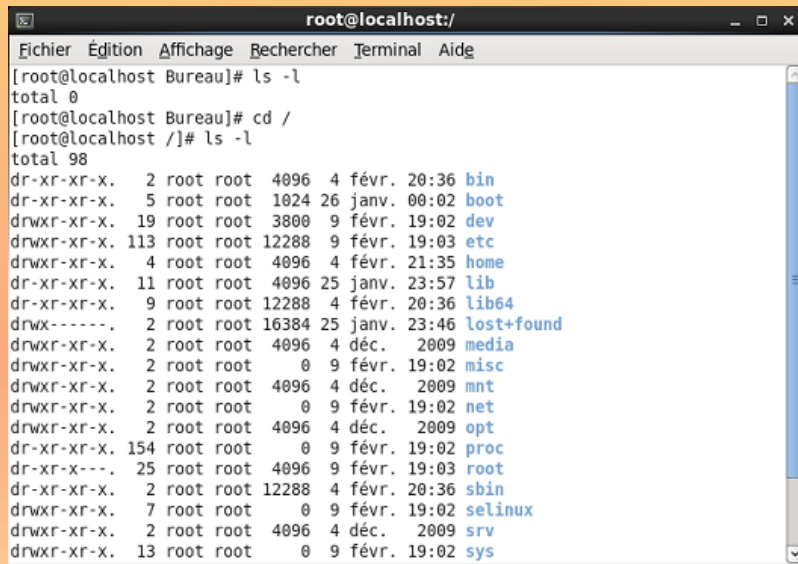
- Interface utilisateur non graphique
    - terminaux texte
    - accès distant
  - Interpréteur de scripts
    - traitement "par lots"
    - automatisation
  - Lancement de tâches multiples
    - tâche combinées (pipes)
    - job control
  - Le choix est vaste
    - sh, bash, ksh, csh, tcsh, ash, zsh, fish,
- On peut exécuter plusieurs shells en parallèle

# Le Shell

Nom	Description
bash	(Bourne Again Shell) offre l'édition de la ligne de commande et le rappel des commandes précédentes
csh	(C Shell) développé à Berkeley, compatible avec le shell Bourne. Pas d'édition de la ligne de commande ni d'historique des commandes
ksh	(Korn Shell) offre l'édition de la ligne de commande (touches compatibles Emacs)
sh	le shell original, pas d'édition de la ligne de commande.
tcsh	version améliorée du csh, avec un support de l'édition de la ligne de commande avec correction des commandes tapées
zsh	shell similaire au Korn shell, avec plus de dynamisme lors des affichages et gère la non redondance des commandes.

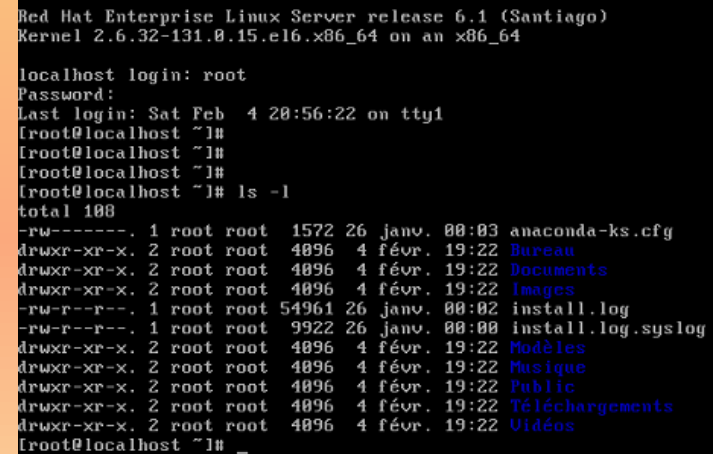
# Terminaux

Le terminal est un programme qui ouvre une console dans une interface graphique, il permet de lancer des **commandes**.



```
root@localhost:/  
Fichier Édition Affichage Rechercher Terminal Aide  
[root@localhost Bureau]# ls -l  
total 0  
[root@localhost Bureau]# cd /  
[root@localhost /]# ls -l  
total 98  
dr-xr-xr-x.  2 root root  4096  4 févr. 20:36 bin  
dr-xr-xr-x.  5 root root 1024 26 janv. 00:02 boot  
drwxr-xr-x. 19 root root 3800  9 févr. 19:02 dev  
drwxr-xr-x. 113 root root 12288  9 févr. 19:03 etc  
drwxr-xr-x.  4 root root  4096  4 févr. 21:35 home  
dr-xr-xr-x. 11 root root  4096 25 janv. 23:57 lib  
dr-xr-xr-x.  9 root root 12288  4 févr. 20:36 lib64  
drwx-----  2 root root 16384 25 janv. 23:46 lost+found  
drwxr-xr-x.  2 root root  4096  4 déc.  2009 media  
drwxr-xr-x.  2 root root    0  9 févr. 19:02 misc  
drwxr-xr-x.  2 root root  4096  4 déc.  2009 mnt  
drwxr-xr-x.  2 root root    0  9 févr. 19:02 net  
drwxr-xr-x.  2 root root  4096  4 déc.  2009 opt  
dr-xr-xr-x. 154 root root    0  9 févr. 19:02 proc  
dr-xr-x---. 25 root root  4096  9 févr. 19:03 root  
dr-xr-xr-x.  2 root root 12288  4 févr. 20:36 sbin  
drwxr-xr-x.  7 root root    0  9 févr. 19:02 selinux  
drwxr-xr-x.  2 root root  4096  4 déc.  2009 srv  
drwxr-xr-x. 13 root root    0  9 févr. 19:02 sys
```

Gnome-terminal



```
Red Hat Enterprise Linux Server release 6.1 (Santiago)  
Kernel 2.6.32-131.0.15.el6.x86_64 on an x86_64  
  
localhost login: root  
Password:  
Last login: Sat Feb  4 20:56:22 on tty1  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]#  
[root@localhost ~]# ls -l  
total 108  
-rw-----.  1 root root  1572 26 janv. 00:03 anaconda-ks.cfg  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Bureau  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Documents  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Images  
-rw-r--r--.  1 root root 54961 26 janv. 00:02 install.log  
-rw-r--r--.  1 root root  9922 26 janv. 00:00 install.log.syslog  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Modèles  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Musique  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Public  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Téléchargements  
drwxr-xr-x.  2 root root  4096  4 févr. 19:22 Vidéos  
[root@localhost ~]# _
```

console



# Le Login user et root

**/bin/bash** est le shell par défaut sur redhat il est lancé quand vous vous loggez depuis un terminal virtuel (**Ctrl-Alt-F1 à Ctrl-Alt-F6**).  
ou lorsque vous lancez une session **gnome-terminal** :

- Le compte utilisateur simple *classique*
- Le compte *root* (Administrateur)

Un symbole de l'invite de commande vous permet de déterminer le type de compte que vous utilisez.

Accès simple utilisateur :

- **user@localhost \$**

Accès root :

- **root@localhost #**

**Module 02:**

**Installation Linux Redhat  
Enterprise**

## Votre matériel est-il compatible ?

- La compatibilité matérielle constitue un point essentiel si vous possédez un système plus ancien ou élaboré par vos soins.
- **RedHat Enterprise Linux** est théoriquement compatible avec la plupart des configurations matérielles assemblées en usine au cours des deux dernières années.
- La liste la plus récente de compatibilité du matériel est fournie à l'adresse :

<http://hardware.redhat.com/hcl/>

## Avez-vous suffisamment d'espace disque ?

-À moins que vous n'ayez une raison particulière pour agir différemment, nous vous conseillons de créer les partitions suivantes pour les systèmes **x86, AMD64 et Intel® 64** :

- Une partition swap (d'au moins 256 Mo)
- Une partition **/boot/ (100 Mo)**
- Une partition **root (3.0 Go - 5.0 Go)** — où se trouve "/" (le répertoire racine).

Une **partition de 3.0 Go** vous permet d'effectuer une installation minimale, alors qu'une **partition root de 5.0 Go** vous permet d'effectuer une installation complète.

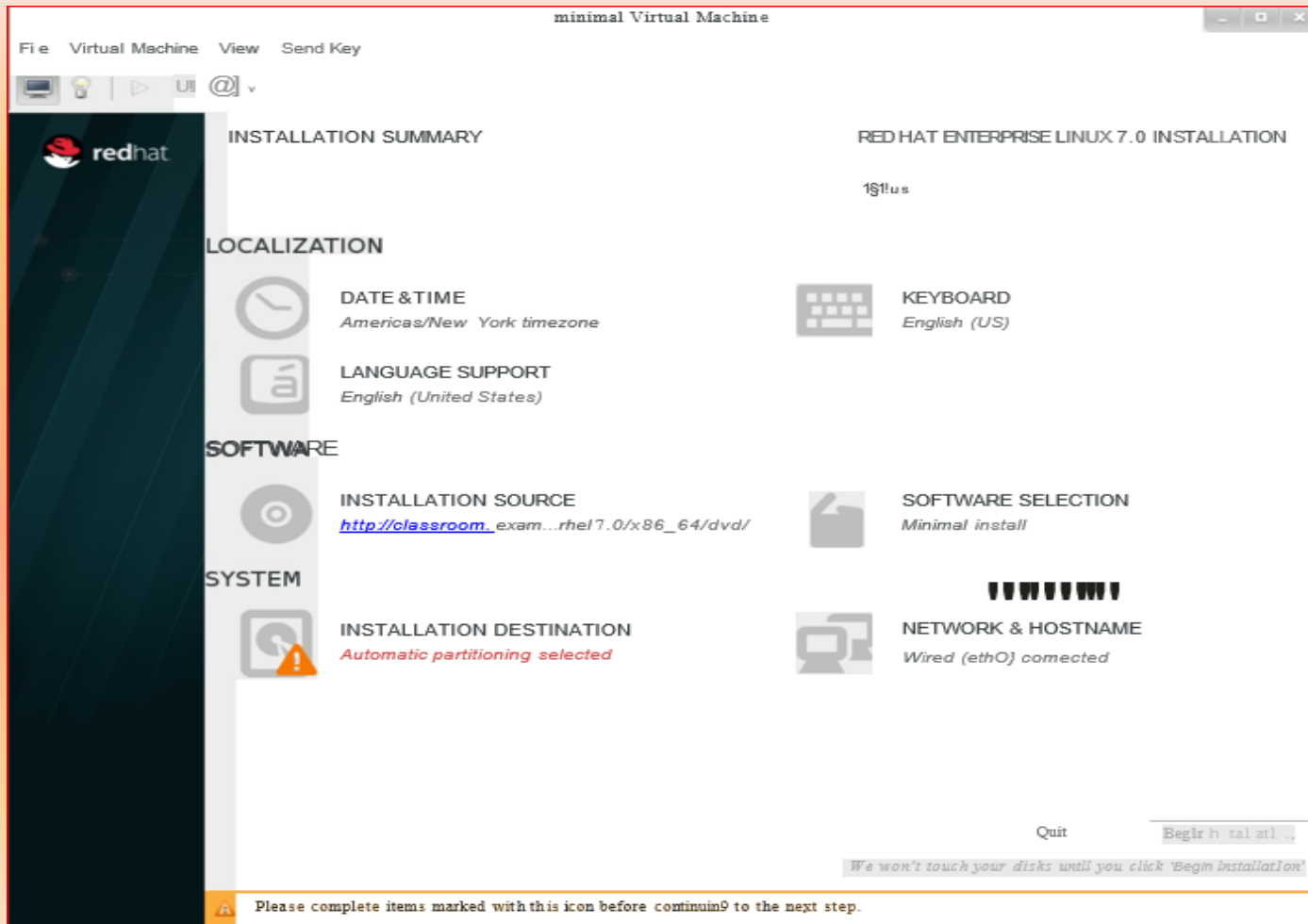
# Installation De Base

- Une installation de base ne nécessite que deux partitions, la partition racine / et une partition de **swap** (mémoire virtuelle).
- - **Le kernel Linux 2.6** est notamment plus performant lorsqu'un swap est disponible:
- - il peut ainsi y déplacer des données non fréquemment utilisées et donc gagner en performance disque (puisque les caches disques sont allouées de la mémoire libre).

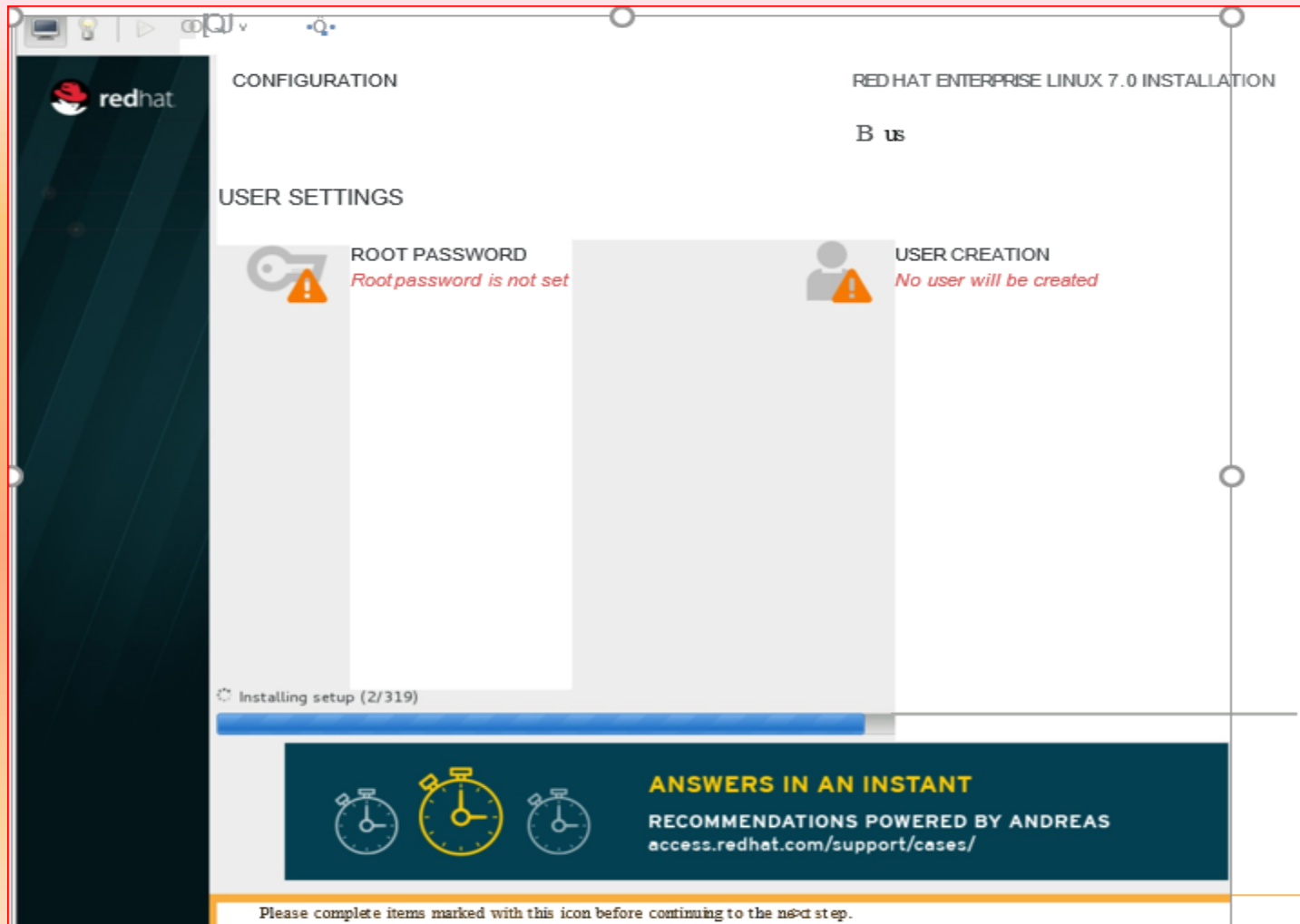
# Installation Personnalisée

- On peut également subdiviser le système en plusieurs partitions si désiré. Les raisons d'un tel partitionnement plus fin sont :
- **Maintenir le démarrage** sous la limite de 1024 cylindres (disques IDE et anciens BIOS) :
- Petit **/** séparé de **/usr**, ou carrément partition de démarrage **/boot**.
- **Performance** : les opérations I/O de lecture et écriture sur disque
- **Utilisation de plusieurs FS** : systèmes de fichiers différents (p.ex. : ext4, xfs....)

# Installation De Base



# Installation De Base





# LAB

# **Module 03:**

## **Les commandes de base partie1**

# La Ligne De commande

- Toutes les lignes de commande ont cette syntaxe:

**command**    **[options]**    **[arguments]**

Les lignes de commande peuvent être assemblées dans un fichier pour former un script.

- **Par exemple :**

-Pour afficher les information du système

**# uname -a** Afficher toutes les informations décrites ci-dessus

Afficher le type (matériel) de machine.

Afficher le nom d'hôte de la machine sur le réseau.

Afficher le numéro de version du système d'exploitation

# Les pages de manuel (Man pages)

- Les manuels en ligne décrivent la plupart des commandes utilisées dans votre système. **man [command]**

Exemples:

```
[root@test ] /# man mkdir  
[root@test ] /# man cal
```

- Pour rechercher un mot clé dans les pages de manuel, il faut utiliser l'option k

```
[root@test ] /# man -k compress  
[root@test ] /# apropos compress
```

- L'emplacement des pages de manuel peut être modifié avec la variable MANPATH. Pour afficher le contenu de MANPATH.

```
[root@test ] /# echo $MANPATH  
/usr/local/man:/usr/share/man:/usr/X11R6/man
```

# Aide : info, help, apropos, whatis

- **info commande**

info est constitué d'un ensemble de pages réparties en plusieurs niveaux, à travers lesquelles il est possible de naviguer de diverses façons.

- **help Commande**

affiche le manuel d'une commande interne (builtin)

- **apropos sujet**

affiche les pages de man correspondant au sujet

- **whatis Commande**

affiche une information succincte sur la commande

## Balade dans le «fs»

- Se déplacer dans le filesystem

**cd chemin (chemin relatif)   cd /chemin (chemin absolu)**

- Si vous êtes en administrateur système la commande par **cd ~** vous placera dans le répertoire **/root**.
- Dans le cas où je suis (je suis connecté en tant qu'utilisateur *user1*) je vais automatiquement me retrouver dans le répertoire de l'utilisateur *user1* qui se trouve dans **/home/user1**.
- Les répertoires des utilisateurs sont tous sous **/home**.

## Balade dans le «fs»

- pour connaître le chemin du répertoire où l'on se trouve est d'utiliser la commande **pwd** (print working directory) :

**# pwd**

- Pour changer de répertoire courant :

**# cd *répertoire***

- Le symbole (**~**) représente votre répertoire d'accueil
- Le symbole (**..**) représente le répertoire parent.
- Le symbole (**.**) représente le répertoire courant.

## commandes de base : ls

**ls option argument**

-a : affiche tous les fichiers (y compris ceux commençant par '.')

-l : listing étendu

-R : récursif

-S : tri par taille

-t : tri par date de modification

-1 : affichage sur une colonne

– exemples

ls \*.txt

ls /etc

ls /etc/host\*

ls /etc/rc[13].d

ls laR ~



## commandes de base : {mk,rm}dir

**mkdir [p] répertoire1 répertoire2 ...**

crée les répertoires répertoire1, répertoire2, ...

l'option -p per et destination (fichier ou répertoire)

– options

-p : crée les répertoires supérieurs si nécessaire

– exemples

mkdir \$HOME/documents

mkdir test \$HOME/images\_iso

mkdir p \$HOME/documents/personnel/{photos,factures}

## commandes de base : {mk,rm}dir

**rmdir répertoire1 répertoire2**

supprime les répertoires répertoire1, répertoire2, ...

ces répertoires doivent être vides pour pouvoir être supprimés  
(utiliser **rm -rf** sinon)

– exemples

**rmdir \$HOME/documents**

**rmdir test \$HOME/images\_iso**

**rmdir \$HOME/documents/personnel/{photos,factures,}**

## commandes de base : cp & mv

### **cp source destination**

copie la source (fichier ou répertoire) vers la destination (fichier ou répertoire)

– options

-p : préserve les droits

-R : récursif

-f : force l'écrasement de la destination

– exemples

**cp \*.txt /tmp**

**cp test.txt toast.txt**

**cp -Rf /home /var/backup**

## commandes de base : cp & mv

### **mv source destination**

- déplace la source (fichier ou répertoire) vers la destination (fichier ou répertoire).
- permet aussi de renommer un fichier

– exemples

```
mv *.txt /tmp
```

```
mv test.txt toast.txt
```

## commandes de base : rm

**rm**    **argument**

Supprime le fichier ou répertoire argument

– options

-R : récursif

-f : force la suppression

– exemples

**rm -rf /myrep1**

**rm toast.txt**

**Fait Attention !!!!!!!!!!!    rm -rf /\***

## commandes de base : touch

### **touch fichier**

Crée le fichier s'il n'existe pas, ou met la date de modification du fichier à l'heure courant s'il existe.

La commande '>' et 'echo' permet aussi de créer un fichier

– exemples

```
touch toast.txt
```

```
>file1.txt
```

```
echo > file2.txt
```

## commandes de base : cat, less

**cat fichier1 fichier2 ...**

affiche le contenu de fichier1 fichier2 ... sur la sortie standard  
si cat est appelé sans arguments, la source est l'entrée standard  
– exemple

**cat /var/log/messages**

**less fichier1 fichier2 ...**

comme cat, affiche le contenu de fichier1 fichier2 ... sur la sortie standard mais effectue un arrêt à chaque page si less est appelé sans arguments, la source est l'entrée standard  
– exemple

**less /etc/passwd**

## commandes de base : more, tee, wc

### **more** fichier

a l'avantage d'afficher le fichier page par page. Pour passer d'une page à l'autre, tapez sur la touche **ESPACE**

– exemple : `more /var/log/messages`

### **tee** fichier

duplique l'entrée standard vers la sortie standard et dans un fichier

– exemple : `vmstat 1 | tee toto`

### **wc** option fichier

compte le nombre de lignes (-l), bytes (-c), mots (-w) .... dans fichier (ou sur STDIN si aucun fichier n'est spécifié)

– exemple: `wc -l /etc/passwd`



## commandes de base : head & tail

**head [nX] fichier1**

affiche les X premières lignes de fichier1 sur la sortie standard

– exemple **head -n1 /etc/passwd**

**tail [nX] fichier1**

affiche les X dernières lignes de fichier1 sur la sortie standard

si tail est appelé sans arguments, la source est l'entrée standard et le nombre de lignes est 10

– exemple

**tail -n5 /var/log/syslog**

**tail -f /var/log/syslog**

## commandes de base : head & tail

**head [nX] fichier1**

affiche les X premières lignes de fichier1 sur la sortie standard

– exemple **head -n1 /etc/passwd**

**tail [nX] fichier1**

affiche les X dernières lignes de fichier1 sur la sortie standard

si tail est appelé sans arguments, la source est l'entrée standard et le nombre de lignes est 10

– exemple

**tail -n5 /var/log/syslog**

**tail -f /var/log/syslog**

## commandes de base : head & tail

Une combinaison des deux permet d'afficher la nième ligne d'un fichier :

```
head -n10 /etc/passwd | tail -n1
```

affiche la 9ème (10-1) ligne de /etc/passwd

```
head -n20 /etc/group | tail -n3
```

affiche les lignes 17 à 20 (20-3 -> 20) lignes de /etc/group

## commandes de base : cut

- La commande cut permet d'afficher des zones spécifiques d'un fichier.
- Par exemple :

**cut -c1 /etc/passwd**

affichera la première colonne du fichier /etc/passwd. Il existe d'autres spécifications

- On peut également spécifier un *séparateur de champs* avec l'option *-d*. Par exemple :

**cut -d: -f6 /etc/passwd**

- affichera le 6<sup>eme</sup> champ du fichier /etc/passwd, dont le séparateur de champs est le caractère double point (":").

## commandes de base : sort

- La commande **sort** trie les lignes d'un ou plusieurs fichiers de texte. Par défaut, le tri se fait suivant l'ordre lexicographique.
- Un certain nombre d'options sont fournies pour modifier l'ordre du tri :
  - **-n** pour tri numérique,
  - **-r** pour tri inversé,
  - **-k x** pour tri à partir du champ x,
  - **-t c** pour utiliser le séparateur de champs
- Quelques exemples de la commande sort :

```
ls -l | sort -n -k5
```

```
sort -n -k3 /etc/passwd
```

## commandes de base : uniq

Ce filtre élimine les lignes dupliquées depuis un fichier trié.  
On le voit souvent dans un tube combiné avec un **sort**

```
cat liste-1 liste-2 liste-3 | sort | uniq > liste.finale
```

Dans cet exemple

la 1<sup>er</sup> commande cat Concatène les fichiers liste1 list2 et list3

la 2em commande sort les trie,

la 3em commande uniq efface les lignes doubles,

et enfin > écrit le résultat dans un fichier de sortie.

## la commande sed

**sed** est un éditeur de ligne non interactif, il lit les lignes d'un fichier une à une leur applique un certain nombre de commandes d'édition et renvoie les lignes résultantes sur la sortie standard.

- Il ne modifie pas le fichier traité, il écrit tout sur la sortie standard.

-la syntaxe n'est franchement pas très conviviale, mais il permet de réaliser des commandes complexes sur des gros fichiers.

**La syntaxe de sed est la suivante:**

**sed -e 'programme sed' fichier-a-traiter**

**Ou**

**Sed -f fichier-programme fichier-a-traiter**

## la commande sed

La fonction de substitution **s** permet de changer la première ou toutes les occurrences d'une chaîne par une autre. La syntaxe est la suivante:

**sed "s/toto/TOTO/" fichier**

va changer la première occurrence de la chaîne toto par TOTO (la première chaîne toto rencontrée dans le texte uniquement)

**sed "s/toto/TOTO/3" fichier**

va changer la troisième occurrence de la chaîne toto par TOTO

Dans même ligne ex: ....toto ....toto.....**toto**.....

(la troisième chaîne toto rencontrée dans le texte uniquement)



# la commande split

**La commande split** permet de découper un fichier en plusieurs plus petits unités. Ses options sont :

- **b** *n* (Bytes) découpage par blocs de *n* octets
- ou
- **l** *n* (Lignes) découpage par blocs de *n* lignes

Syntaxe : `split fichier`

Exemple :

**# split -b 135000 vacances.mpeg**

# commandes de base : grep & egrep

**grep options patron fichier**

liste les fichiers contenant une chaîne correspondant à un patron.

– options

-v : inverse le comportement de grep (n'affiche que les lignes qui ne correspondent pas)

-i : insensible à la casse

-R : récursif

– patrons d'expression régulières

. n'importe quel caractère

\* le caractère précédent 0 ou plusieurs fois

+ le caractère précédent 1 fois au moins

[az] un caractère en minuscule

[azAZ] une lettre

[09] un chiffre

^/\$ le début/la fin de ligne

| séparateur pour spécifier de multiples expressions (ou logique)

## commandes de base : grep & egrep

- **Ex1:-** pour chercher tous les lignes qui contiennent le mot root

Dans le fichier /etc/group

**grep** root /etc/group

- **Ex2:** pour chercher tous les lignes qui ne contiennent pas le mot root dans le fichier /etc/group

**grep -v** root /etc/group

- **Ex3:-** pour chercher les noms des fichiers qui contient le mot root

**grep -l** root group passwd hosts

- **Ex4:** pour compter le nombre de lignes qui contiennent le mot root dans le fichier /etc/group

**grep -c** root /etc/group

## commandes de base : grep & egrep

- Matcher une ligne commençant par « foo » :

```
egrep "^foo.*"
```

- Matcher une ligne commençant par « foo » ou commençant par « bar » :

```
egrep "^(foo|bar) "
```

- Matcher une ligne commençant par « foo » et se terminant par « bar » :

```
egrep "^foo.*bar$"
```

## commandes de base : find

- La commande find permet de rechercher des fichiers au sein de l'arborescence du système de fichiers à l'aide de critères et donne la possibilité d'agir sur les résultats retournés.
- **find chemin critères options**
- La commande find étant réursive, il suffit d'indiquer un répertoire de base pour que toute l'arborescence depuis ce répertoire soit développée.

L'option de base est **-print** (souvent implicite sur la plupart des Unix) qui permet d'afficher sur écran les résultats.

find

## commandes de base : find

- Critères de recherche :

**-name** : `find . -name "fic*" -print`

**-type** : permet une sélection par type de fichier `find / -type d`

**-size** : permet de préciser la taille des fichiers recherchés

`find -size +100k`

**-mtime** : recherche sur la date de dernière modification

`find . -mtime -1`

**-perm** : permet d'effectuer des recherches sur les autorisations d'accès ex: `find -type d -perm -111`

**-inum** : -inum permet une recherche par numéro d'inode et

`find / -inum 13456`

## commandes de base : find

- **Commande**

- Outre l'option **-print** on trouve d'autres options permettant d'effectuer une action sur les fichiers trouvés.

- ls**: affiche des informations détaillées sur les fichiers trouvés

- exec** : La commande exécutée par **-exec** doit se terminer par un caractère spécial doit s'écrire `\` pour ne pas être interprété par le shell.

Pour passer comme paramètre pour la commande le fichier trouvé par **find**, il faut écrire `{ }` (substitution du fichier).

Exemple pour effacer tous les fichiers finissant par « .mp3 » :

```
find . -type f -name "*.mp3" -exec rm -f { } \;
```

## commandes de base : **whereis**, **which**, **locate**

- La commande **whereis** recherche dans les chemins de fichiers binaires, du manuel et des sources les fichiers correspondant aux critères fournis.

### **whereis** date

- La commande **which** recherche une commande dans le PATH (chemin des exécutables) et vous fournit la première qu'elle trouve

### **which** date

- La commande **locate** recherche un fichier selon le modèle donné dans une base de données de fichiers construite par la commande **updatedb**.

### **updatedb**

### **locate** toto



## commandes de base : ln

**Les liens** sont des fichiers spéciaux permettant d'associer plusieurs noms (liens) à un seul et même fichier

On distingue deux types de liens :

- **Les liens symboliques** représentant des pointeurs virtuels (raccourcis) vers des fichiers réels. En cas de suppression du lien symbolique le fichier pointé n'est pas supprimé.
- **Les liens physiques** (aussi appelées liens durs ou en anglais **hardlinks**) représentent un nom alternatif pour un fichier

## Commandes De Base : ln

**Les liens** sont des fichiers spéciaux permettant d'associer plusieurs noms (liens) à un seul et même fichier

On distingue deux types de liens :

- **Les liens symboliques** représentant des pointeurs virtuels (raccourcis) vers des fichiers réels. En cas de suppression du lien symbolique le fichier pointé n'est pas supprimé.
- **Les liens physiques** (aussi appelées liens durs ou en anglais **(hardlinks)**) représentent un nom alternatif pour un fichier
  - - Pour créer un lien symbolique :
  - **# ln -s source\_file target\_file**
  - Les liens physiques sont créées à l'aide de la commande ln (sans l'option -n) selon la syntaxe suivante :
  - **# ln nom-du-fichier-reel nom-du-lien-physique**

# LAB

# **Module 04:**

## **Les commandes de base partie2**

## VI Fonctionnement

- Il y a trois modes de fonctionnement :

**mode commande** : les saisies représentent des commandes. On y accède en appuyant sur **[Echap]**.

Chaque touche ou combinaison de touches déclenche une action (suppression de lignes, insertions, déplacement, copier, coller, etc.).

**mode saisie** : c'est la saisie de texte classique.

**mode ligne de commande** : une ligne en bas d'écran permet de saisir des commandes spéciales, validée avec la touche **[Entrée]**. On y accède en appuyant, en mode commande, sur la touche « : ».

# Les commandes VI

- **La saisie**
- Les actions suivantes sont à effectuer en mode commande. Elles doivent être précédées d'un appui sur **[Echap]** :

Commande	Action
<b>a</b>	Ajout après le caractère actuel.
<b>A</b>	Ajout de texte en fin de ligne.
<b>i</b>	Insertion devant le caractère actuel, comme dans un traitement de texte.
<b>I</b>	Insertion de texte en début de ligne.
<b>o</b>	Ajout d'une ligne sous la ligne actuelle.
<b>O</b>	Insertion d'une ligne au-dessus de la ligne actuelle.

# Les commandes VI

## - Quitter et sauvegarder

- Pour mémoire, les «:» signifient que la commande se tape en ligne de commande: **[Echap]** :, saisie de la commande, puis **[Entrée]**.

Commande	Action
<b>ZZ</b>	Sauve le fichier et quitte.
<b>:q!</b>	Quitte sans sauver.
<b>:q</b>	Quitte si le fichier n'a pas été modifié (apparition d'un message d'erreur sinon).
<b>:w</b>	Sauve le fichier. Vous pouvez préciser un nom à la suite.
<b>:wq</b> ou <b>:x</b>	Sauve et quitte.
<b>1,10w fic</b>	Sauve les lignes de 1 à 10 dans fic.

# Les commandes VI

## - Déplacement

- il est possible de se passer des touches fléchées, en mode commande.

Commande	Action
<b>h</b>	Aller vers la gauche.
<b>l</b> (petit L)	Aller vers la droite.
<b>k</b>	Aller vers le haut.
<b>j</b>	Aller vers le bas.
<b>0</b> (zéro)	Début de ligne.
<b>:0</b>	Début de fichier (première ligne).
<b>\$</b>	Fin de ligne.



# Les commandes VI

## - La correction

Commande	Action
<b>x</b>	Efface le caractère sous le curseur.
<b>X</b>	Efface le caractère devant le curseur.
<b>r&lt;c&gt;</b>	Remplace le caractère sous le curseur par le caractère <c>.
<b>dw</b>	Efface depuis le curseur jusqu'à la fin du mot.
<b>d\$</b> ou <b>D</b>	Efface depuis le curseur jusqu'à la fin de la ligne.
<b>dO</b>	Efface depuis le début de la ligne jusqu'au curseur.
<b>df&lt;c&gt;</b>	Efface tout jusqu'au caractère <c>.
<b>dG</b>	Efface tout jusqu'à la dernière ligne, y compris la ligne actuelle.
<b>d1G</b>	Efface tout jusqu'à la première ligne, y compris la ligne actuelle.
<b>dd</b>	Efface la ligne actuelle.
<b>u</b>	Undo. Annule la dernière action.

# Les commandes VI

## Recherche dans le texte :

- La commande de recherche est le caractère **/**

**ex:** **/echo**

- recherche la chaîne 'echo' dans la suite du fichier. Quand la chaîne est trouvée, le curseur s'arrête sur le premier caractère de cette chaîne.

## Copier Coller :

- pour couper (déplacer), c'est la commande « **d** » ;
- pour coller le texte à l'endroit choisi, c'est la commande **p**
- Pour copier une ligne : **yy**.

# Archivage et compression : tar, gzip & bzip2

- Vous pouvez utiliser l'archivage pour récupérer les données supprimés ou endommagés.
- Très utile si on veut envoyer un backup d'une machine à une autre via le réseau.
- L'avantage des fichiers compressés c'est qu'ils utilisent Moins d'espace disque et se téléchargent rapidement.
- **tar** : commande utiliser pour la création et l'extraction des fichiers a partir d'une archive
- **gzip, bzip2** : instruments et outils de compression

# Création D'archives

**tar** : commande utiliser pour la création et l'extraction des fichiers a partir d'une archive.

Syntaxe : **tar fonction(s) FichierArchive NomFichier**

Table des fonctions de la commande **tar**

Fonction	Définition
c	Création d'un nouveau fichier <b>tar</b>
T	Lister le contenu du fichier <b>tar</b>
X	Extraction d'un fichier a partir d'un fichier <b>tar</b>
F	Spécifier le fichier d'archive ou le périphérique de bande
V	Exécution en mode verbeuse
h	Affiche le lien symbolique comme standard fichier ou dossier

# tar: utilisation

- Pour Créer l'archive

```
[server@root]# tar cvf files.tar file1 file2 file3
```

- Pour visualiser le contenu d'une archive

```
[server@root]# tar tvf files.tar
```

file1

file2

File3

- Pour extraire l'archive

```
[server@root]# tar xvf files.tar
```

# gzip: utilisation

- La commande **Gzip** crée un fichier compressé terminé par .gz

```
[server@root]# gzip file1 file2 file3 file4
```

```
[server@root]# ls *.gz
```

```
file1.gz file2.gz file3.gz file4.gz
```

- La commande **Gunzip** extrait les fichiers compressés et efface le fichier .gz

```
[server@root]# gunzip file1.gz
```

# bzip2: utilisation

- compresser avec la commande bzip2

```
[server@root]# bzip2 file1 file2 file3
```

```
[server@root]# ls
```

```
file1.bz2 file2.bz2 file3.bz2
```

```
[server@root]# bzip2 -c -1 file1 > text.bz2
```

- c Compresser ou décompresser à un résultat standard.
- 1 Effectue une compression rapide, créant des fichiers relativement larges.

- Pour restaurer le fichier compressé :

```
[server@root]# bunzip2 text.bz2
```

# zip: utilisation

- Vous pouvez compresser et archiver a la fois en utilisant la commande zip

```
[server@root]# zip file.zip file1 file2 file3
```

```
-adding: file1 (deflated 48%)
```

```
-adding: file2 (deflated 16%
```

```
-adding: file3 (deflated 26%)
```

- Pour restaurer le fichier compressé :

```
[server@root]# unzip file.zip
```



# zip: utilisation

- vous pouvez compresser et archiver a la fois en utilisant la commande zip

```
[server@root]# zip file.zip file1 file2 file3
```

```
-adding: file1 (deflated 48%)
```

```
-adding: file2 (deflated 16%
```

```
-adding: file3 (deflated 26%)
```

- Pour restaurer le fichier compressé :

```
[server@root]# unzip file.zip
```

# LAB

# **Module 05:**

## **Les Scripts SHELL**

# LE SHELL

- Le bourne shell conçu par **Steve Bourne** en 1970 a été remplacé par le shell bash (**B**ourne **A**gain **S**hell) qui est devenu le standard sous Linux et est inclu dans toutes les distributions.
- Mais vous pouvez utiliser un autre Shell, voir même un shell différent pour chaque utilisateur

Shell	chemin	invite	link
bash	/bin/bash	#	/bin/sh
tcsh	/bin/tcsh	>	/bin/csh
ash	/bin/ash	\$	
zsh	/bin/zsh	%	

# Pipeline et redirection

- Deux caractéristiques fortes du CLI Linux.
- I/O redirection permet de rediriger le output standard | messages d'erreurs vers un fichier pour une analyse ultérieure.
- Les Pipes permet de connecter la sortie d'un programme avec une entrée d'un autre qui va le suivre dans l'exécution .
- Permet pour plusieurs petits programmes exécutés d'être enchaînés sur une seule ligne.

# Pipeline et redirection

Nom	Description	Numéro	Default
<b>STDIN</b>	Standard input	0	Keyboard
<b>STDOUT</b>	Standard Output	1	Terminal
<b>STDERR</b>	Standard Error	2	Terminal

- Trois différents « fichiers » sont toujours ouverts par défaut, **stdin** (le clavier), **stdout** (l'écran) et **stderr** (la sortie des messages d'erreur vers l'écran).
- **La redirection** signifie simplement **la capture** de la sortie d'un fichier, d'une commande, d'un programme, d'un script, voire même d'un bloc de code dans un script et le renvoi du flux comme entrée d'un autre fichier, commande, programme ou script.

# Opérateurs de redirection

Keyword	Définition	Exemple
>	Redirige le STDOUT vers un fichier 'écrasement'	date > fichier cat file1 file2 > fichier_total.txt
>>	Redirige le STDOUT vers un fichier 'ajout'	cat file1 file2 >> fichier_total.txt ls >> fichier
2>&1	Combine le STDERR avec STDOUT	find /etc -name passwd 2>&1  less
<	Redirige le STDIN	grep 'root' < /etc/passwd
	Envoie le stdout d'une commande vers le stdin de l'autre commande	ls /usr/lib64  grep '^lib'

# Les Opérateurs logiques

Keyword	Signification	Syntaxe explication	
<b>&amp;&amp;</b>	ET Logique	<b>Commande1 &amp;&amp; commande2</b>	la 2eme commande est exécuté uniquement si la 1 <sup>er</sup> commande renvoi un code vrai
<b>  </b>	OU Logique	cat file1 file2 >> fichier_total.txt ls >> fichier	



# Les Variables D'Environnements

- Une variable d'environnement est un espace mémoire qui possède un nom et une valeur qui peut être utilisée par plusieurs **processus**, un espace hérité d'un processus (père) et transmis à un processus (fils).
- Elles contiennent une **chaîne de caractères** comme par exemple la variable **PATH (\$PATH)** qui contient les emplacements des exécutables: **PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/sbin**
- Ces variables d'environnement sont toujours écrites en majuscules (sensibles à la casse) et permettent à des programmes d'utiliser ces informations en même temps

# Les Variables D'Environnements

- Pour connaître l'ensemble des variables d'environnement d'un système, vous pouvez avoir recours à la commande **env** .
- Ainsi, vous remarquez que le système stocke des variables contenant par exemple le nom de l'utilisateur courant (**\$USER**), le répertoire personnel (**\$HOME**) ou encore le shell (**\$SHELL**) ... Toutes ces informations vont être mises à disposition des programmes et de l'utilisateur.
- Les modifications apportées à une variable d'environnement seront spécifiques au processus dans lequel ou pour lequel elle aura été définie. On dit qu'elle a une portée locale.

# Les Variables D'Environnements: utilisation

- Comme toute nouvelle variable, celle-ci doit être déclarée. Cette déclaration est simple puisque vous utiliserez la syntaxe suivante:  
**nom=valeur**
- Cette variable pourra être un nombre ou une chaîne de caractère, sera utilisable dans le Shell et devra être précédé du signe « \$ » pour pouvoir être utilisée (comme ci-dessus par la commande `echo` ). La présence des doubles quote `''` oblige le shell à lire la chaîne de caractère et la valeur de la variable. L'utilisation de simple quote `' '` aurait amené le shell à considérer tout comme une chaîne.

# Les Variables D'Environnements: utilisation

- il existe 3 commandes principales qui permettent respectivement de mettre une variable dans l'environnement (**export**), de détruire une variable (**unset**) et de lister les fameuses variables (**env** ou **printenv**).
- Pour assigner une valeur à une variable, il suffit donc d'utiliser le signe d'affectation « = ». Cette variable n'aura d'existence dans ce cas que dans le terminal où elle aura été créée. On parlera alors de « **variable de shell** ». Pour que cette dernière devienne une variable d'environnement et soit mise à disposition des programmes, on utilisera la commande **EXPORT**:

```
root@server:~# EDITOR=gedit
```

```
root@server:~# export EDITOR
```

```
root@server:~# echo $EDITOR
```

```
gedit
```

# Les Variables D'Environnements: utilisation

- Pour effacer une variable d'environnement, rien de plus simple, utiliser **export + valeur vide**.
- Alors qu'avec UNSET, vous ferez totalement disparaître la variable:

```
root@server:~# export EDITOR=
```

```
root@server:~# echo $EDITOR
```

```
root@server:~# unset EDITOR
```

- Pour ajouter une valeur à une variable, vous utiliserez aussi export qui permet, dans l'exemple ci-dessous d'ajouter « valeur3 » dans \$EDITOR (vous remarquerez que les valeurs sont séparées par « : »).

```
root@server:~# EDITOR=valeur2
```

```
root@server:~# export EDITOR=$EDITOR:valeur3
```

```
root@server:~# echo $EDITOR
```

```
valeur2:valeur3
```

# Les Variables D'Environnements permanentes

- Toutes ces actions sur les variables seront perdues à l'arrêt de la session ou à l'arrêt du système. Il est bien entendu possible de définir ces variables de manière permanente. Elles pourront être définies permanente pour la session de l'utilisateur ou pour le système (quelque soit l'utilisateur).
- **pour le Shell bash** Il existe des fichiers de configurations d' environnements:
- **/etc/profile** : est un fichiers script système utilisé pour q' une variable  
soit définie pour l'ensemble du système .
- **\$HOME/.bash\_profile** fichiers script du répertoire personnel de l'utilisateur
- **\$HOME/.bashrc** : permet à chaque utilisateur de personnaliser son shell bash

# Exemples de script: utilisation de echo

*# visualisation des informations de l'utilisateur actuel avec la date*

*echo \*\*\*\*\**

*echo ''displaying user information''*

*echo \*\*\*\*\**

*echo ''Hello \$USER''*

*echo*

*echo ''today is `date`''*

*echo*

*echo ''Number of user login :*

*echo who | wc -l*

*echo ''Calendar ''*

*cal*

*#####*

# Exemples de script: utilisation de for loop

*Syntax    for {variable name} in {list}*  
*do    execute one for each item in the list*  
*done*

```
# testfor  
for i in 1 2 3 4 5  
do  
echo "welcome $i times"  
done
```

```
# tesfor2  
for (( i = 0 ; i <=5 ; i++ ))  
do  
echo "welcome $i times"  
done
```



# Exemples de script: utilisation de read

```
#!/bin/bash
```

```
# ceci est un commentaire, il sera ignoré par l'interpréteur
```

```
echo Quel est votre prénom ?
```

```
# echo affiche son argument
```

```
read prenom
```

```
# lit une variable entrée par l'utilisateur
```

```
echo Bonjour $prenom !
```

```
# Ne pas oublier le $ pour faire référence au contenu de la variable
```

```
# script to demo echo and read
```

```
## echo 'your good name please:'
```

```
read na
```

```
echo 'your age please : '
```

```
read age
```

```
neyr=`expr $age + 1 `
```

```
echo 'hello $na,next year you will be $neyr yrs old'
```

# Exemples de script: utilisation de if then else fi

lorsque vous programmerez des scripts, vous voudrez que vos scripts fassent une chose si une certaine condition est remplie et autre chose si elle ne l'est pas.

La construction de bash qui permet cela est le fameux test :

**if then else fi**

```
if grep -E "^user1:" /etc/passwd > /dev/null ; then
    echo L'utilisateur user1 existe.
else
    echo L'utilisateur user1 n'existe pas.
fi
```

# LAB



# **Module 06:** **Les utilisateurs et les droits**

# Gestion Des Comptes utilisateurs

Le système, dès son installation, avant même la première connexion au système crée des **utilisateurs systèmes**.

Un utilisateur n'est donc pas uniquement une personne physique, le système a besoin d'utilisateurs pour sa gestion interne, notamment comme propriétaire des divers processus.

La commande **ps aux | less** montre qu'avant toute connexion d'utilisateur humain (repérée par les lignes **login --user**), root a lancé init, et la plupart des services, **crond, inetd, lpd, smbd, ...**, avant de lancer les connexions utilisateurs dans les consoles, y compris éventuellement la sienne !

# Le fichier /etc/passwd

Ce fichier possède un format spécial permettant de repérer chaque utilisateur

Sept champs sont explicités séparés par le caractère ":"

- 1- le **nom du compte** de l'utilisateur
- 2- le **mot de passe** de l'utilisateur (codé bien sûr)
- 3- l'**entier** qui identifie l'**utilisateur** pour le système d'exploitation (*UID=User ID, identifiant utilisateur*)
- 4- l'**entier** qui identifie le **groupe** de l'utilisateur (*GID=Group ID, identifiant de groupe*)
- 5 - le **commentaire** dans lequel on peut retrouver des informations sur l'utilisateur ou simplement son nom réel
- 6 - le **répertoire de connexion** qui est celui dans lequel il se trouve après s'être connecté au système
- 7- la **commande** est celle exécutée **après connexion** au système (c'est fréquemment un interpréteur de commandes)

# UID & GID

**UID** : identifiant (unique) de chaque compte utilisateur.

**uid 0** est spécifié pour l'utilisateur root

**uid 1-200**: ID des comptes systèmes assignés statiquement aux processus systèmes par redhat

**uid 200-999**: ID des comptes systèmes assignés automatiquement aux logiciels installés pour faire tourner leur services avec comptes sans privilège

**uid 1000+**: réserver aux utilisateurs réguliers

**GID** : identifiant de groupe.

La commande **id** permet de lister les informations de l'utilisateur actuel

[student@Desktop0] \$ **id**

*Uid=1000(student) gid=1000(student) groups=1000(student)*

## Le fichier `/etc/group`

Le fichier `/etc/group` contient la liste des utilisateurs appartenant aux différents groupes.

En effet, lorsque de nombreux utilisateurs peuvent avoir accès au système, ceux-ci sont fréquemment rassemblés en différents groupes ayant chacun leurs propres droits d'accès aux fichiers et aux répertoires.

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

Le champ spécial est fréquemment vide.

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



## Le fichier `/etc/shadow`

Étant donné que le fichier `/etc/passwd` doit être examinable par tout utilisateur (la raison principale étant que ce fichier est utilisé pour effectuer la conversion de l'UID en nom d'utilisateur),

il est risqué de stocker les mots de passe de tous les utilisateurs dans `/etc/passwd`.

Certes, les mots de passe sont codés. Néanmoins, il est tout à fait possible d'effectuer une attaque contre des mots de passe si leur format codé est disponible

## su - sudo

- **Switcher en utilisateur root**

utiliser la commande **su** (switch user – changer d'utilisateur)

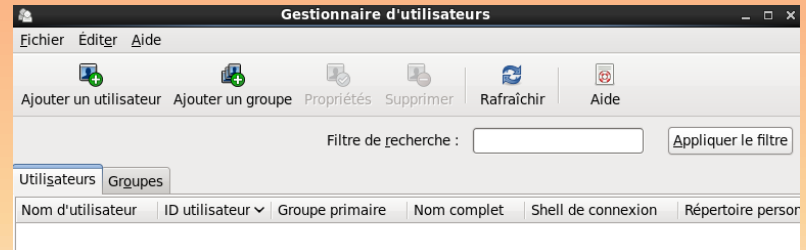
Aussitôt le mot de passe vous sera demandé.

l'invite passera en **#** au lieu de **\$**.

- On peut obtenir la même chose avec la commande **sudo**. Si l'utilisateur est défini dans le fichier **/etc/sudoers** pour exécuter des commandes d'administration. Ou appartient à un groupe avec privilèges élevé.
- Dans RHEL7 le group les membres de group **wheel** peuvent exécuter la commande **sudo**. Cette pratique permet d'avoir des traces de chaque utilisateur dans le fichier **/var/log/secure**
- si vous utilisez un logiciel en interface graphique qui requiert l'accès au root, une fenêtre va s'ouvrir vous demandant votre mot de passe

# Gestion des utilisateurs

- **Création : useradd & newusers**  
`/etc/skel`  
`/etc/default/useradd, /etc/login.defs`  
newusers permet d'ajouter en lot
- **Modification :**  
`usermod –argument valeur`
- **Suppression :**  
`userdel [-r]`
- **Mode graphique :**  
Via `system-config-users`



# Gestion des mots de passe

Par défaut les mots de passe n'expirent pas

Forcer l'expiration des mots de passe fait partie d'une politique de sécurité forte

Modifiez les paramètres d'expiration par défaut dans `/etc/login.defs`

Pour modifier les mots de passe des utilisateurs existants :

Modifier `/etc/shadow` manuellement

Utiliser la commande

**chage** [option] nom\_utilisateur

# Gestion des groupes

- Création de groupes secondaires
  - **groupadd [-g GID] group\_aux**
- Ajouter des utilisateurs à un groupe :
  - # **usermod -aG** group\_aux nom\_utilisateur
  - # **gpasswd -a** nom\_utilisateur group\_aux
  - # **vigr**
- Renommer :
  - **groupmod**
- Supprimer :
  - **groupdel**

# LAB

# Types de Permissions

- Les droits de bases que possède tout fichier ou répertoires sont au

\* Nombre de trois qui sont :

**r** ead lecture

**w** rite écriture

**ex** ecute exécution

- **Mode**

Le mode d'un fichier est l'ensemble des permissions qui lui est associé.

# Types de Permissions

- **Les permissions liées aux fichiers**

- **r=read**

Pour pouvoir afficher le contenu des fichiers, l'utilisateur doit avoir l'autorisation de read sur ces fichiers. L'autorisation de write sur le répertoire n'est pas indispensable.

- **w=write**

Le droit write implique qu'un utilisateur peut changer le contenu de fichiers.

- **x=execute**

Le droit execute sur un fichier implique qu'un utilisateur peut exécuter ce fichier comme par exemple une commande UNIX. Cet aspect est important surtout lors de l'utilisation de scripts shell.



# Types de Permissions

- **Les permissions liées aux répertoires**

- **r=read**

Pour un répertoire, les privilèges read et execute permettent à un utilisateur d'exécuter la commande `ls -l` dans le répertoire ciblé afin de lister son contenu. Avec read seulement, la commande `ls` s'exécute mais `ls -l` ne donne pas les informations étendues.

- **w=write**

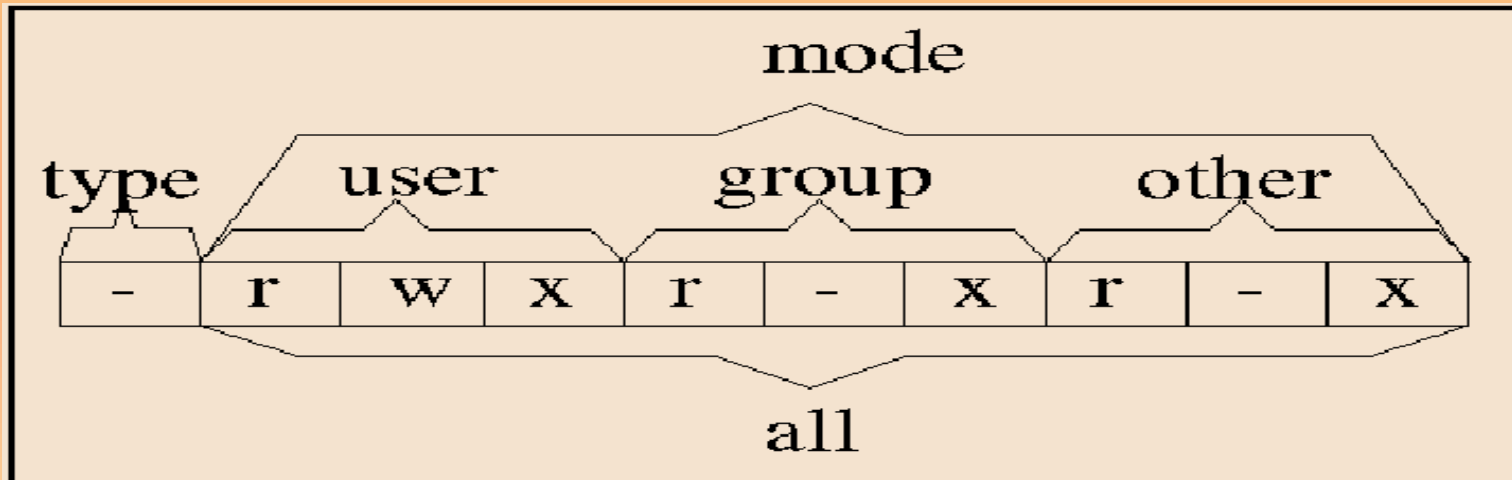
Le droit write pour un répertoire vous permet de créer ou de supprimer des fichiers dans ce répertoire

- **x=execute**

La possibilité de se déplacer dans l'arborescence avec `cd` ou de lister avec détails les fichiers avec `ls -l` est aussi fonction du droit execute positionné sur les répertoires

# Types de Permissions

- *Le mode des fichiers ou répertoires*
- Le mode d'un fichier est composé de plusieurs champs. Ils représentent les autorisations d'accès effectives pour les 3 "classes" d'utilisateurs d'un système Unix.



# Utilisation de la Notation Symbolique

La première méthode pour positionner ou changer les permissions est la notation symbolique. L'élément séparateur est la virgule.

- **Classes**

u utilisateur (propriétaire)

**g** groupe

**o** autres

**a** tous

- **Opérations**

= affectation d'une ou plusieurs permissions

- suppression

+ positionnement

- **Permissions**

**r** read lecture

**w** write écriture

**x** execute exécution

# Positionnement des permissions : chmod

**Exemples : Notez dans le champ libre le nouveau mode obtenu**

**\$ chmod a=rwx fichier**

**\$ ls -l fichier**

-rwxrwxrwx 1 user1 staff 0 Nov 11 20:12 fichier

**\$ chmod g-rwx,o-rwx fichier**

**\$ ls -l fichier**

-rwx- - - - - 1 user1 staff 0 Nov 11 20:12 fichier

**\$ chmod go+rx fichier**

**\$ ls -l fichier**

-rwxr-xr-x 1 user1 staff 0 Nov 11 20:12 fichier

**\$ chmod g+w,o-r fichier**

**\$ ls -l fichier**

-rwxrwx- -x 1 user1 staff 0 Nov 11 20:12 fichier

# Utilisation de la Notation Octale

La seconde méthode pour positionner ou modifier les permissions est l'utilisation de la méthode octale. Chaque permission est assignée à une valeur constante.

<i>Valeur</i>	<i>Permission</i>
4	lecture
2	écriture
1	exécution
0	aucune

# Utilisation de la Notation Octale

Exemples:

```
$ chmod 777 fichier1
```

```
$ ls -l fichier1
```

```
-rwxrwxrwx 1 user1 staff 0 Nov 11 20:12 fichier1
```

```
$ chmod 714 fichier1
```

```
$ ls -l fichier1
```

```
-rwx- -xr- - 1 user1 staff 0 Nov 11 20:12 fichier1
```

```
$ chmod 600 fichier1
```

```
$ ls -l fichier1
```

```
-rw- --- --- 1 user1 staff 0 Nov 11 20:12 fichier1
```

# chmod - Tableau récapitulation

Notations symboliques					
Classe		Opérations		Permissions	
u	user (propriétaire)	=	assigne	r	lecture
g	group	-	retire	w	écriture
o	others	+	donne	x	exécution
a	all				
Notations octales					
Valeur	Permissions		Explication		
7	rwx		lecture, écriture, exécution		
6	rw-		écriture, lecture		
5	r-x		lecture, exécution		
4	r--		lecture		
3	-wx		écriture, exécution		
2	-w-		écriture		
1	--x		exécution		
0	---		aucun accès		

# Utilisation de umask

**\$ umask**

**022**

Le 1<sup>er</sup> chiffre représente les permissions par défauts du propriétaire

Le 2eme chiffre représente les permissions par défauts du group

Le 3eme chiffre représente les permissions par défauts de tous

*Exemples :*

*\$ umask*

*022*

*\$ touch fichier01*

*\$ ls -l fichier01*

*-rw-r- -r- - 1 user1 staff 0 Nov 11 20:12 fichier01*

*\$ umask 027*

*\$ umask*

*027*

*\$ touch fichier02*

*\$ ls -l fichier01*

*-rw-r- - - - - 1 user1 staff 0 Nov 11 20:12 fichier02*



# Permission spéciale : SUID, SGID & Sticky Bit

- **SUID** L'attribut indique que, lors de l'exécution du programme, c'est l'identifiant du propriétaire du fichier qui est utilisé plutôt que celui de l'utilisateur l'ayant lancé. Cela se passe donc comme si c'est ce propriétaire qui le fait s'exécuter.
- **SGID** est utilisé pour créer un répertoire de collaboration
  - Lorsqu'un fichier est créé dans répertoire avec le bit SGID défini, il appartient au même groupe que le répertoire, plutôt qu'au groupe principal du créateur
  - # **chmod g+s nom\_répertoire**
- **Sticky bit** permet au propriétaire d'un fichier de le supprimer uniquement
  - Généralement, les utilisateurs dotés d'autorisations d'écriture sur un répertoire peuvent supprimer tout fichier contenu dans ce répertoire sans se soucier des autorisations ou du propriétaire du fichier
  - # **chmod o+t nom\_répertoire**

# Les Permissions spéciales

Pour la notation octale, un chiffre supplémentaire est indiqué devant les trois autres. Sa valeur est calculée en ajoutant les valeurs associée aux attributs selon le tableau suivant :

Valeurs symboliques et octales des attributs spéciaux		
Attribut	Valeur symbolique	Valeur octal
SUID	s (pour l'utilisateur)	4
SGID	s ( pour le group)	2
Sticky-bit	t	1

# Les Permissions spéciales

le programme **chmod** peut être utilisé avec les notations symbolique ou octale. On spécifie ensuite un ou plusieurs fichiers auxquels appliquer les valeurs spécifiées.

```
> ls -l /usr/bin/passwd  
-rws r-x r-x
```

permet à tout le monde (utilisateur, groupe, autres) de lire et exécuter le fichier, alors que seul le propriétaire peut le modifier. Le 4 indique que de plus l'attribut SUID doit être mis.

```
> chmod u=rwx,go=rx /usr/bin/passwd
```

# Contrôle d'accès aux ressources : ACL

- Autoriser ou refuser l'accès à plusieurs utilisateurs ou groupes
  - Les utilisateurs non '**root**' ne peuvent pas utiliser **chown** pour les fichiers
  - Empêche les utilisateurs de partager des fichiers avec **chmod 777**
  - Utilise les mêmes autorisations **rwX**
- Implémenté comme option de la commande **mount(acl)**
  - Incorporé dans un superbloc de système de fichiers lors de l'installation
- Des utilitaires/scripts de sauvegarde doivent éventuellement être MAJ pour la prise en charge
- Avec la RHEL 7 et le NFS v4, les ACLs peuvent être partagées via le réseau

# Syntax : ACL

- **Ajouter les ACL**

Pour ajouter une ACL, vous devez utiliser la commande **setfacl**

**setfacl -m permissions fichierOuDossier**

- les permissions s'écrivent sous cette forme :

**préfixe:[utilisateurOuGroupe:]droits**

Les préfixes disponibles sont :

**u:** Pour modifier les droits d'un utilisateur

**g:** Pour modifier les droits d'un groupe

**o:** Pour modifier les droits du reste du monde (other)

Les droits s'écrivent sous la forme d'un triplet **rwX** :

**r** = droit de lecture

**w** = droit d'écriture

**x** = droit d'exécution pour les fichiers, pour les dossiers

Pour ne pas attribuer un droit, vous pouvez ne pas écrire sa lettre correspondante ou la remplacer par un tiret (**r--** est équivalent à **r**)

# Contrôle d'accès aux ressources : ACL suite

- Affichage des ACLS :
  - `getfacl nom_fichier`
- Ajout / Modification des ACLS :
  - `setfacl -m u:nom_utilisateur:rwX nom_fichier`
- Suppression des ACLS :
  - `setfacl -x u:nom_utilisateur nom_fichier`

# Exemples : ACL

- **Exemples :**

- autoriser à "utilisateur1" la lecture et l'écriture sur "fichier"

**# setfacl -m u:utilisateur1:rw fichier**

- modifier les permissions de plusieurs utilisateurs/groupes sur "fichier" en même temps

**# setfacl -m u:utilisateur1:rwx,u:utilisateur2:r,g:groupe1:rw fichier**

- définir l'accès en lecture par défaut pour "utilisateur1" pour les nouveaux fichiers créés dans "dossier"

**# setfacl -m d:u:utilisateur1:r dossier**

- Pour dupliquer les mêmes acl d un fichierA sur un autre fichierB il suffit de utiliser le output getfacl:

**# getfacl fichierA | setfacl --set-file=- fichierB**

# LAB



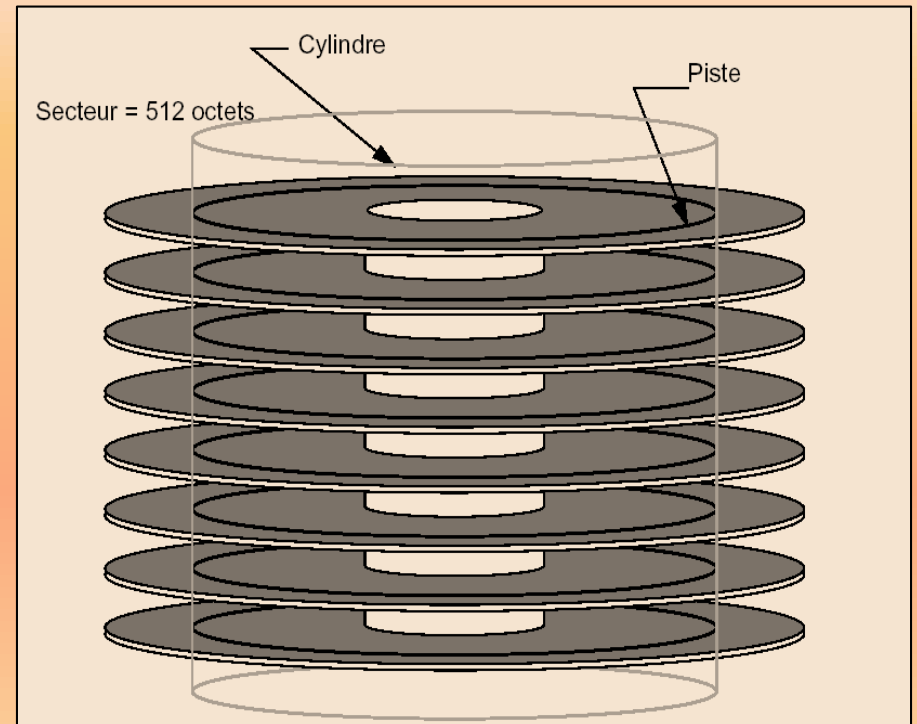
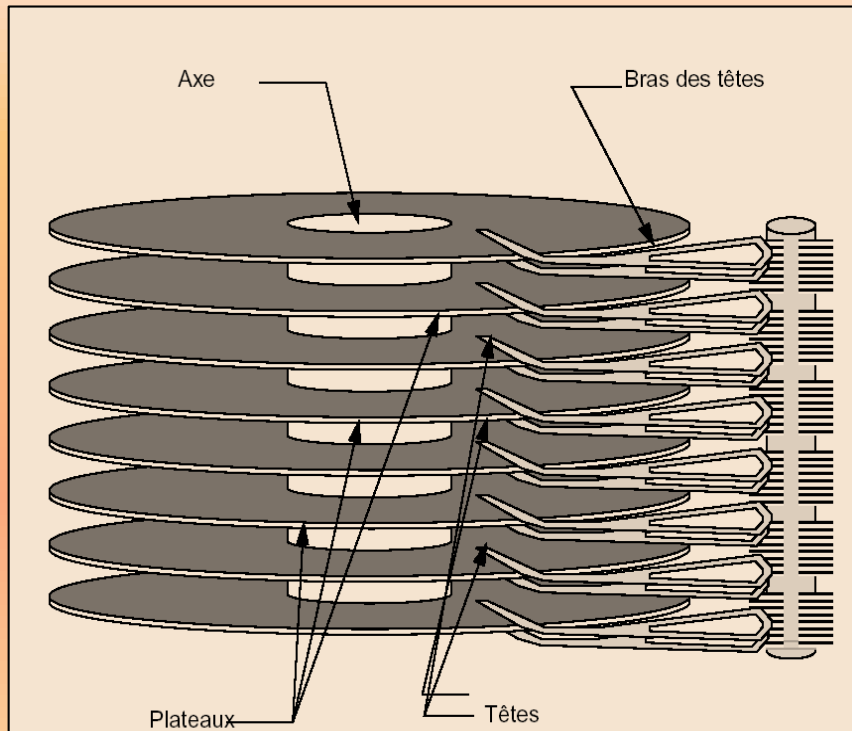


# **Module 07:** **Gestion du système de fichier**

# Gérer les disques basiques

- Un disque ne peut contenir que **4 partitions primaires**. Mais il est possible de remplacer une partition primaire par une partition étendue.
- Cette partition étendue peut contenir maximum **12 partitions logiques**.
- Notre disque pourra donc, au maximum, contenir **15 partitions** utilisables soit 3 primaires plus les 12 partitions logiques.

# La géométrie du disque



# Ajout de nouveaux FS

**1-** Les disques sont divisés en partitions :

Principale, étendue, logique

```
# fdisk /dev/sdc
```

**2-** Les partitions sont formatées avec des systèmes de fichiers pour permettre aux utilisateurs de stocker les données.

```
# mkfs -t ext4 /dev/sdc1
```

**3-** Les FS sont montés dans l'arborescence avant l'accès

```
# mount /dev/sdc1 /data
```

# Partitions et systèmes de fichiers

- La sauvegarde de la table des partitions

```
# sfdisk -d /dev/sda > /tmp/partitions.sda
```

- Procéder au partitionnement :

```
# fdisk /dev/sda
```

- Restauration de la table des partitions après une erreur

```
# sfdisk /dev/sda < /tmp/partitions.sda
```

- Mettre à jour le /proc/partitions :

```
# partprobe /dev/sda
```

# Création du système de fichiers

- Créer le système de fichiers avec un Label

```
# mkfs -t ext4 -L guest_data /dev/sda5
```

- **mkfs.ext4** [options] device

```
# mkfs.ext4 -t /dev/sdb6
```

- Appelle des utilitaires de système de fichiers spécifiques, tels que **mke2fs**

# Le montage des FS et le fameux FSTAB

- Création du point de montage

```
# mkdir -p /srv/guest_data
```

- Montage d'un FS :

```
# mount -o acl LABEL=guest_data /srv/guest_data
```

- le FSTAB :

- Maintient la hiérarchie à travers les redémarrages du système
- Utilisé par mount, fsck et d'autres programmes
- Peut utiliser des labels de volume du système de fichiers dans le champ périphérique
- **mount -a** monte tous les systèmes de fichiers **auto** dans /etc/fstab
- Recommandé pour le test de la syntaxe **fstab** avant redémarrage  
!

# Ajout de mémoire virtuelle « SWAP »

- **Le SWAP** correspond à l'espace du disque dur qui étend la RAM du système
- Peut être un fichier ou une partition « simple ou LVM »

```
# dd if=/dev/zero of=/var/local/swapfile bs=1k  
count=2M
```

– **mkswap device**

```
# mkswap -c /dev/sdb3
```

- Ecriture d'une signature spéciale :

```
# mkswap /var/local/swapfile
```

- Ajouter une entrée dans le /etc/fstab
- Activation du fichier swap via **swapon -a**



# LAB