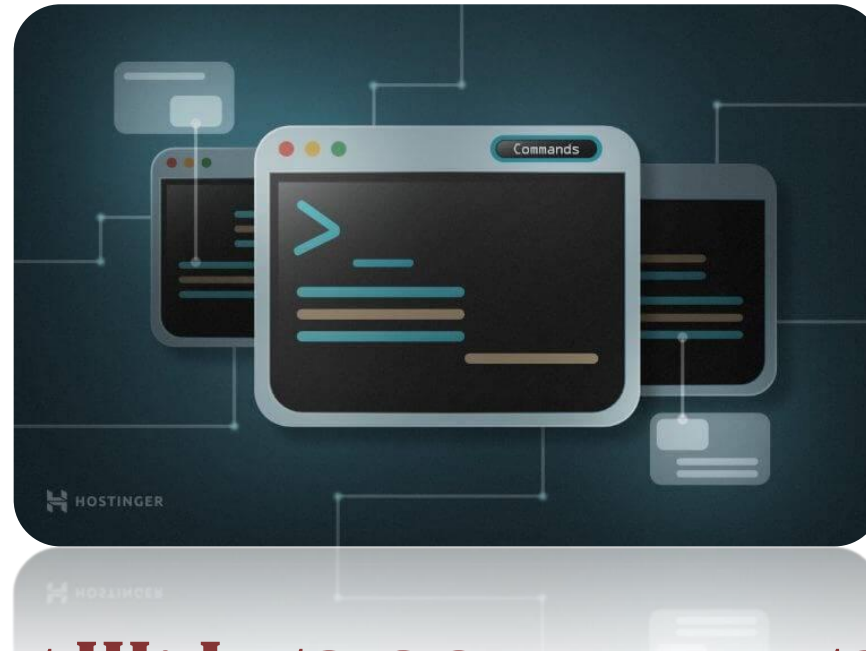


Support de cours

SYSTÈMES D'EXPLOITATION



CHAPITRE III: LES COMMANDES LINUX I

Plan

- ☐ **Introduction**
- ☐ **Se documenter sur le fonctionnement de Linux**
- ☐ **Manipulation de fichiers**
- ☐ **Contrôler l'accès aux fichiers**
- ☐ **Gestion des utilisateurs**
- ☐ **Gestion des processus**

→Linux- Les commandes

Syntaxe :

- commande [options] <arguments>
- séparateur : caractère espace

ATTENTION : Toutes les commandes s'écrivent en minuscule.

Commande

- Action à accomplir ou application à démarrer

Arguments

- Objets ou fichiers auxquels la commande s'applique

Options

- Modification du comportement de la commande
- Commencent généralement par un - (moins)

→ La commande: **man**

Syntaxe

man *[nom de la commande]*

Description

- Permet d'accéder à la documentation d'utilisation d'une commande (i.e. les pages de man).
- Les pages de man décrivent les syntaxes, les options, les arguments des commandes.
- Elles décrivent les résultats des évaluations et le format de ces résultats.

Exemple

man ls : affiche la page de manuel pour la commande **ls**

➤ Pour obtenir des man pages en français : `sudo apt-get install manpages-fr`

Manipulation de l'arborescence des fichiers

Syntaxe

pwd

Description

- Affiche le nom du répertoire courant
-

Syntaxe

cd

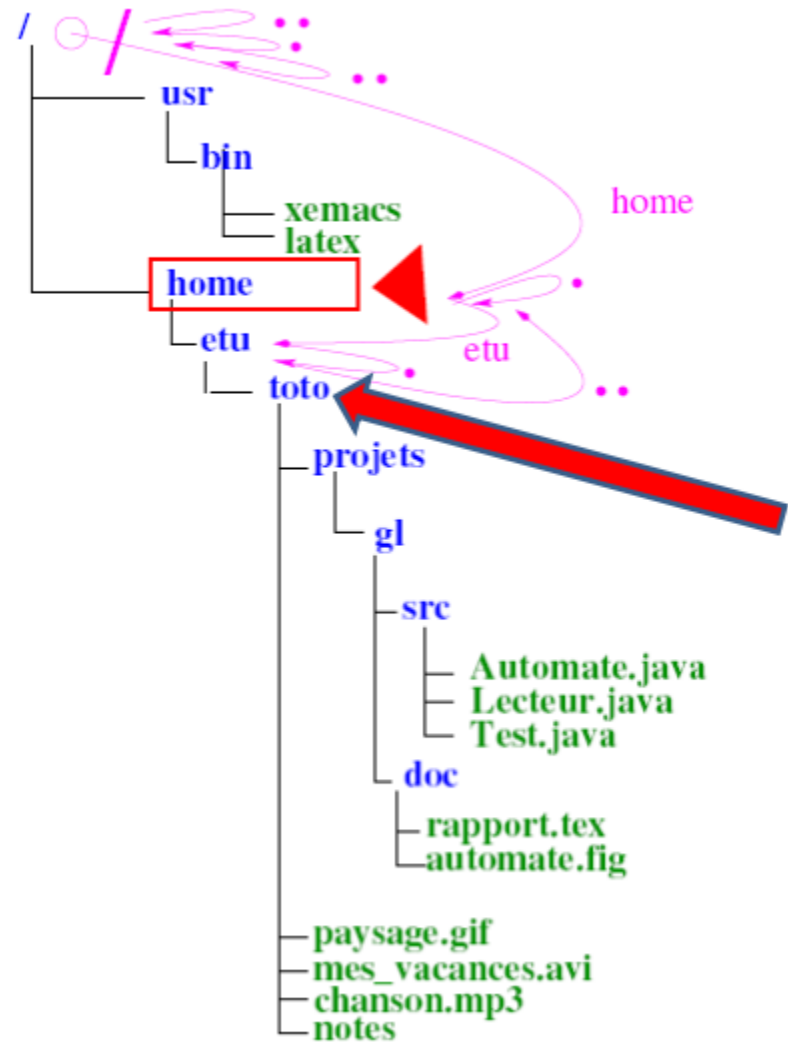
Description

- permet de changer de répertoire courant.

Manipulation de l'arborescence des fichiers

Exemple:

```
$ pwd
/home/etu/toto
$ cd projets/gl
$ pwd
/home/etu/toto/projets/gl
$ cd ..
$ pwd
/home/etu/toto/projets
$ cd ../../../../etu/toto
$ pwd
/home/etu/toto
$ cd /usr/bin
$ pwd
/usr/bin
$ cd ../../../../home/./etu/../../
$ pwd
/home
```



Manipulation de l'arborescence des fichiers

Syntaxe

ls [options] [nom_fichiers]

Description

- Affiche le contenu d'un répertoire.
- Par défaut, si aucun nom de fichier répertoire n'est indiqué, la commande affiche le contenu du répertoire courant.

Options

-R	Afficher récursivement le contenu des sous-répertoires.
-a	Afficher tous les fichiers des répertoires, y compris les fichiers commençant par un '.'
-i	Afficher le numéro d'index (i-noeud) de chaque fichier à gauche de son nom.
-l	En plus du nom, afficher le type du fichier, les permissions d'accès, le nombre de liens physiques, le nom du propriétaire et du groupe, la taille en octets, et l'horodatage.

Manipulation de l'arborescence des fichiers

Exemple

\$ **ls -ail**

718024	d	rw	xr-xr-x	3	estbm	cigm	4096	2023-01-24	21:13	.
2769	d	rw	xr-xr-x	33	estbm	cigm	4096	2023-01-24	20:29	..
718024	-	rw	xr-xr-x	2	estbm	cigm	231	2023-01-24	21:25	toto

numéro d'inode	Le type du fichier	Les droits	nombre de liens physique sur le fichier	Propriétaire	Groupe du propriétaire	la taille du fichier	La date de dernière modification du fichier	Le nom du fichier
----------------	--------------------	------------	---	--------------	------------------------	----------------------	---	-------------------

Manipulation de l'arborescence des fichiers

- rwx r-x ---
t u g o

```
graph TD; A["- rwx r-x ---  
t u g o"] --> B["Type :  
d répertoire  
b type bloc  
c type caractère  
l lien symbolique  
s socket  
p pipe"]; A --> C["u : user (utilisateur)  
g : group  
o : other (autre)  
r : read (lecture)  
w : write (écriture)  
x : execute (exécution)"]
```

Type :

d répertoire
b type bloc
c type caractère
l lien symbolique
s socket
p pipe

u : user (utilisateur)
g : group
o : other (autre)
r : read (lecture)
w : write (écriture)
x : execute (exécution)

Manipulation de l'arborescence des fichiers

Syntaxe

basename [*nom_fichiers*]

Description

- Elimine les répertoires en tête du chemin d'accès du fichier.

Syntaxe

dirname [*nom_fichiers*]

Description

- La commande **dirname** ne conserve que les répertoires en tête du chemin d'accès du fichier.

Exemple

```
$ basename /usr/local/bin/lynx  
lynx
```

```
$ dirname /usr/local/bin/lynx  
/usr/local/bin
```

Manipulation de fichiers

Syntaxe

file [options] [nom_fichier]

Description

- Affiche le type d'un fichier (répertoire, exécutable binaire,...) .

Options

- **-b** pour que la sortie affichera uniquement le type de fichier en omettant le nom de fichier

Exemple

```
$ file demo.php  
demo.php: PHP script, ASCII text
```

```
$ file -b demo.php  
PHP script, ASCII text
```

Manipulation de fichiers

Syntaxe

mkdir [options] [chemins/nom_rép]

Description

- Création d'un ou de plusieurs répertoires aux endroits spécifiés par les chemins.
- Si le chemin est occupé par un fichier ou un répertoire, il y a un message d'erreur

Options

-p : Créer les répertoires parents s'ils manquent

Exemple

\$ ***mkdir monrep***

Manipulation de fichiers

Syntaxe

rmmdir [options] [chemins/nom_rép]

Description

- supprime un répertoire vide indiqué
- **rm -r** : pour supprimer récursivement des répertoires non-vides,

Options

- **-p**: Pour supprimer un sous-répertoire vide et son répertoire parent.

Exemple

\$ **rmmdir monrep_vide**

\$ **rmmdir monrepNvide**

failed to remove 'monrepNvide': Directory not empty

\$ **rm -r monrepNvide**

Attention ! Si vous supprimez un dossier non vide à l'aide **rmmdir -r**, tous les répertoires et fichiers qu'il contient seront définitivement supprimés.

Manipulation de fichiers

Syntaxe

mv *[chemin_source] [chemin_cible]*

Description

- Déplace/Renomme un fichier ou répertoire.
- Modifie le chemin d'accès à la source qui devient le chemin cible.
- Le chemin source disparaît et le chemin cible est créé.
- Le fichier ou répertoire pointe reste le même.
- La cible doit être un chemin non occupé ou un répertoire

Options

- b** (b=backup) :effectue une sauvegarde des fichiers avant de les déplacer. La copie porte le même nom suivi d'un tilde.
- i** (i=interactive) :demande confirmation avant pour chaque fichier.
- u** (u=update) :pour ne pas supprimer le fichier si sa date de modification est postérieure à celle du fichier remplaçant.

Manipulation de fichiers

Exemple

```
$ ls -i
65338 fichier1 65340 fichier2
$ mv fichier1 fichier3
$ ls -i
65340 fichier2 65338 fichier3
$ mv fichier2 ..
$ ls -i ../fichier2
65340 ../fichier2
```

Manipulation de fichiers

Syntaxe

cp *[options] [chemin_source] [chemin_cible]*

Description

- Copie le fichier source vers la cible.
- La source doit être un fichier ordinaire (pas un répertoire),
- Si la cible :
 - est le chemin d'un répertoire existant, le fichier sera copie dans ce répertoire et conservera son nom.
 - ne correspond pas a un répertoire existant, le fichier sera copie avec le nom cible.

Options

- R** :recopie récursive, permet de copier toute une arborescence
- i** :avertit l'utilisateur de l'existence d'un fichier du même nom et lui demande s'il veut le remplacer.
- p** :effectue une copie en gardant le propriétaire et le groupe d'origine.
- v** :affiche en clair le nom des fichiers copiés.

Manipulation de fichiers

Exemple

```
$ ls -il
65338 -rw-r--r-- 1 SBakkkouri cigm 0 20 jan 17 :58 fichier3
$ cp fichier3 fichier4
$ ls -il
65338 -rw-r--r-- 1 estbm cigm 0 20 jan 17 :58 fichier3
65341 -rw-r--r-- 1 estbm cigm 0 20 jan 18 :01 fichier4
$ mkdir -p Rep1/ESTBM
$ cp fichier3 Rep1
$ ls Rep1
Fichier3 ESTBM
$ cp -r Rep1 Rep2
$ ls -R Rep2
fichier3 Rep1/ESTBM
```

Manipulation de fichiers

Syntaxe

rm [options] [chemin/nom_fich]

Description

- Supprime le fichier pointé par le(s) chemin(s).

Options

- **-d** : Supprime un répertoire vide à l'aide de **rm**.
- **-r** : Supprime un répertoire non vide et son contenu.
- **-f** : cette option force la suppression même si le fichier n'est pas disponible pour l'écriture.
- **-i** : Affiche une invite avant la suppression de chaque fichier.

Manipulation de fichiers

Exemple

→ Supprimer un répertoire non vide et ses sous-répertoires sous Linux

```
$ rm -d Directory_1
```

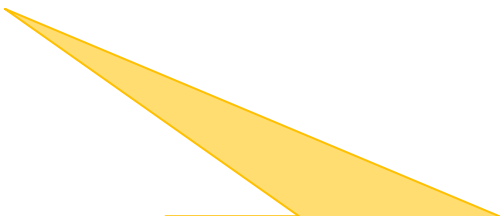
```
$ rm -r Directory_1 Directory_2 Directory_3
```

→ Supprimer un ou plusieurs un fichier sous Linux

```
$ rm file.txt
```

```
$ rm file1.txt file2.txt file3.txt
```

```
$ rm -i file1.txt file2.txt file3.txt
```



Le terminal vous demande de confirmer chaque suppression de fichier.

→ Les liens

Les liens sont utiles pour faire apparaître un même fichier dans plusieurs endroits, même avec des noms différents.

Les types des liens : il existe deux types de liens, à savoir :

- a) **Lien symbolique**
- b) **Lien physique**

→ Les liens symboliques

- Fait référence à un fichier dans un répertoire.
- Si suppression du fichier source alors le lien sera considéré comme "cassé".
- Utile dans le cas des fichiers binaires (commandes)

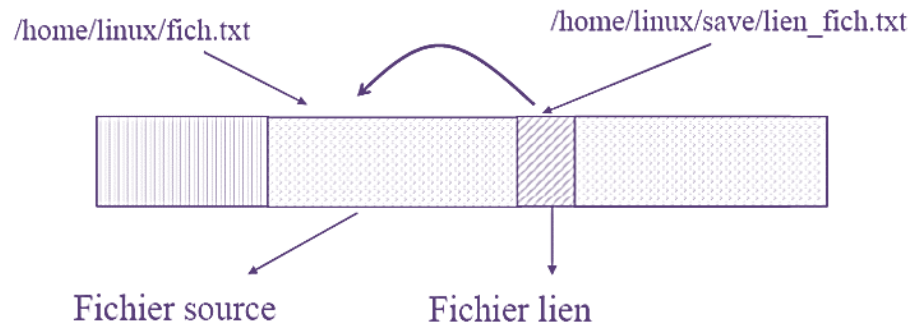
→ Les liens physiques

- Associe deux ou plusieurs fichiers à un même espace disque
- Les deux fichiers restant indépendants.
- Le fichier sera supprimé seulement si tous ces liens sont supprimés
- Utile dans le cas des fichiers de données

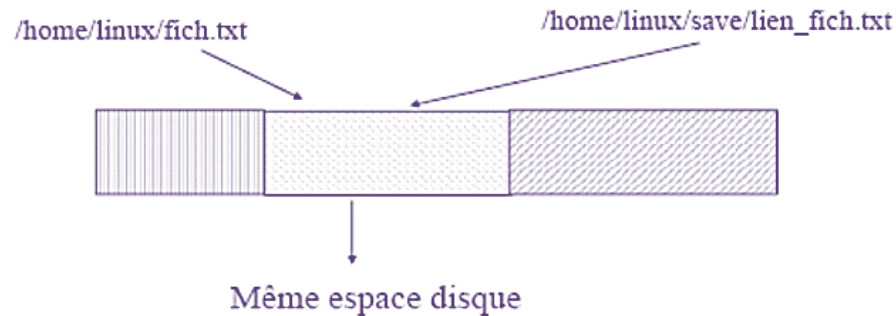
Manipulation de fichiers

→ Les liens

→ Les liens symboliques



→ Les liens physiques



Manipulation de fichiers

Syntaxe

ln [options] [fich_source] [fich_lien]

Description

- Crée un lien sur le fichier ou répertoire spécifié.
- Par défaut, la commande crée **un lien physique**
 - Si *fich_source* est effacé, l'inode continue à exister et est encore accessible au moins par le *fich_lien*.

Options

- **-s** : Création de **lien symbolique**
 - Si *fich_source* est supprimé, *fich_lien* référencera un fichier qui n'existe plus, alors le lien sera considéré comme "cassé"

Manipulation de fichiers

Exemple

→ Création de liens physique

```
$ ls -i
```

```
65329 fichier
```

```
65350 fichier2
```

```
$ ln fichier fichier3
```

```
$ ls -i
```

```
65329 fichier
```

```
65350 fichier2
```

```
65329 fichier3
```

```
$ rm fichier
```

```
$ ls -i
```

```
65350 fichier2
```

```
65329 fichier3
```

```
$ ln fichier3 fichier4
```

```
65350 fichier2
```

```
65329 fichier3
```

```
65329 fichier4
```

Manipulation de fichiers

Exemple

→ Création de liens symbolique

```
$ ls -i
```

```
65350 fichier2
```

```
65329 fichier3
```

```
65329 fichier4
```

```
$ ln -s fichier3 fichier5
```

```
$ ls -i
```

```
65350 fichier2
```

```
65329 fichier3
```

```
65329 fichier4
```

```
65378 fichier5 □ fichier3
```

```
$ rm fichier3
```

```
$ ls -i
```

```
65350 fichier2
```

```
65329 fichier4
```

```
65378 fichier5 □ fichier3
```

```
$ cat fichier5
```

```
cat :fichier5 :No such file or directory
```


Manipulation de fichiers

→ Autres commandes utiles:

- \$ **touch** [*nom_fich*] : permet de créer un nouveau fichier vide
- \$ **nl** [*nom_fich*] : permet d'afficher un fichier texte en numérotant les lignes. Par défaut, les lignes vides ne sont pas numérotées.
- \$ **cat** [*nom_fich*] : affiche le contenu du fichier à l'écran en ASCII
- \$ **more** [*nom_fich*] : affiche progressivement un fichier à l'écran (page par page):
Entrer = descend d'une ligne, Espace = descend d'une page, q = quitte
- \$ **diff** [*nom_fich1*] [*nom_fich2*] : affiche les différences entre *fich1* et *fich2*
- \$ **locate** [*logiciel*] : indique les fichiers associés au logiciel
- \$ **type** [*exécutable*] : indique le chemin d'un exécutable
- \$ **du -h** [*nom_rép*] : donne la taille du contenu du répertoire
- \$ **df -h** : donne l'espace disponible par partition, **-h** indique d'utiliser les unités usuelles

Contrôler l'accès aux fichiers

Syntaxe

chmod *[droit] [nom_fich]*

Description

- Modifie les droits et permissions accordés par le propriétaire aux différents utilisateurs du système.
- **Il existe plusieurs notations des droits:**
 1. La notation alphanumérique:(ugoa) (+/-) (rwx)
 - **u** : user (utilisateur), **g** :group, **o** :other (autre), **a**:All (tous les utilisateurs)
 - **r** : read (lecture) , **w** : write (écriture), **x** : execute (exécution)
 - **+** :est utilisé pour attribuer de nouveaux droits de fichier à une catégorie d'utilisateurs
 - **-** :Supprime un droit de fichier à une catégorie d'utilisateurs.
 2. La notation octale :
 - **4** :Read (Lire), **2** :Write (Ecrire), **1**:Execute (Exécuter), **0**: Aucun droit

Contrôler l'accès aux fichiers

Position binaire	Valeur octale	Les droits	Commentaire
000	0	- - -	Aucun droits
001	1	- - x	Executable
010	2	- w -	Ecriture
011	3	- w x	Ecrire et executer
100	4	r - -	Lire
101	5	r - x	Lire et executer
110	6	r w -	Lire et ecrire
111	7	r w x	Lire ecrire et executer

Exemples

La notation alphanumérique :

\$ **chmod a+rwX EST.txt**

La notation octale :

\$ **chmod 751 ESTBM.txt**

Utilisateur	U	G	O
Droits d'accès	r w x	r - x	- - x
Position binaire	1 1 1	1 0 1	0 0 1
Octale	7	5	1

Contrôler l'accès aux fichiers

Syntaxe

umask [*valeur*]

Description

- Permet de définir des droits d'accès par défaut pour l'ensemble des fichiers et des répertoires que vous créez. c'est à dire **les permissions attribués à un fichier ou un répertoire lors de sa création.**
- Comme la commande *chmod*, *umask* utilise un code numérique pour représenter les droits d'accès absolus aux fichiers. Toutefois, la méthode de calcul du code de la commande *umask* est différente de celle de la commande *chmod*.

Remarque! Par défaut:

- Tous les **fichiers** que vous créez ont les droits d'accès (en lecture, écriture, mais non en exécution) suivants :

rw-rw-rw- (mode **666**)

- Tous les **répertoires** créés ont les droits d'accès (en lecture, écriture et exécution) suivants
rw-rw-rwx (mode **777**)

Contrôler l'accès aux fichiers

exemple

\$ umask 022

→ Comme pour le code numérique de la commande **chmod**, les trois chiffres à utiliser avec la commande **umask** sont les suivants :

- Le premier chiffre contrôle les droits d'accès de l'utilisateur propriétaire.
- Le deuxième chiffre contrôle les droits d'accès d'un groupe d'utilisateurs.
- Le troisième chiffre contrôle les droits d'accès des autres utilisateurs.

→ Pour déterminer la valeur à utiliser pour la commande **umask**, vous devez soustraire la valeur des droits d'accès souhaités des droits d'accès par défaut en cours affectés aux fichiers. Le résultat de l'opération représente la valeur à utiliser pour la commande **umask**.

Par défaut :

Fichier : 666 : rw-rw-rw , **Répertoire:** 777: rwxrwxrwx

\$ umask 022

- **Création d'un fichier :** $666 - 022 = 644$: rw- r-- r--
- **Création d'un répertoire:** $777 - 022 = 755$: rwx r-x r-x

Contrôler l'accès aux fichiers

umask Value Octal (xyz)	Default File Permissions	666 - xyz	Default Directory Permissions	777 - xyz
000	rW-rW-rW	666	rWxrWxrWx	777
002	rW-rW-r--	664	rWxrWxr-X	775
022	rW-r--r--	644	rWxr-Xr-X	755
026	rW-r-----	640	rWxr-X--X	751
046	rW--W----	620	rWx-WX--X	731
062	rW----r--	604	rWx--Xr-X	715
066	rW-----	600	rWx--X--X	711
222	r--r--r--	444	r-Xr-Xr-X	555
600	---rW-rW-	066	--XrWxrWx	177
666	-----	000	--X--X--X	111
777	-----	000	-----	000

Contrôler l'accès aux fichiers

→ Autres commandes:

- **chown** *[options] [username] :[group-name] [nom_fich]* : Modifier le propriétaire et le groupe d'un fichier
- **chgrp** *[options] [group-name] [nom_fich]* : Modifier le groupe d'un fichier.

→ Exemples

\$ **chown** estbm:group2 EST.txt

\$ **chgrp** group3 EST.txt

Remarquez bien les ":" entre **estbm** et **group2**, « estbm » correspond au nom du propriétaire et « group2 » correspond au nom du groupe.

Gestion des utilisateurs et des groupes

Sous Linux, un utilisateur est une personne physique ou virtuelle possédant des droits d'accès au système.

- Un groupe est, aussi pour Linux, un ensemble d'utilisateurs qui partagent les mêmes fichiers et répertoires. Nous verrons que les fichiers accordent des droits d'accès réglables à ces groupes.
- Chaque utilisateur doit faire partie au moins d'un groupe, son ***groupe primaire***. Celui-ci est défini au moment de la création du compte, et *par défaut*, l'utilisateur appartient à un nouveau groupe créé, portant son nom.
- Chaque utilisateur est identifié par un numéro (**UID**).
- Les utilisateurs peuvent se regrouper dans des groupes identifiés par (**GID**).
- Le super-utilisateur **root** est l'administrateur du système. Il possède tous les droits sur le système, les fichiers et les utilisateurs, les autres ne peuvent pas modifier le système.

Gestion des utilisateurs et des groupes

- Si l'on veut devenir **root** temporairement pour une commande, utiliser sudo.

\$ **sudo nom_commande**

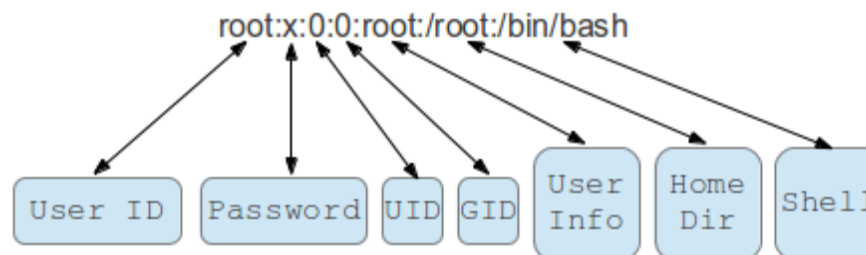
- Tout ce qui concerne la gestion et l'authentification des utilisateurs est inscrit dans un seul fichier **/etc/passwd**
- La gestion des groupes est assurée par **/etc/group**
- Les mots de passe cryptés sont maintenant placés dans **/etc/shadow**, par sécurité lisible seulement par root.

Gestion des utilisateurs et des groupes

→ Structure de */etc/passwd*

Ce fichier comprend 7 champs, séparés par le symbole :

- Nom de connexion
- Ancienne place du mot de passe crypté
- Numéro d'utilisateur **UID**, sa valeur est le véritable identifiant pour le système Linux; l'uid de root est 0, le système attribut conventionnellement un uid à partir de 500 aux comptes créés.
- Numéro de groupe **GID**, dans lequel se trouve l'utilisateur par défaut; le gid de root est 0, des groupes d'utilisateurs au delà de 500
- Nom complet, il peut être suivi d'une liste de renseignements personnels
- Rép. personnel (c'est également le rép. de connexion)
- Shell, interpréteur de commandes (par défaut /bin/bash)

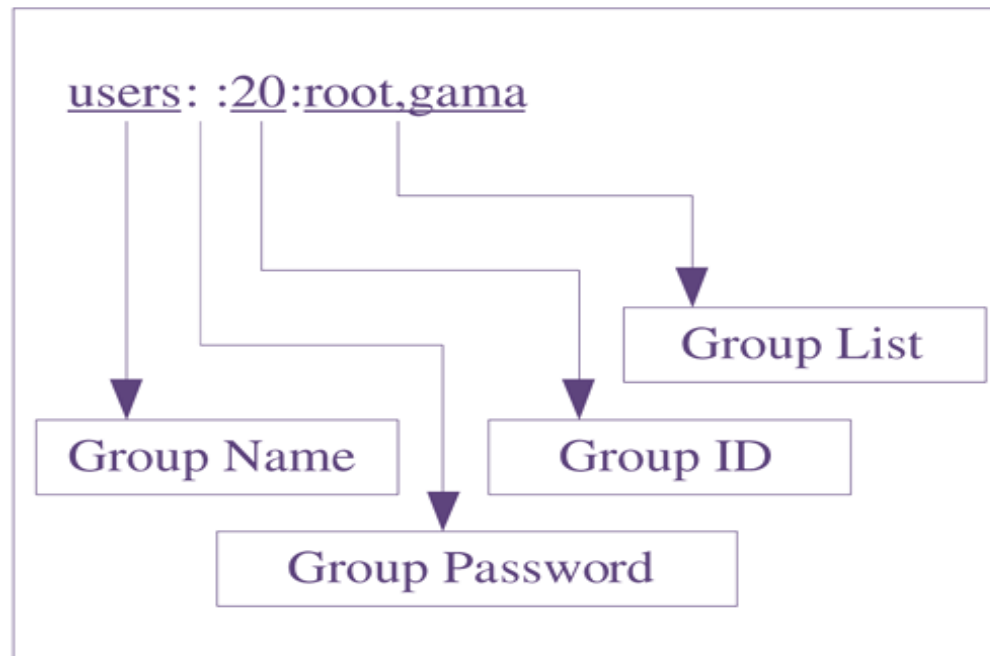


Gestion des utilisateurs et des groupes

→ Structure de */etc/group*

Ce fichier comprend 4 champs, séparés par le symbole « : »

- x pour remplacer un mot de passe non attribué maintenant
- Numéro de groupe, c-à-d l'identifiant **GID**
- La liste des membres du groupe



Gestion des utilisateurs et des groupes

→ Les principales commandes

→ Ajouter un utilisateur au système

\$ **useradd** *[options] [nom_utilisateur]*

Options

- **-u uid** :pour fixer l'identifiant **UID**
- **-g gid** :indiquer le **GID** de l'utilisateur
- **-G liste** :fixe l'appartenance de l'utilisateur à une liste de groupes secondaires (séparateur , sans espace)
- **-s shell** :indiquer le shell de connexion (/bin/bash, /bin/false, ...), par défaut bash
- **-d rep. Personnel** :indiquer le répertoire home à créer, par défaut dans le répertoire /home
- **-e date-expiration** :fixe la date d'expiration du compte (format MM/JJ/AA)
- **-m** :pour créer le répertoire personnel, par défaut /home/utilisateur.

Exemple

\$ **useradd** ESTBM **-u** 1200 **-g** 520 **-G** groupes **-s** /bin/bash

Gestion des utilisateurs et des groupes

→ Les principales commandes

→ Modifier le compte d'un utilisateur

\$ **usermod** *[options] [nom_utilisateur]*

- Les options sont les mêmes que *useradd*

→ Ajouter un utilisateur à un groupe

\$ **adduser** *[nom_utilisateur] [nom_groupe]*

→ Suppression d'un utilisateur

\$ **userdel** *[-r] [nom_utilisateur]*

- L'option **-r** supprime aussi le rép. personnel et les fichiers de l'utilisateur
- La commande supprime toute trace de l'utilisateur dans le fichier de configuration : **/etc/passwd** y compris dans les groupes d'utilisateurs.

→ Ajouter / changer le mot de passe d'un utilisateur

\$ **passwd** *[nom_utilisateur]*

Gestion des utilisateurs et des groupes

→ Les principales commandes

→ Afficher la liste des utilisateurs :

\$ **compgen** -u ou \$ **cat** /etc/passwd

→ Afficher la liste des groupes

\$ **compgen** -g ou \$ **cat** /etc/group

→ Lister tous les groupes (primaire et secondaires) d'un utilisateur :

\$ **groups** *[nom_utilisateur]*

→ Ajouter un groupe au système

\$ **groupadd** *[nom_groupe]*

→ Supprimer un groupe

\$ **groupdel** *[nom_groupe]*

- Le groupe est supprimé du fichier /etc/group.

Gestion des utilisateurs et des groupes

→ Les principales commandes

→ Afficher la liste des utilisateurs qui sont connectés au système:

\$ **who** *[options]*

- Option *[-b]* :afficher le dernier démarrage du système

→ affiche le nom d'utilisateur actuel qui est connecté:

\$ **whoami**

→ Afficher l'utilisateur qui est connecté et ce qu'il fait:

\$ **w**

Gestion des processus

Un système d'exploitation se compose de processus. Ces derniers, sont exécutés dans un ordre bien précis et observent des liens de parenté entre eux.

- On distingue **deux catégories** de processus, ceux axés sur l'environnement utilisateur et ceux sur l'environnement matériel.
- Lorsqu'un programme s'exécute, le système va créer un processus en plaçant les données et le code du programme en mémoire et en créant **une pile d'exécution**.
- Un processus est donc une instance d'un programme auquel est associé un environnement processeur (compteur ordinal, registres, etc.) et un environnement mémoire.

→ Chaque processus dispose :

- D'un ***PID*** : *Process IDentifiant*, **identifiant unique de processus** ;
- D'un ***PPID*** : *Parent Process IDentifiant*, **identifiant unique de processus parent**.

→ Visualisation des processus

Syntaxe

ps [*options*]

Description

- affiche l'état des processus en cours.
- Sans option précisée, la commande **ps** n'affiche que les processus exécutés à partir du terminal courant.

Options

- **-e** :Affiche tous les processus.
- **-f** :Affiche des informations supplémentaires.
- **-aux** :donne des informations , qui l'a lancé, quelles ressources il utilise, son état, son nom, son PID (un numéro unique identifiant chacun).

Gestion des processus

→ La commande de gestion des processus

Syntaxe

kill [-signal] [PID/processus_name]

Description

- envoie un signal d'arrêt à un processus

Code	Signal	Description
2	SIGINT	Interruption du processus (kbd:[CTRL+D])
9	SIGKILL	Terminaison immédiate du processus
15	SIGTERM	Terminaison propre du processus
18	SIGCONT	Reprise du processus
19	SIGSTOP	Suspension du processus

Exemple

\$ **kill -9 1664**

Gestion des processus

→ Autre commandes:

- \$ **killall** : permet de fermer un ensemble de processus.
- \$ **pstree** : affiche l'arbre hiérarchique des processus, permettant de trouver le processus parent d'un processus donné.
- \$ **at** options time permet de lancer une commande à un moment précis.
- \$ **chkconfig** pour gérer les services et le niveau de démarrage

Remarques

- Un processus qui ne répond plus est représenté par la lettre 'Z' (zombie).
- Un processus normalement inactif est représenté par la lettre 'S' (sommeil).
- En appuyant simultanément sur les touches kbd:[CTRL+Z], le processus synchrone est temporairement suspendu.