# SYSTÈMES D'EXPLOITATION

## Partie 2: Les Systèmes Unix

**Dr. Jarray Ridha**



**Année Universitaire 2023/2024**

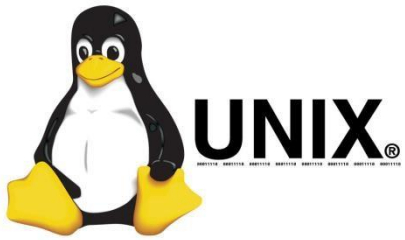
# Plan

1. Caractéristiques d'Unix
2. Composantes du système Unix
3. Système de fichiers (FHS)
4. Interpréteur de commandes Shell
5. Commandes de base

# Caractéristiques d'Unix

➢ **Unix** est un système d'exploitation **multi-utilisateurs** et **multitâches**.

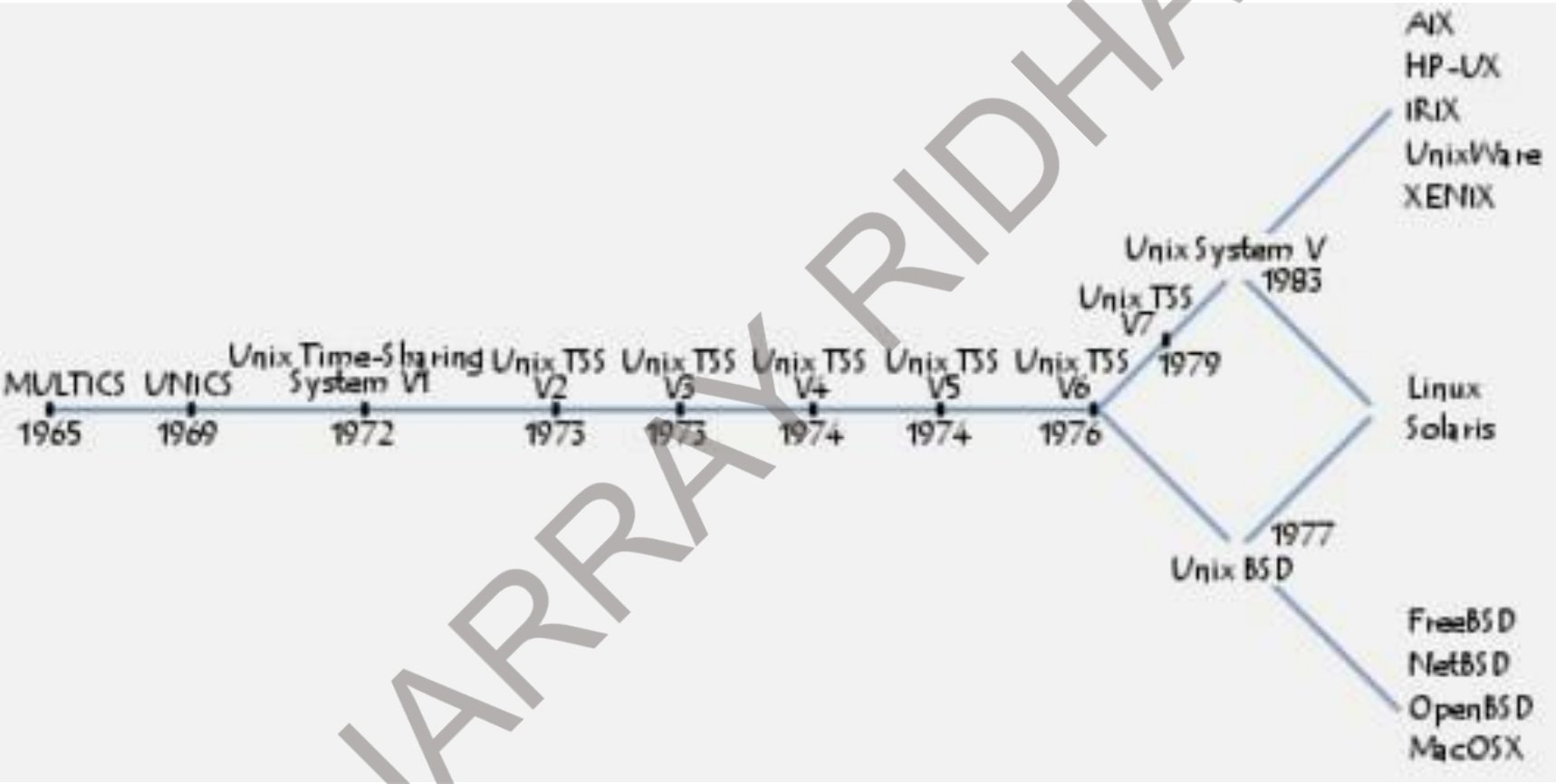
- **Multi-utilisateurs :**
  - Unix comporte des **mécanismes d'identification** et de **protection** permettant d'**éviter** toute **interférence** entre utilisateurs.
  - Deux types d'utilisateurs :
    - **Utilisateurs normaux**,
    - **Super-utilisateur (root)** : gère tout le système.
- **Multi-tâches :**
  - Unix permet à **plusieurs programmes** d'être en cours d'exécution en **même temps** sur une **même machine**.



# Caractéristiques d'Unix

- **Portabilité** (la plupart des programmes sont écrits en **C,** permettant ainsi une portabilité sur la plupart des plates-formes matérielles )
- **Système de fichiers hiérarchisé** (Organisation arborescente)
- **Interactivité** ( Il est possible de dialoguer avec l'ordinateur (**Shell**))
- **Une vision simplifiée des entrées-sorties** (les périphériques sont représentés par des **fichiers**, ce qui rend le système indépendant du matériel et en assure la portabilité).

# Historique

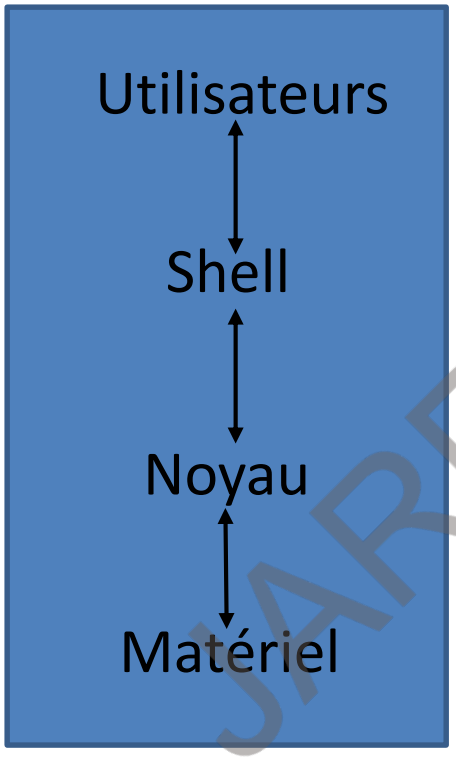





**Historique de l'apparition des principaux systèmes de type Unix**

# Architecture d'Unix

Schéma d'exploitation de la machine




- **Shell** , interpréteur de commandes Unix (vérifie, interprète les commandes, exécute et renvoie les réponses). Le Shell envoie des appels au noyau en fonction des requêtes des utilisateurs.
- **Noyau,** couche logicielle la plus interne du SE Unix dédiée à la gestion des composants matériels: processeur, mémoire, périphérique.
- Autour du noyau gravite un certain nombre d'utilitaires

# Systèmes Linux

## Principales distributions Linux

- Il existe différentes versions de Linux.
- Les distributions se différencient par le choix du noyau et le choix des différents utilitaires disponibles.

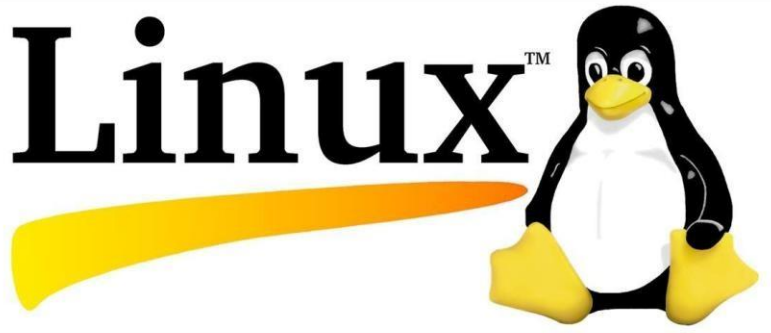
  - un ensemble de logiciels de base issus du projet GNU

  - une méthode pour installer et désinstaller facilement ces programmes

  - etc….

**Exemples :**

- CentOS,
- RedHat,
- Fedora,
- Mandriva,
- Debian,
- Ubuntu.



# Composantes du système Unix

## Composantes du système Unix

- **Le noyau (kernel)**.
- **Shell** : Il existe plusieurs Shells pour le système Unix dont :
  - **Le shell de Bourne : sh** (le shell standard d'Unix),
  - **Le shell de Korn : ksh** (englobe celui de Bourne),
  - **Le C shell : csh** (le shell d'Unix BSD).
  - **Le Bourne-Again shell : Bash** (le shell du projet GNU).


Il existe plusieurs commandes communes à ces interpréteurs.

# Identification et compte

## Identification et compte

- Linux possède un mécanisme d'identification connu sous le nom de login.
- Pour utiliser un système Linux sur une machine, il faut avoir un compte sur cette machine.
- Pour se connecter sur une machine il faut rentrer au clavier:
  - son nom d'utilisateur: **login**
  - son mot de passe: **password**
- Le système vérifie la correspondance entre login et mot de passe
  - si échec, il refuse l'accès
  - si correct, il lance la procédure de login
- L' utilisateur est alors placé dans son répertoire d'accueil (exp:/home/Etud)

# Les commandes GNU

## Accès à un interpréteur des commandes

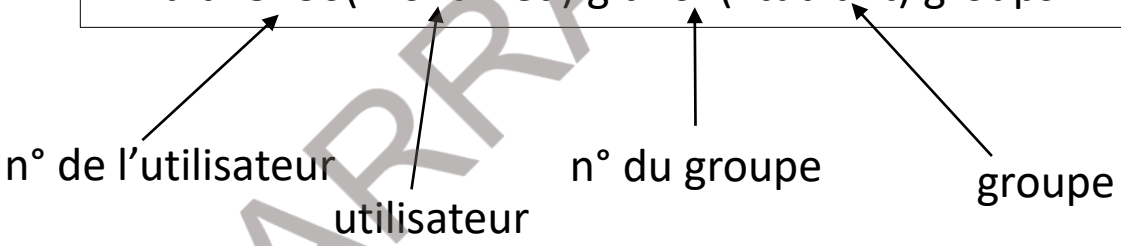
- ✓ Le shell gère la communication avec SE par l'intermédiaire d'un langage de commandes.
- ✓ Comparable à l'invite de commandes sous Windows
- ✓ Exécuter les commandes du système d'exploitation
- ✓ Un meilleur contrôle sur les applications
- ✓ Utiliser certains outils non graphiques
- ✓ Pour l'ouverture d'un interpréteur (terminal: xterm)
- ✓ Pour connaître le shell utilisé, tapez: **%echo $SHELL**
- ✓ La liste des shells autorisés: **/etc/shells**

# Les commandes GNU

## Identité sous Linux

- ✓ La commande **whoami** donne l'identité de l'utilisateur connecté.
- ✓ L'utilisateur appartient à un ou plusieurs groupes
- ✓ La commande id donne l'identité et le groupe de l'utilisateur connecté

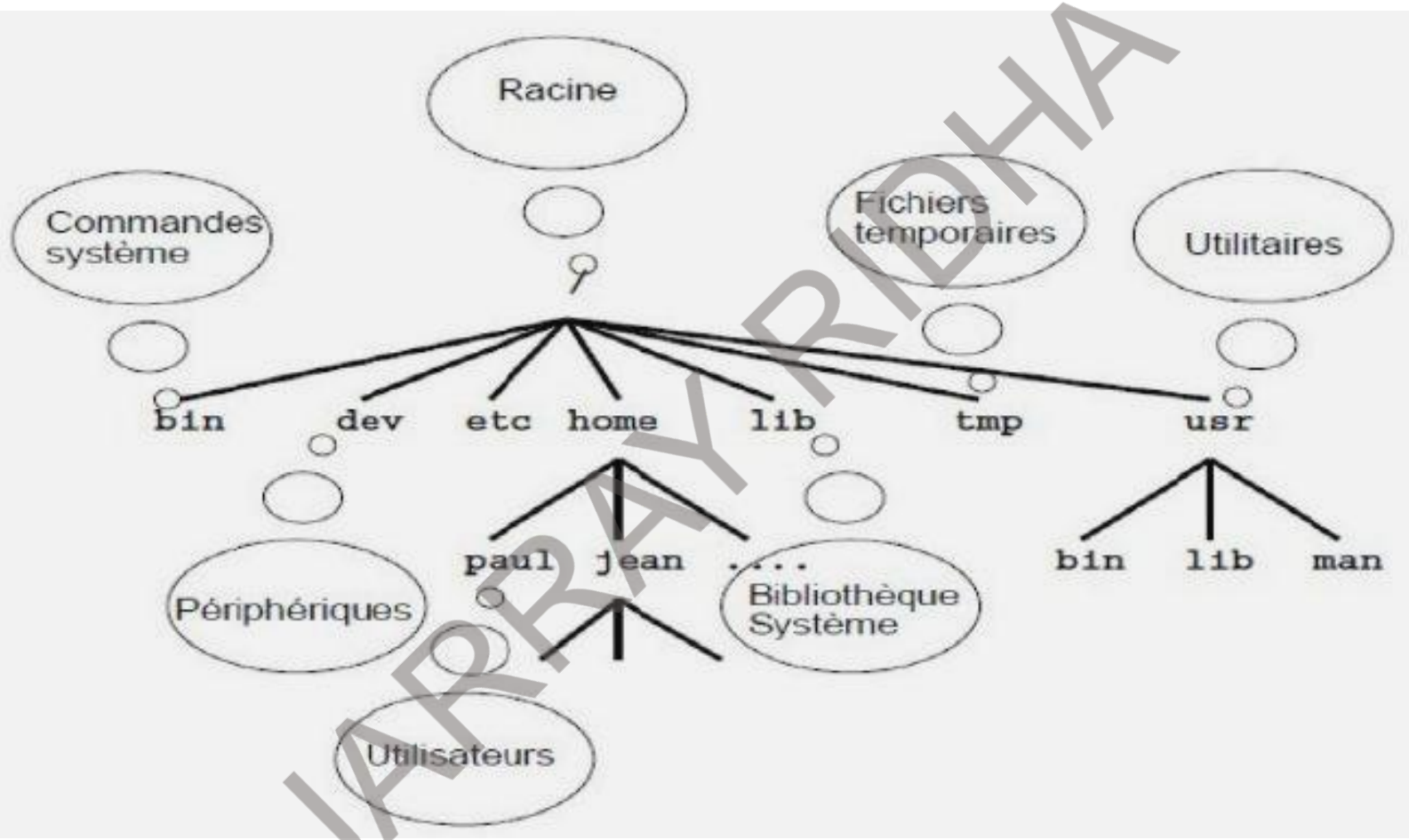
```
%id
uid=5230(Mohamed) gid=64(Etudiant) groups
```



# Organisation des fichiers

- Dans les systèmes Linux, **tout est fichier**, donc, contrairement aux systèmes Windows un fichier Linux peut être :
  - un fichier
  - un périphérique
  - une partition
  - un programme en cours d'exécution
  - un répertoire …
- Pour structurer ces fichier, Linux utilise le standard ***FHS (Filesystem Hierarchy Standard)*** pour définir son **arborescence**.
- Ce standard propose une structure de répertoires dont chacun possède un rôle spécifique définit dans le **système de fichier**.

# Arborescence des répertoires (des fichiers)






**Aperçu des répertoires dans Unix/Linux**

# Principaux répertoires

- **/bin** : contient les commandes de base.
- **/dev** : contient les fichiers représentant les points d'accès aux périphériques de votre système.
- **/etc** : contient la plupart des fichiers de configuration.
- **/home** : contient les répertoires personnels des utilisateurs (l'utilisateur Paul a pour répertoire /home/paul).
- **/lib** : contient les librairies partagées ou des liens vers ces librairies.
- **/tmp** : contient les fichiers temporaires.
- **/usr** : indique les logiciels installés avec le système.

**Remarque**

- **/** : désigne le dossier racine.
- **..** : désigne le dossier parent.
- **.** : désigne le dossier lui-même.

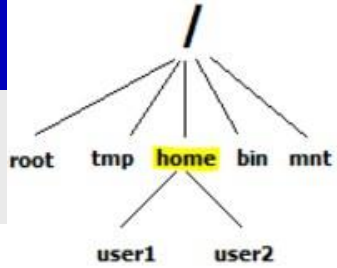
# Chemin absolu, chemin relatif

- L'emplacement de chaque **ressource** (fichier ou répertoire) dans le système de fichiers est appelé son **chemin**. Dans un chemin Linux, le séparateur entre deux répertoires est le caractère " **/** ".
- On distingue **deux** types de **chemins :**
  - **Un chemin absolu** identifie une ressource en commençant à la **racine** de l'arborescence, avec le caractère " **/** ". Un chemin absolu **ne dépend pas** du **répertoire courant** et est donc valide partout.

    **Exemple:** **/**home/user1/hello.txt ; **/**etc/apache/httpd.conf
  - **Un chemin relatif** identifie une ressource à partir du répertoire courant. Il **dépend** donc du **répertoire courant** et n'est pas valide partout.

    **Exemple:** **..**/marc/adresses.txt ; **.**/cours/sys.pdf

# Répertoire personnel




- Sous Linux, chaque utilisateur (sauf root) dispose d'un **répertoire personnel** à son nom situé dans **/home**. Par exemple, le répertoire personnel de l'utilisateur nidhal est **/home/nidhal**.
- A la connexion, l'utilisateur est automatiquement positionné dans son répertoire de connexion.
- Le **chemin absolu** du répertoire personnel peut s'écrire de manière abrégée avec le caractère **~** (*tilde*)
  - Répertoire personnel de l'**usage courant :**

    **Exemple:** **/home/nidhal/music => ~/music**
    (Répertoire personnel pour l'utilisateur courant nidhal)
  - Répertoire personnel d'un **autre usager :**

    **Exemple:** **/home/ali/music => ~ali/music**

# Caractères spéciaux pour les noms de fichiers et de répertoires

- **.** : désigne le répertoire courant
- **..** : désigne le répertoire père
- **~** : désigne le répertoire personnel de l'utilisateur (/home/ nom-utilisateur)
- **?** : remplace un caractère quelconque.
- ***** : remplace une chaîne de caractères quelconque (y compris une chaîne vide).
- **[...]** : désigne un caractère quelconque dans l'ensemble défini entre crochets ( une liste de caractères [adefgw] ou un intervalle [0-9] [a-z] )
- **[!...]** : désigne un caractère quelconque hors de l'ensemble.

**Exemple:**

***.c** → noms suffixés .c
***.[csp]** → noms suffixés .c, .s ou .p
**[!0-9]*** → noms ne commençant pas par un chiffre

# Les métacaractères

- Il existe plusieurs méthodes pour enchaîner des commandes sur une même ligne:
  - **;** sépare les commandes
  - **&** lance en tâche de fond
  - **||** séparateur conditionnel, si échoue
  - **&&** séparateur conditionnel, si succès
- **Cmd1; cmd2; …; cmdN** : Exécution séquentielle
- **Cmd1 || cmd2 || … || cmdN** : Exécution sous condition d'erreur si cmd1 ne se termine pas correctement, alors cmd2 est exécuté, et ainsi de suite
- **Cmd1 && cmd2 && … && cmdN** : Exécution sous condition de réussite si cmd1 s'est bien déroulée, alors cmd2 sera exécutée, et ainsi de suite
- **Cmd1&** : le système lance cmd1 et redonne immédiatement la main à l'utilisateur pour d'autres travaux

# Ligne de commande

- Unix fonctionne en mode **ligne de commandes** et non en mode graphique.
- Une commande est un programme.
- Pour exécuter une commande, il faut taper son nom, éventuellement suivi d'options et d'arguments, suivi de la touche Entrée.

**Syntaxe d'une commande :**

**nom_commande** [liste_options] [liste_arguments]

**Exemple :** **ls –l /bin**

- Lors de l'appui sur la touche Entrée, le shell analyse la ligne de commande et l'interprète.

# Commande pour consulter le manuel

## Commande man

- La commande man permet de consulter le manuel et visualiser à l'écran des informations concernant le mot-clé spécifié.

**Syntaxe :**

**man [n] keyword**

- Le manuel est divisé en huit sections allant de 1 à 8. Le numéro de la section dans laquelle nous voulons effectuer la recherche est indiqué grâce au paramètre **n**.
- **Exemple:** :
  - **man** 3 intro (pour lire l'introduction de la section 3 )
  - **man** ls (afficher le manuel de la commande ls )

# Commande pour afficher une chaîne de caractères

## Commande echo

- La commande **echo** permet d'afficher la chaîne passée en paramètre.

**Syntaxe :**

**echo chaine**

**Exemple**: **echo** Hello world (afficher la chaîne sur la sortie standard)

- La commande **echo** permet aussi d'afficher le contenu d'une variable.

**Syntaxe :**

**echo $NomVariable**

**Exemple:** **var**="commande echo"
echo $**var**

# Commandes pour manipuler des répertoires

## Commande ls

La commande **ls** permet de lister le contenu d'un répertoire.

**Syntaxe:**

**ls [options] nom_répertoire**

**Options :**

- **-l :** permet d'obtenir l'ensemble des informations relatives à chaque fichier du répertoire :
  - type de fichier : **"-" (fichier ordinaire), "d" (répertoire), "b ou c "(fichiers spéciaux)**
  - droits d'accès
  - nom du propriétaire
  - nombre de liens
  - taille…
- **-a:** permet l'affichage des fichiers et répertoires cachés, qui commencent par un **.** (point)

# Commandes pour manipuler des répertoires

## Commande cd

- La commande **cd** permet de changer de répertoire courant pour celui spécifié par le **chemin**.

**Syntaxe :**

**cd chemin**

- **Exemple:**
  - **cd** :permet de revenir au répertoire personnel /home/utilisateur (identique à cd ~)
  - **cd -** : permet de revenir au répertoire précédent
  - **cd ..** : permet de remonter d'un niveau dans l'arborescence (répertoire parent)
  - **cd /** : permet de revenir à la racine de l'arborescence.

# Commandes pour manipuler des répertoires

## Commande pwd

La commande **pwd** (*print working directory*) affiche le chemin du répertoire courant.

## Commande mkdir

La commande **mkdir** (*make directory*) permet de créer un répertoire

**Syntaxe :** **mkdir répertoire**

## Commande rmdir

La commande **rmdir** permet de supprimer un répertoire vide.

**Syntaxe :** **rmdir répertoire**

# Commandes pour manipuler des fichiers

## Commande cat

La commande **cat** permet de visualiser le contenu du (ou des) fichier(s).

**Syntaxe :** **cat nom_fichier**

**Exemple:**

- **cat fich** : affiche le contenu du fichier (sur la sortie standard).
- **cat fich1 fich2** : concatène et affiche (sur la sortie standard) le contenu des fichiers.

## Commande more

La commande **more** permet d'afficher le fichier une page à la fois, passer à la page suivante en utilisant la barre d'espace (sans retour en arrière)

**Syntaxe :** **more nom_fichier**

# Commandes pour manipuler des fichiers

## Commande grep

La commande **grep** permet de chercher une chaîne de caractère dans un ou plusieurs fichiers.

**Syntaxe :** **grep [options] expression fichiers**

- **sans option** : recherche dans les fichiers les lignes contenant l'expression.
- **-i** (sans tenir compte des majuscules/minuscules)
- **-c** (compte le nombre de lignes contenant l'expression spécifiée)
- **-v** (inverse la recherche, affiche les lignes ne contenant pas l'expression spécifiée)

# Commandes pour manipuler des fichiers

## Commande find

La commande **find** permet de retrouver des fichiers à partir de certains critère

**Syntaxe :** **find** <répertoire de recherche> <**critères de recherche**>

- **Les critères de recherche:**

  **-name** : nom du fichier
  **-user** : propriétaire du fichier
  **-type** : type (**d**=répertoire, **c**=caractère, **f**=fichier normal)
  **-size** : taille du fichier en nombre de blocs (1 bloc=512octets)

- **Exemple:**

  - Pour afficher tous les fichiers se terminant par "**.c**" :
    **find . -name "*.c"**
  - Pour afficher tous les répertoires dont le nom se termine par "**s**" :
    **find . -type d -name "*s"**
  - Pour afficher tous les fichiers ayant une taille de **10** blocs :
    **find . -size 10**

# Commandes pour manipuler des fichiers

## Commande cp

La commande **cp** permet de copier un fichier source dans une destination.

**Syntaxe :**

**cp source destination**

- Si la **destination** est un **fichier** qui n'existe pas, alors il sera créé. Sinon son contenu sera écrasé sans avertissement.
- Si la **destination** est un **répertoire**, alors la source peut être une liste de fichiers.

# Commandes pour manipuler des fichiers

## Commande mv

La commande **mv** permet de renommer ou déplacer un fichier source dans une destination.

**Syntaxe :**

**mv source destination**

- Si la **destination** est un **fichier**, alors **mv** a pour action de renommer le fichier source en destination.
- Si la **destination** est un **répertoire**, alors **mv** déplace le fichier source dans ce répertoire.

# Commandes pour manipuler des fichiers

## Commande rm

La commande **rm** permet de supprimer un ou plusieurs fichiers.

**Syntaxe :**

**rm fichiers**

- **Options:**
  - **-i** : mode interactif, demande une confirmation sur chaque fichier.
  - **-f** : force la suppression du fichier.
  - **-r** : récursif, permet d'effacer un répertoire et son contenu.
- **Exemple:**
  - Supprimer un répertoire entier et tout ce qu'il inclut des fichiers et des sous-répertoires: **rm -rf nom_répertoire**
  - Supprimer tous les fichiers d'extension ".o" : **rm *.o**

# Protection des fichiers

## Commande chmod

L'accès aux fichiers est déterminé par trois bits de permission: **r w x (Read, Write, eXecute)** applicables à trois classes d'utilisateurs : **u g o** le propriétaire, le groupe et les autres (**Users**, **Group**, **Others**).

- **mode symbolique:**

**chmod <qui> <permission> <opération> <fichier>**

- **<qui> :** **u**: utilisateur **g**: groupe **o**:autres et **a** :tous
- **<permission> :** **+** : pour autoriser **-** : pour interdire
- **<opération> :** **r** : lecture **w** : écriture **x** : exécution

**Exemple:**

**chmod g+w montp.c** (les membres du groupe peuvent écrire dans le fichier "montp.c")

# Protection des fichiers

- **mode octal:**

**chmod <permission> <fichier>**

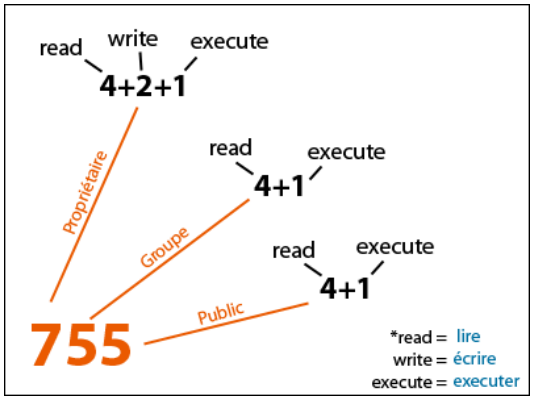
**<permission >** : **UGO** (**U**ser,**G**roup ,**O**thers : chiffre octal codant les bits **r w x)**

**U** : droits de propriétaire
**G** : droits du groupe
**O** : droits pour les autres
0 : aucun droit
1 : droit en exécution
2 : droit en écriture
3 : droit en écriture et en exécution
4 : droit en lecture
5 : droit en lecture et en exécution
6 : droit en lecture et en écriture
7 : tous les droits

**Exemple:**

**chmod 740 montp**

(rend le fichier "montp" accessible en lecture au groupe et inaccessible aux autres)



# Protection des fichiers

Manipulation des fichiers

**Opérations sur les fichiers:**

- **find :** recherche des fichiers ou répertoires.
- **grep :** recherche d'une chaîne de caractères dans un fichier.
- **head/tail :** affiche le début/la fin de fichier
- **ln :** créer un lien avec un fichier existant
- **sort :** trie les lignes d'un fichier
- **umask :** choix des permissions par défaut
- **wc :** compte le nombre de mots/lignes/caractères d'un fichier

# Commandes pour manipuler des fichiers

Manipulation des fichiers

**Opérations sur les fichiers:**

- **cp :** copie des fichiers.
- **mv :** déplacement des fichiers.
- **rm :** destruction des fichiers
- **cat :** visualisation et/ou concaténation des fichiers
- **more :** visualisation d'un fichier texte page par page
- **chmod :** change les droits d'un fichier/répertoire
- **chown :** change les propriétaire d'un fichier/répertoire
- **chgrp :** change le groupe propriétaire d'un fichier/répertoire

# Commandes pour manipuler des fichiers

## Commande head/tail

La commande **head** permet d'afficher les n premières lignes, alors que **tail** sert à afficher les dernières lignes d'un fichier. Si n n'est pas précisé, il prend la valeur 0.

**Syntaxe:**

```
head  -n  fichier
tail   -n|+n  -f  fichier
```

- **Options:**
  - **-n** : nombre de lignes à afficher depuis le début/la fin de fichier
  - **+n** : affichage à partir de la ligne numéro n.
  - **-f** : attente de nouvelles lignes (sortie par Crt-c).

# Commandes pour manipuler des fichiers

## Commande sort

La commande **sort** permet de trier des fichiers en arguments et affiche le résultat à l'écran.
Par défaut sort effectue un tri par ordre alphabétique; mais les options suivantes en modifient les critères.

**Syntaxe:**

**sort -ufnr -o fic fichier**

- **Options:**
  - **-u** : permet de n'afficher qu'une seule fois les lignes multiples.
  - **-f** : ne différencie pas les minuscules et majuscules.
  - **-n** : effectue un tri numérique.
  - **-r** : ordre décroissant.
  - **-o fic** : enregistre la sortie dans fic.

# Commandes pour manipuler des fichiers

## Commande ln

La commande **ln** sur Linux permet de créer des liens symboliques.
On peut créer des liens symboliques entre deux fichiers ou vers un répertoire.
C'est donc une commande importante à savoir utiliser pour gérer son système de fichiers.

**Syntaxe:**

```
ln -s fichier1 fichier2
```

- **Options:**
  - **-s** : permet de faire un lien symbolique.

# Commandes pour manipuler des fichiers

## Commande umask

La commande **umask** permet de définir les droits affectés par défaut aux fichiers lors de leur création.

**Syntaxe:**

**umask [???]**

- **Options:**
  - **???** : chaque ? représente une valeur entre 0 et 7 qui le complément à 7 des droits à affecter aux fichiers. Si l'on veut avoir des fichiers avec 751 (rwxr-x--x) comme droits, il faudra définir comme masque 026.

# Commandes pour manipuler des fichiers

## Commande wc

La commande **wc** permet de compter le nombre de lignes, mots, ou caractères d'un fichiers texte.

**Syntaxe:**

**wc [-lwc] fichiers**

- **Options:**
  - **-c** : - bytes, - chars: affiche uniquement le nombre de caractères.
  - **-w** : - words: affiche uniquement le nombre de mots.
  - **-l** : - lines: affiche uniquement le nombre de lignes(saut de lignes).

# Commandes pour manipuler des fichiers

## Commande chown/chgrp

La commande **chown** permet de changer le propriétaire spécifiés sur la ligne de commande.
La commande **chgrp** change le groupe des fichiers spécifiés sur la ligne de commande.

**Syntaxe:**

**chown [-R] [-h] utilisateurs nom […]**
**chgrp [-R] [-h] groupe nom […]**

- **Options:**
  - **-R** : récursif sur tous les fichiers et sous-répertoires contenus si nom un répertoire.
  - **-h** : traitement sur les liens symboliques.

* utilisateur représente soit le nom de l'utilisateur, soit son UID(user IDentification).

* groupe représente soit le nom du groupe, soit son GID(Group IDentification)

# Commandes pour manipuler des fichiers

## Commande tar

La commande **tar** permet d'archiver un ensemble de fichiers ou répertoires, ou d'extraire le contenu d'une archive.

**Syntaxe:**

**tar [options] [-h] fichiers_ou_repertoires**

- **Options:**
  - **-c,** -create : créer une nouvelle archive.
  - **-u,** -update : ajoute seulement les fichiers plus récents que ceux de l'archive.
  - **-x,** -extract, -get : extrait les fichiers contenus dans une archive.
  - **-f,** -file F: utilise le fichier d'archive spécifie. L'extension généralement employée pour créer un fichier d'archive est **.tar**.
  - **-z,** -gzip : compresse l'archive avec **.**gzip. L'extension est alors en général **.tgz**.
  - **-v,** -verbose : affichant des informations sur l'archive pendant l'archivage.
  - **-t,** -list : liste les fichiers contenus dans une archive.

# Commandes pour manipuler des fichiers

**Exemple :**

- **Pour créer une nouvelle archive :**

  tar –cvf nom_archive.tar répertoire

- **Pour afficher le contenu d'un archive :**

  tar –tvf nom_archive.tar

- **Pour extraire les fichiers archivés :**

  tar –xvf nom_archive.tar répertoire

Les fichiers sont créés à partir du répertoire courant

# Commandes pour manipuler des fichiers

## Commande compress/uncompress

La commande **compress** permet de compresser un ou plusieurs fichiers en remplaçant chacun par un fichier de même nom, mais avec une extension **.Z**, chaque fichier est compressé séparément.
La commande **uncompress** permet de décompresser les fichiers se terminent par **.Z**

**Syntaxe:**

**compress [options] fichiers_ou_repertoires**
**uncompress [options] fichiers_ou_repertoires**

- **Options:**
  - **-d** : décompresse au lieu de compresser, identique à **uncompress**.
  - **-f** : force la génération du fichiers de sortie.
  - **-r** : si un des fichiers spécifiés est un répertoire, compresse son contenu récursivement.

# Commandes pour manipuler des fichiers

## Commande gzip/gunzip

La commande **gzip** permet de compresser un ou plusieurs fichiers en remplaçant chacun par un fichier de même nom, mais avec une extension **.gz**.
La commande **uncompress** permet de décompresser les fichiers se terminent par **.gz** .

**Syntaxe:**

```
gzip   [options]  fichiers_ou_repertoires
gunzip   [options]  fichiers_ou_repertoires
```

- **Options:**
  - **-d** : décompresse au lieu de compresser, identique à **gunzip**.
  - **-f** : force la génération du fichiers de sortie.
  - **-r** : si un des fichiers spécifiés est un répertoire, compresse son contenu récursivement.
  - **-v** : affiche le nom et la réduction de taille en pourcentage de chaque fichier.