

# SYSTÈME D'EXPLOITATION LINUX

UNIVERSITÉ CADI AYYAD

ECOLE SUPÉRIEURE DE TECHNOLOGIE ESSAOUIRA

ANNÉE UNIVERSITAIRE : 2022-2023

MR. YOUSSEF TAOUIL

# Sommaire

- I. Présentation du système LINUX
- II. Gestion de fichiers sous LINUX
- III. Gestion des utilisateurs sous LINUX
- IV. Gestion de processus sous LINUX

# Système d'exploitation LINUX

- LINUX est le noyau d'un OS (Operating System) de type Unix, initié par Linus Torvalds en 1991.
- Unix a été conçu en 1969 aux Bell Labs (AT&T) aux USA, ensuite il a été réécrit en langage C puis porté sur de nombreuses architectures matérielles.
- UNIX est OS multi-utilisateur et multitâches, ce qui l'a rendu convenable pour le contrôle des serveurs et gestion informatique des grandes organismes.



Linus Torvalds

# Système d'exploitation LINUX

## Historique

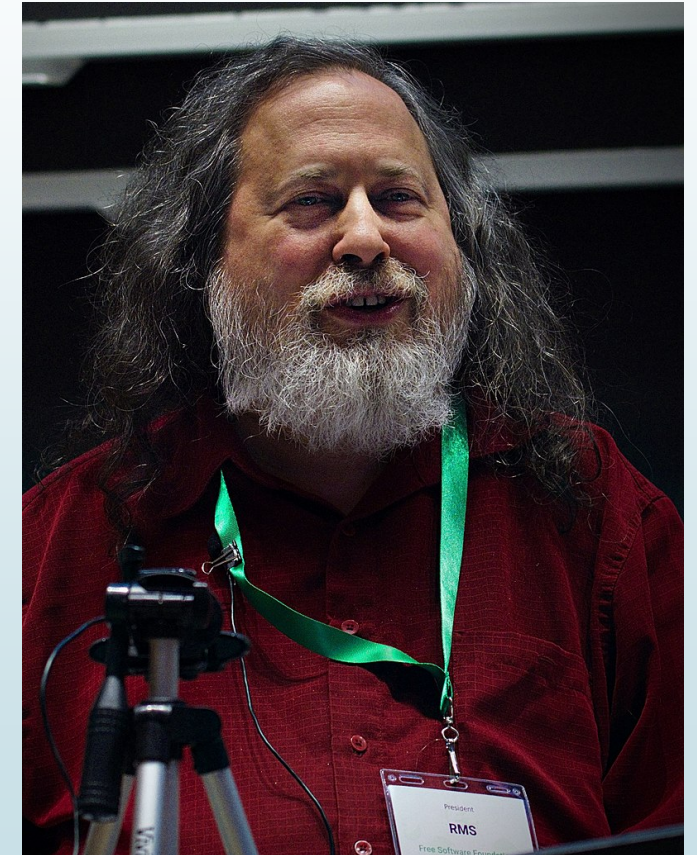
- 1965 : Ken Thompson créa MULTICS pour faire fonctionner un jeu de simulation du système solaire (lab. Bell de AT&T)
- 1969 : Ken Thompson et Denis Ritchie ont réécrit MULTICS pour une machine de 4Ko de mémoire  
UNiplexed Information Computing System (UNICS)
- 1970-1980: les entreprises s'approprient les codes et les rendent inaccessibles.



Ken Thompson & Denis Ritchie

# Système d'exploitation LINUX

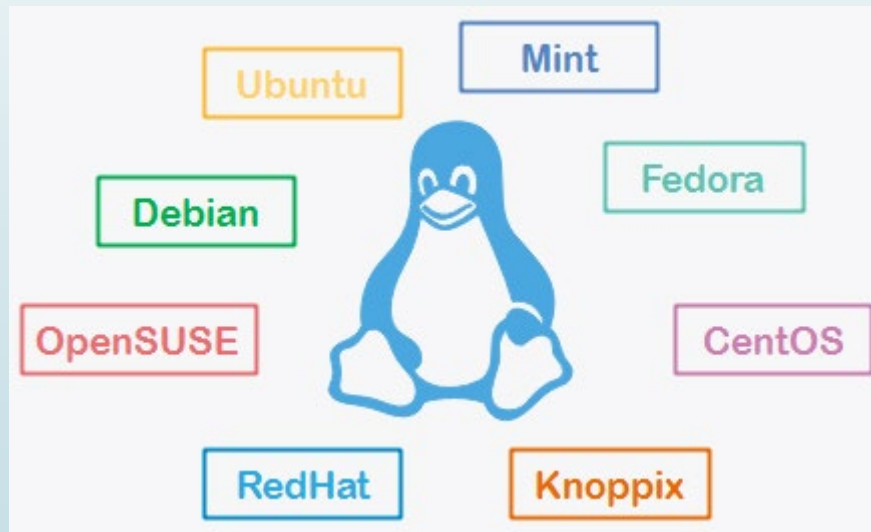
- 1982: Richard Stallmann crée la Free Software Foundation (FSF), pour perpétuer le mouvement des hackers des années 70, qui avaient pris l'habitude d'échanger les codes source de leurs programmes.
- 1983 : Richard Stallmann lance donc le projet GNU, dont l'objectif est de créer un système d'exploitation libre compatible Unix.
- 1984 : L'interface graphique X-Window a été créée et incluse aux systèmes UNIX.
- 1991 : GNU/LINUX est né à base de UNIX. **LINUX est par conséquent open source.**



Richard Stallmann

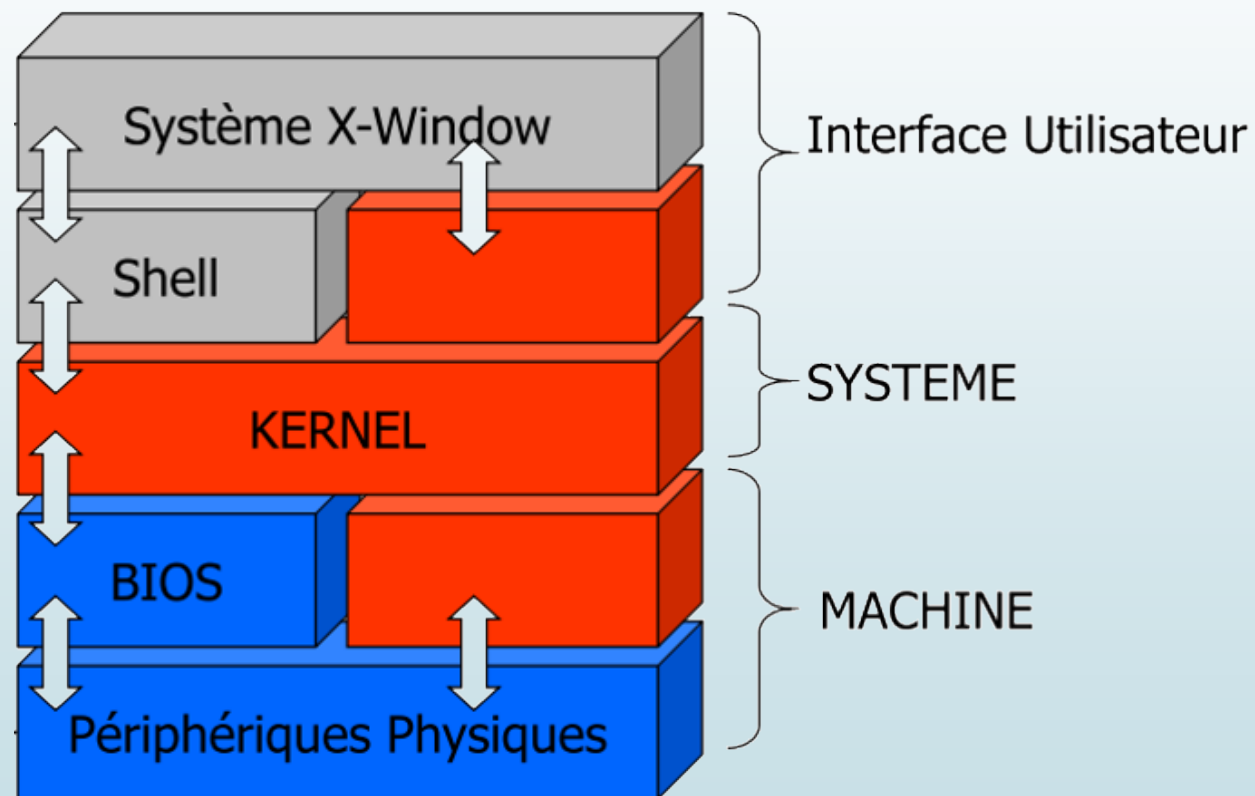
# Distributions LINUX

- Le mot distribution est issu de la phrase anglaise « software distribution »
- Une distribution GNU/LINUX est une collection de logiciels GNU (donc libres) qui forment un système d'exploitation.



- La distribution utilisée dans ce cours est **UBUNTU**

# Architecture d'un SE Linux



# Quelques applications dans LINUX

- Bureautique : LibreOffice, Openoffice.org, Abiword, Gnumeric ;
- Développement (IDE) : Eclipse, Netbeans ;
- Graphisme : The GIMP, Inkscape ;
- Multimédia : XMMS, VLC, Amarok, Mplayer ;
- Internet : Firefox, Konqueror, Opera ;
- Serveurs : HTTP, FTP, mail, DNS, etc



# Avantages et inconvénients de LINUX

## ► Avantages

- ❑ Gratuit : Code source disponible (Licence GPL) ;
- ❑ Distributions multiples (RedHat, Debian, Ubuntu, Mandriva, Mint, Suse, Fedora, ArchLINUX, ...) ;
- ❑ Système multitâche, multi-utilisateur et multiprocesseur
- ❑ Possibilité d'essais sans installation (Knoppix, Ubuntu) ;

## ► Inconvénients

- ❑ Demande une certaine connaissance en informatique ;
- ❑ Support de matériel récents non garanti ;
- ❑ Difficulté de paramétrage de quelque périphériques ;
- ❑ Peu d'applications commerciales/spécialisées

# Gestion de fichiers sous LINUX

## 1.1 Système de gestion de fichiers

Le système de gestion de fichiers permet à l'utilisateur de :

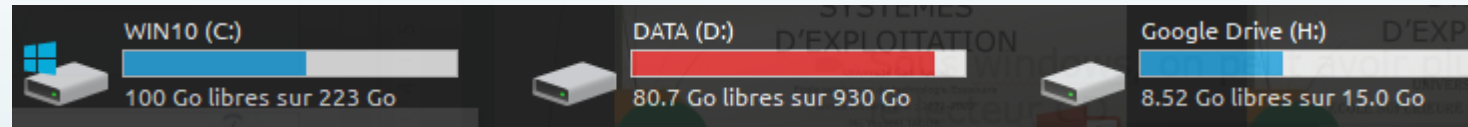
- Stocker les données sans qu'elles disparaissent lorsqu'on éteint l'ordinateur.
- Manipuler les fichiers : ouverture, copie, déplacement, suppression, ... etc.

Il existe trois types de fichiers :

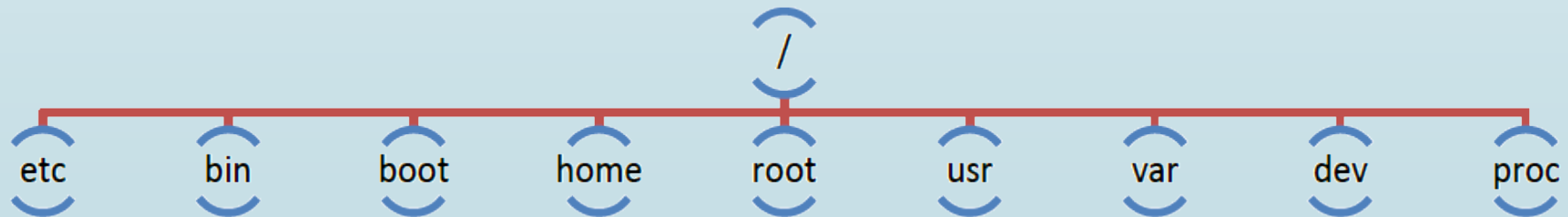
- fichier normal (image, texte, audio, vidéo, ...)
- répertoire (dossier)
- fichier spécial comme par exemple les fichiers représentant des périphériques matériel (imprimante) ou les liens symboliques (expliqués plus tard).

# Arborescence

- Sous Windows, on peut avoir plusieurs racines, C:\ ou D:\ ... ou E:\ pour le lecteur CD



- Sous LINUX, il n'y a qu'une et une seule racine : « / ». Il n'y a pas de lettre de lecteur, LINUX ne donne pas de nom aux lecteurs comme le fait Windows. Il dit juste « La racine, c'est / ». Au lieu de séparer chaque disque dur, lecteur CD, lecteur de carte mémoire ... LINUX place en gros tout au même endroit sous une seule racine.



# Arborescence

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

# Chemin relatif et chemin absolu

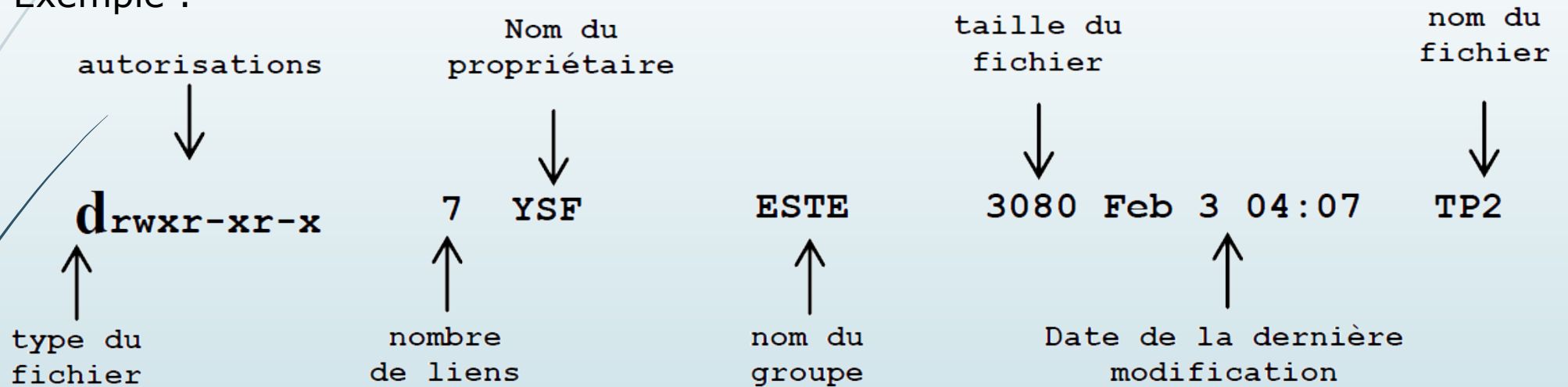
- Un chemin absolu spécifie la suite des répertoires à traverser **en partant de la racine**, séparés par des caractères / (et non \ comme sous Windows).
  - Exemple: `/usr/bin/biblio`
- Un chemin relatif commence par le répertoire courant.
  - Exemple: `../../home/ESTE_GI_S2`
- Tout chemin qui ne commence pas par un caractère / (slash) est interprété comme un chemin relatif au répertoire courant. On peut ainsi accéder aux fichiers du répertoire courant en donnant simplement leur nom.

# Les i-nodes des fichiers (index node)

Le i-node d'un fichier est un ensemble d'informations (metadata) qui définissent ce fichier :

Type | droits d'accès | nombre de liens | propriétaire | groupe | taille | date | nom

Exemple :



Type de fichier : (-)=fichier normal, (d)=répertoire, (l)=lien

Droits d'accès : r=lecture, w=écriture, x=exécution.

Autorisation pour 3 catégories d'utilisateurs : utilisateur, groupe, autres

# Commande sur l'environnement

- id:** affiche le nom de connexion et le numéro d'identifiant de l'utilisateur et du groupe auquel il appartient
- logname:** affiche le nom de connexion
- hostname:** affiche le nom de la machine
- uname:** affiche le nom du système d'exploitation
- clear:** efface l'écran
- who:** liste les utilisateurs connectés sur le même site



# Commandes usuelles de gestion de fichiers

## ■ Syntaxe générale d'une commande LINUX

Nom\_commande                      -options                      arguments

## ■ Commande pwd (print working directory)

Lorsqu'un utilisateur se connecte à une station LINUX, il est placé dans un répertoire personnel, en général situé dans le répertoire /home. La commande pwd affiche le chemin d'accès au répertoire courant de travail.

## ■ Commande cd (change directory)

**cd** ou **cd ~**      remonte au répertoire personnel.

**cd /**                      retourne à la racine (root)

**cd ..**                      retourne au répertoire parent.

**cd -**                      retourne au répertoire précédent.

## Commandes usuelles de gestion de fichiers

■ Commande ls (list files) : ls [-options] chemin

**ls** affiche les noms des fichiers et répertoires.

**ls -l** affiche les i-nodes des fichiers et répertoires.

**ls -a** affiche les noms de tous les fichiers et répertoires (même les fichiers cachés).

**ls -S** affiche les fichiers triés par ordre de taille décroissante.

**ls -s** affiche les fichiers triés par nom par ordre croissant.

**ls -r** affiche les fichiers triés par nom par ordre décroissant.

**ls -R** affiche tous les fichiers du sous arbre.

# Commandes usuelles de gestion de fichiers

## ► Commande cat

**cat file1**

affiche le contenu du fichier **file1**.

**cat file1 file2**

affiche le contenu des fichiers **file1** et **file2** (concaténés).

**cat > file**

permet de créer et écrire dans le fichier **file**

**cat >> file**

permet d'ajouter du contenu dans le fichier **file**.

**Cat -n file**

affiche les lignes du fichier **file** numérotées.

## ► Commande more, less

**more file**

affiche le contenu du fichier **file** page à page.

**less file**

permet en plus de revenir en arrière ou de rechercher une chaîne

## ► Commande head, tail

**head file**

affiche les premières lignes de fichier **file**.

**tail file**

affiche les dernières lignes de fichier **file**.

**head -n p file**

affiche les premières p lignes de fichier **file**.

**tail -n p file**

affiche les dernières p lignes de fichier **file**.

# Commandes usuelles de gestion de fichiers

## ► Commande touch

**Touch file1** permet de créer le fichier **file1**.

## ► Commande file

**file file\_name** indique le type de fichier **file\_name**.

## ► Commande mv (move)

**mv file1 rep** déplace le fichier **file1** dans le répertoire **rep**.

**mv -i file1** demande la confirmation avant le déplacement.

**mv file1 file2 rep** déplace **file1** et **file2** dans le répertoire **rep**.

**mv file1 file2** renommer le fichier **file1** en **file2**

**mv rep1 rep2** renommer le répertoire **rep1** en **rep2**

# Commandes usuelles de gestion de fichiers

## ➤ Commande cp

**cp file rep** copie le fichier **file** dans le répertoire **rep**.

**cp file1 file2** crée une copie du fichier **file1** appelée **file2**.

**cp -i** avertit l'utilisateur de l'existence d'un fichier du même nom et lui demande s'il peut ou non remplacer son contenu.

**cp -b** permet également de s'assurer que la copie n'écrase pas un fichier existant : le fichier écrasé est sauvegardé, seul le nom du fichier d'origine est modifié et cp ajoute une tilde (~) à la fin du nom du fichier.

**cp -p** permet lors de la copie de préserver toutes les informations concernant le fichier comme le propriétaire, le groupe, la date de création.

**cp -r** permet de copier de manière récursive un répertoire et ses sous-répertoires.

## Lien symbolique et lien dur

### ► Lien symbolique

Les liens symboliques représentant des pointeurs virtuels (raccourcis) vers le nom d'un fichier réel. En cas de suppression du lien symbolique, le fichier pointé n'est pas supprimé. Les liens symboliques sont créés à l'aide de la commande `ln -s` :

```
ln -s real_file_name symb_link_name
```

### ► Exemple

```
ln -s fichier1 lien_symb
```

```
ls -li fichier1 lien_symb donne : 6341 fichier1 4230 lien_symb
```

```
ls -l lien_symb
```

```
4230 lrwxrwxrwx 1 ..... lien_symb ➔ fichier1
```

## Lien symbolique et lien dur

- **Lien physique** : c'est la création de deux ou plusieurs noms vers une i-node unique au moyen de la commande `ln`. Ainsi, lorsqu'un fichier possède deux liens physiques, la suppression de l'un ou l'autre de ces liens n'entraîne pas la suppression du fichier. Lorsque le fichier d'origine est supprimé, alors le lien dur ne disparaît et fonctionne toujours.

`ln real_file_name hard_link_name`

- **Exemple :**

```
ls -li fichier1
```

```
6341 -rw-rw-r-- 1 ..... fichier1
```

```
ln fichier1 lien_dur
```

```
ls -li fichier1 lien_dur
```

```
6341 -rw-rw-r-- 2 ..... fichier1
```

```
6341 -rw-rw-r-- 2 ..... lien_dur
```

## Commandes usuelles de gestion de fichiers

### ► Commande mkdir (make directory) :

La commande mkdir sert à créer un répertoire. Le chemin de création peut être :

Relatif      `mkdir ../cours_SE`

Absolu      `mkdir /home/zaid/S2/cours_SE`

La commande `mkdir -p rep1/rep2/rep3` permet de créer récursivement la suite de répertoires rep1, rep2 et rep3.

### ► Commande du (disk usage) :

La commande **du** donne la taille occupée par un répertoire.

**du -a rep**      affiche la taille des sous repertoires du répertoire **rep**.

**du -h rep**      afficher la taille en Ko, Mo, Go



# Commandes usuelles de gestion de fichiers

## ► Commande **rm** (remove) :

La commande **rm** est utilisée pour supprimer un fichier

- rm -i** permet de demander à l'utilisateur s'il souhaite vraiment supprimer le ou les fichiers en question.
- rm -r** agit de façon récursive, c'est à dire détruit aussi les répertoires (pleins ou vides) et leurs sous-répertoires.
- rm -f** permet de supprimer les fichiers protégés en écriture et répertoires sans que le prompt demande une confirmation de suppression.
- rmdir rep** permet de supprimer le répertoire **rep**.
- rmdir -i rep** permet de supprimer le répertoire **rep** en demandant la confirmation de l'utilisateur.

# Recherche des fichiers

## Commandes LINUX pour la recherche des fichiers

### ➤ Commande locate (localiser) :

**locate file\_name** : cherche le chemin du fichier **file\_name** dans une base de données qui contient les chemins de tous les répertoires et fichiers.

### ➤ Commande find (trouver)

La commande **find** parcourt dans le disque dur le sous arbre du répertoire spécifié.

- ❑ La commande **locate** est plus rapide que **find**, mais elle ne trouve pas les fichiers qui n'ont pas été encore inclus dans la base de données.

Principe de la commande find : **find where what to\_do**

**Where** : le répertoire où la recherche doit être faite (non obligatoire).

**What** : le fichier qu'on cherche (obligatoire).

**to\_do** : ce qu'on veut faire avec le fichier après l'avoir trouvé (non obligatoire)

## Commandes LINUX pour la recherche des fichiers

### ➤ Commande find (trouver) :

<b>find -name file_name</b>	cherche tous les fichiers nommés <b>file_name</b> dans le répertoire courant.
<b>find rep -name file_name</b>	cherche tous les fichiers nommés <b>file_name</b> dans le répertoire <b>rep</b> .
<b>find -name "chaine*"</b>	cherche tous les fichiers dont le nom commence par <b>chaine</b> .
<b>find -name "*chaine"</b>	cherche tous les fichiers dont le nom se termine par <b>chaine</b> .
<b>find -name "*chaine*"</b>	cherche tous les fichiers dont le nom contient le mot <b>chaine</b> .
<b>find rep -size oct</b>	cherche dans le répertoire <b>rep</b> tous les fichiers dont la taille = <b>oct</b> .
<b>find rep -size +oct</b>	cherche dans le répertoire <b>rep</b> tous les fichiers dont la taille > <b>oct</b> .
<b>find rep -size -oct</b>	cherche dans le répertoire <b>rep</b> tous les fichiers dont la taille < <b>oct</b> .

Pour la recherche par taille : k (minuscule) pour kilo octets, M et G pour mega et giga

## Commandes LINUX pour la recherche des fichiers

### ► Commande find (trouver) :

**find -atime +nbr\_days**      cherche tous les fichiers qu'on a ouvert il y a plus de **nbr\_days** jours.

**find -atime -nbr\_days**      cherche tous les fichiers qu'on a ouvert il y a moins de **nbr\_days** jours.

**find -name nom -type d**      cherche tous les **répertoires** nommés **nom** dans le répertoire courant.

Pour l'action **to\_do**, on peut ajouter à la fin de la commande **find** :

**Find where what -delete** permet de supprimer les fichiers trouvés.

**Find where what -exec new\_command {} \;** permet d'abord de chercher les fichiers **what** dans le répertoire **where**, ensuite les fichiers trouvés sont référenciés par les accolades **{}** dans la commande **new\_command**. Le symbole **\;** est important pour l'exécution de **new\_command**.

Par exemple : la commande **Find ~/Downloads -name "\*.jpg" -exec cp {} ~/Pictures \;** va chercher toutes les images de format **jpg** se trouvant dans le répertoire **Downloads**, puis elle va les copier dans le répertoire **Pictures**. (on rappelle que **~** fait référence au répertoire personnel).

## Commandes LINUX pour le tri et découpage dans un fichier

### ► Commande sort (trier) :

**sort file\_name** fait le tri du contenu du fichier **file\_name** et l'affiche.

**sort -r file\_name** fait le tri inverse du contenu du fichier **file\_name** et l'affiche.

**sort -n file\_name** fait le tri des nombres dans le fichier **file\_name** et l'affiche.

### ► Commande cut (couper) :

**cut -c j-k file\_name** Pour chaque ligne du fichier **file\_name**, elle affiche du **j-ème** caractère au **k-ème**.

**cut -c -k file\_name** Pour chaque ligne du fichier **file\_name**, elle affiche du **1er** caractère au **k-ème**.

**cut -c j- file\_name** Pour chaque ligne du fichier **file\_name**, elle affiche du **j-ème** caractère au dernier.

**cut -d symb -f j file\_name** découpe les lignes selon **symb**, puis affiche seulement **j-ème** le champs. Le délimiteur **symb** peut être « **virgule** » ou « **:** » ou « **.** » Ou autre symbole.

## Filtres LINUX pour la recherche dans les fichiers

- **Commande grep (Global Regular Expression Parser)** : cette commande sert à chercher un mot dans un fichier et afficher les lignes où ce mot a été trouvé.

<b>grep word file_name</b>	affiche les lignes du fichier <b>file_name</b> qui contiennent le mot <b>word</b> .
<b>grep -i word file_name</b>	affiche les lignes du fichier <b>file_name</b> qui contiennent le mot <b>word</b> sans tenir compte des lettres minuscules et majuscules.
<b>grep -n word file_name</b>	affiche les lignes (et leurs numéros aussi) du fichier <b>file_name</b> qui contiennent le mot <b>word</b> .
<b>grep -v word file_name</b>	affiche les lignes du fichier <b>file_name</b> qui ne contiennent pas le mot <b>word</b> .
<b>grep -r word rep</b>	affiche les lignes de tous les fichiers dans <b>rep</b> qui contiennent le mot <b>word</b> .

## Commandes LINUX pour la recherche dans les fichiers

- Utilisation de **grep** avec des expressions régulières :

<b>grep ^word file_name</b>	affiche les lignes du fichier <b>file_name</b> qui commencent par le mot <b>word</b> .
<b>grep word\$ file_name</b>	affiche les lignes du fichier <b>file_name</b> qui se terminent par le mot <b>word</b> .
<b>grep [c1-c2] file_name</b>	affiche les lignes du fichier <b>file_name</b> qui contiennent des lettres compris entre <b>c1</b> et <b>c2</b> .
<b>grep [c1-c2c3-c4] file_name</b>	affiche les lignes du fichier <b>file_name</b> qui contiennent des lettres compris entre <b>c1</b> et <b>c2</b> et entre <b>c3</b> et <b>c4</b> .
<b>grep [n1-n2] file_name</b>	affiche les lignes du fichier <b>file_name</b> qui contiennent des nombres compris entre <b>c1</b> et <b>c2</b> .



## Commandes LINUX pour la recherche dan les fichiers

- Commande wc (word count) : compter les mots dans un fichier

<b>wc file_name</b>	affiche le nombre de lignes, mots et octets – dans cet ordre – du fichier <b>file_name</b> .
<b>wc -l file_name</b>	affiche le nombre de lignes seulement du fichier <b>file_name</b> .
<b>wc -w file_name</b>	affiche le nombre de mots seulement du fichier <b>file_name</b> .
<b>wc -c file_name</b>	affiche le nombre d'octets seulement du fichier <b>file_name</b> .
<b>wc -m file_name</b>	affiche le nombre de caractères seulement du fichier <b>file_name</b> .

# Gestion des utilisateurs sous LINUX

# Gestion des utilisateurs

- Linux est un système multi-utilisateur.
- Nécessité d'organisation : chaque personne a son propre compte utilisateur
- Un compte utilisateur ne peut pas tout faire dans Linux, il a des droits limités pour raisons de sécurité et stabilité du système.
- Si on attribue certaines commandes à un compte utilisateur, il risque d'endommager le système. Par exemple : `rm -rf`
- **Super utilisateur** : c'est le compte administrateur (**root**) qui peut tout faire dans le système Linux.
- Sous Windows, on se connecte toujours en tant qu'administrateur. Si un virus frappe, c'est dangereux. Ce n'est pas le cas sur Linux, car on se connecte juste en tant que compte utilisateur.

# Gestion des utilisateurs

- Identification d'un utilisateur:  
Nom: « login »  
Mot de passe: « password »
- Référencement de tous les utilisateurs dans le fichier « **/etc/passwd** »  
et/ou « **/etc/shadow** »
- Référence à un groupe: « **/etc/group** »
- Répertoire personnel: « **/home/<login>** »
- Devenir root pendant un instant : **sudo** (Substitute User DO)
- Devenir root toujours: **sudo su** (le symbole # apparait)
- Quitter root : **exit**

# Gestion des utilisateurs

- Le fichier « **/etc/passwd** » contient toutes les informations sur les comptes utilisateurs du système.
- Seul **root** peut les modifier.
- Chaque utilisateur est référencé par une ligne donnant:
  - ❑ Son login
  - ❑ Son numéro d'identification sur le système (uid)
  - ❑ Son numéro de groupe
  - ❑ Un commentaire (Nom complet en général)
  - ❑ Son répertoire personnel de base
  - ❑ Son SHELL par défaut

# Gestion des utilisateurs

- Commande d'ajout d'un compte

**sudo adduser user\_name**

Exemple : **sudo adduser zaid**

On vous demande d'entrer des informations sur le compte comme le nom complet et le mot de passe.

- Commande de changement de mot de passe :

**sudo passwd user\_name**

**Attention ! Si vous appelez passwd sans préciser user\_name, c'est le mot de passe de root que vous changerez.**

# Gestion des utilisateurs

- Commande de suppression d'un compte

**sudo deluser user\_name**

**Si vous supprimez votre compte utilisateur, il n'y aura plus que root sur la machine. UBUNTU interdit de se logger en root. Donc, au prochain démarrage de l'ordinateur, vous ne pourrez pas vous connecter.**

La commande « **deluser --remove-home user\_name** » supprime tous les fichiers personnels du compte **user\_name**.

- **Remarque** : On peut vérifier la création et suppression des comptes en regardant le contenu du répertoire /home

# Gestion des groupes

- Commande de création d'un groupe

**sudo addgroup group\_name**

- Commande d'ajouter un compte à un groupe

**sudo usermod -g group\_name username**

- Commande d'ajouter un compte à plusieurs groupes

**sudo usermod -G group1,group2 user\_name (annule les anciens groupes)**

**sudo usermod -aG group1,group2 user\_name (garde les anciens groupes)**

- Commande de supprimer un groupe

**Sudo delgroup group\_name**



# Changement de propriétaire d'un fichier

- Changer le compte propriétaire d'un fichier

```
sudo chown user_name file_name
```

- Changer le groupe propriétaire d'un fichier

```
sudo chgrp group_name file_name
```

```
sudo chown user_name:group_name file_name
```

- Changer les propriétaires d'un répertoire et tout son contenu

```
sudo chown -R user_name:group_name directory_name
```

# Les droits d'accès

- Chaque fichier possède 3 types de droits pour 3 types d'utilisateurs :
  - ❑ lecture (r)
  - ❑ écriture (w)
  - ❑ exécution(x)
- La commande `ls -l` permet d'afficher ces droits:
  - ❑ Première lettre : - si fichier normal, d si répertoire,
  - ❑ les 3 caractères suivants représentent les droits du propriétaire du fichier r w x.
  - ❑ idem pour groupe et pour autres.

# Les permissions - exemple

```
ysf@ysf-virtual-machine:~$ ls -l
total 44
drwxr-xr-x 2 ysf grp1 4096 18:23 30 یناير Desktop
drwxr-xr-x 2 ysf grp1 4096 23:22 21 یناير Documents
drwxr-xr-x 3 ysf grp1 4096 11:46 1 فبرایر Downloads
drwxrwxr-x 2 ysf grp1 4096 11:20 1 فبرایر folder_1
drwxrwxr-x 2 ysf grp1 4096 11:33 31 یناير folder_2
drwxr-xr-x 2 ysf grp1 4096 15:32 1 فبرایر Music
drwxr-xr-x 2 ysf grp1 4096 23:22 21 یناير Pictures
drwxr-xr-x 2 ysf grp1 4096 23:22 21 یناير Public
```

d :	type de fichier
<b>rwX :</b>	<b>droits d'accès ou permissions</b>
2 :	nombre de liens
ysf :	propriétaire
grp1 :	groupe
4096 :	taille
18:23 30 یناير :	date de la dernière modification
Desktop :	nom du fichier ou répertoire

# Droit d'accès sur les fichiers

décimale	binaire	permission
7	111	rwX
6	110	rw-
5	101	r-X
4	100	r--
3	011	-WX
2	010	-W-
1	001	--X
0	000	---

Permissions des fichiers

- Les commande pour la gestion d'accès:
  - chown : change le propriétaire
  - chgrp : change le groupe
  - chmod : Change les droits d'accès:

Exemple :

La commande **chmod 751 exemple.txt** indique que:

- Le propriétaire peut lire, écrire, et exécuter le fichier (7 ou encore rwX)
- Les membres du groupe peuvent lire et exécuter le fichier, mais ils ne peuvent pas y écrire (5 ou encore r-X)
- Les autres utilisateurs peuvent exécuter le fichier, mais ils ne peuvent ni le lire ni y écrire (1 ou encore --X)

## Droit d'accès sur les fichiers

- Commande chmod (change mode) :

Code	répertoire	fichier
r : lecture	Explorer	Voir le contenu
w : écriture	Ajouter ou supprimer des fichiers	Modifier le contenu
x : exécution	Accéder	Exécution

Le droit d'accès est codé sur un nombre binaire de 3 bits, le poids 2 correspond à la lecture, le poids 1 correspond à l'écriture, le poids 0 correspond à l'exécution.

Exemple : `chmod 754 file1`

$7 = 2^2 + 2^1 + 2^0 = (111)_2$ , donc l'utilisateur a tous les droits d'accès au fichier file1.

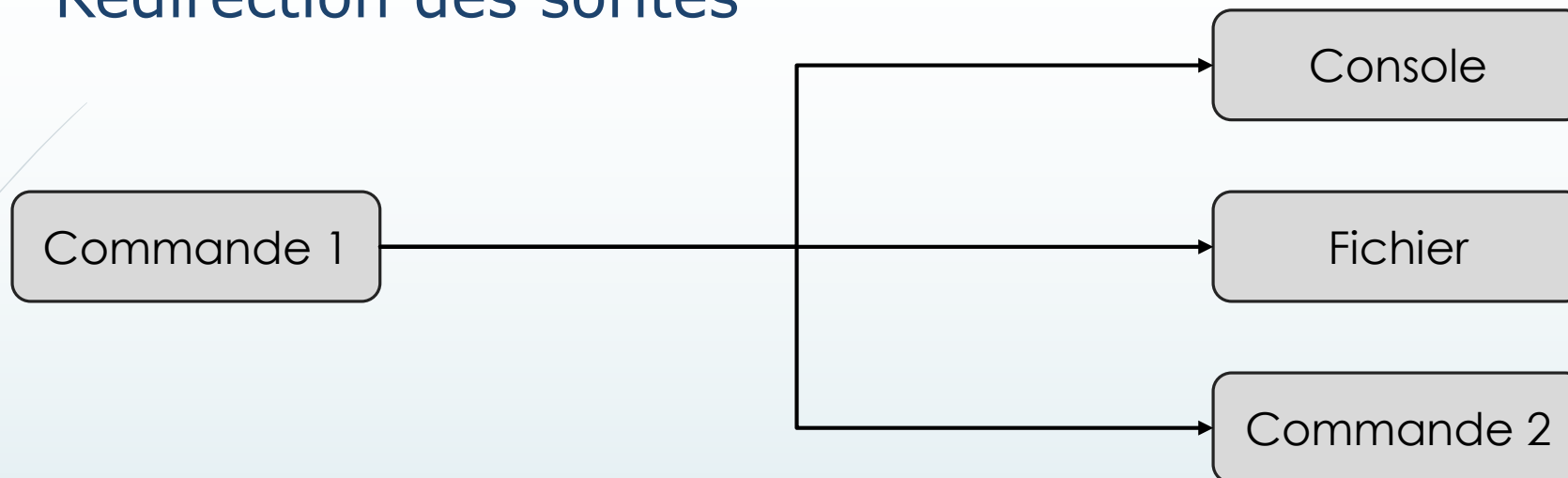
$5 = 2^2 + 2^0 = (101)_2$ , donc le groupe n'a pas le droit d'écriture.

$4 = 2^2 = (100)_2$ , les autres ont seulement le droit de lecture.

Donc `chmod 754 file 1` correspond à : `-rwxr-xr--`

# Pipes et redirection des entrées et des sorties

## Redirection des sorties



- Le symbole **>** et **>>** redirigent le résultat d'une commande vers un fichier :

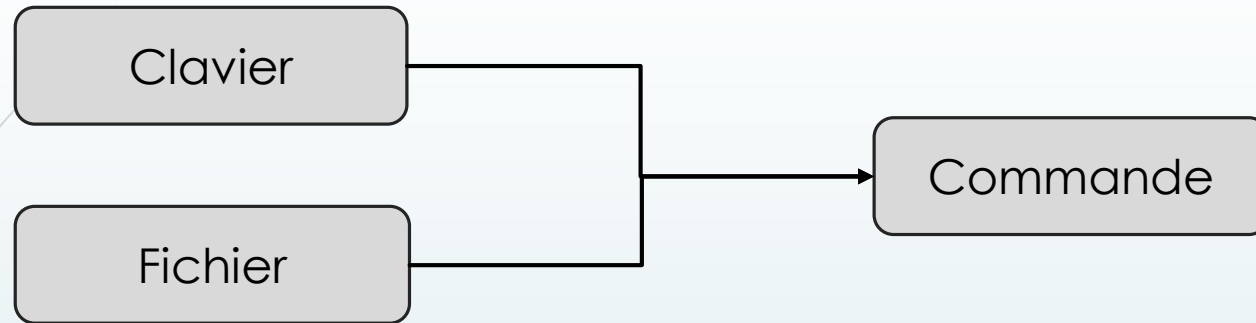
**ls -l rep > file\_name** remplace le contenu du fichier **file\_name** par les détails du contenu du répertoire **rep**.

**Sort file\_1 > file\_2** effectue le tri des lignes du fichier **file\_1** et stocke le résultat dans le fichier **file\_2** après avoir écrasé le contenu de **file\_2**.

**ls -l rep >> file\_name** ajoute les détails du contenu du répertoire **rep** à la fin du fichier **file\_name**.

**Sort file\_1 >> file\_2** effectue le tri des lignes du fichier **file\_1** et ajoute le résultat à la fin du fichier **file\_2**.

## Redirection des entrées



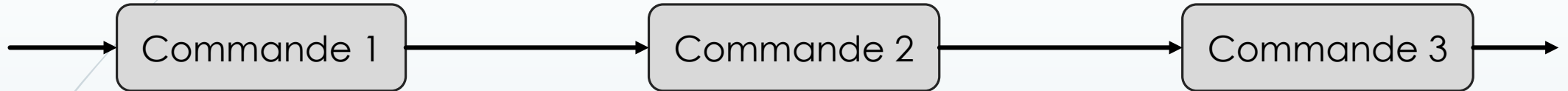
- Grace aux symboles `<` et `<<`, l'entrée d'une commande peut venir du clavier ou d'un fichier :

<b><code>sort &lt; file</code></b>	effectue le tri des lignes du fichier <b>file_1</b> et affiche le résultat dans la console.
<b><code>sort &lt;&lt; end_key_word</code></b>	donne la main à l'utilisateur pour écrire des mots tant qu'il n'a pas écrit le <b>end_key_word</b> , puis affiche ces mots après les avoir triés.
<b><code>sort &lt;&lt; end &gt; file</code></b>	donne la main à l'utilisateur pour écrire des mots tant qu'il n'a pas écrit le mot <b>end</b> , puis fait le tri de ces mots et les stocke dans le fichier <b>file</b> .

**Remarque :** **end** n'est pas un mot clé dans la commande, il peut être remplacé par un mot quelconque.



## Pipe : chaîner les commandes



➤ **commande1 | commande2 | commande3 | ...**

La **commande1** est effectuée d'abord, puis son résultat sera entrée de la **commande2**, la **commande3** s'effectue sur le résultat de la **commande2**, etc.

**date | cut -d -f 2**

la commande donne la date (jour, h:mn:s), la commande sépare les éléments de la date, puis retient seulement les minutes.

supposons qu'un fichier nommé **NOTES** contient les noms et notes des élèves comme suit :

Ayoub, 18

israa, 19

Yaaqoub, 17

Ecrire une pipe de commandes afin d'afficher les notes des élèves seulement après les avoir trié.

Réponse : `cut -d , -f 2 NOTES | sort -nr`

# Editeur Vim

# Editeur Vim



```
ysf@ysf-virtual-machine: ~  
  
      VIM - Vi IMproved  
      version 9.0.242  
      by Bram Moolenaar et al.  
      Modified by team+vim@tracker.debian.org  
      Vim is open source and freely distributable  
  
      Help poor children in Uganda!  
type  :help iccf<Enter>      for information  
  
type  :q<Enter>              to exit  
type  :help<Enter> or <F1>   for on-line help  
type  :help version9<Enter> for version info
```

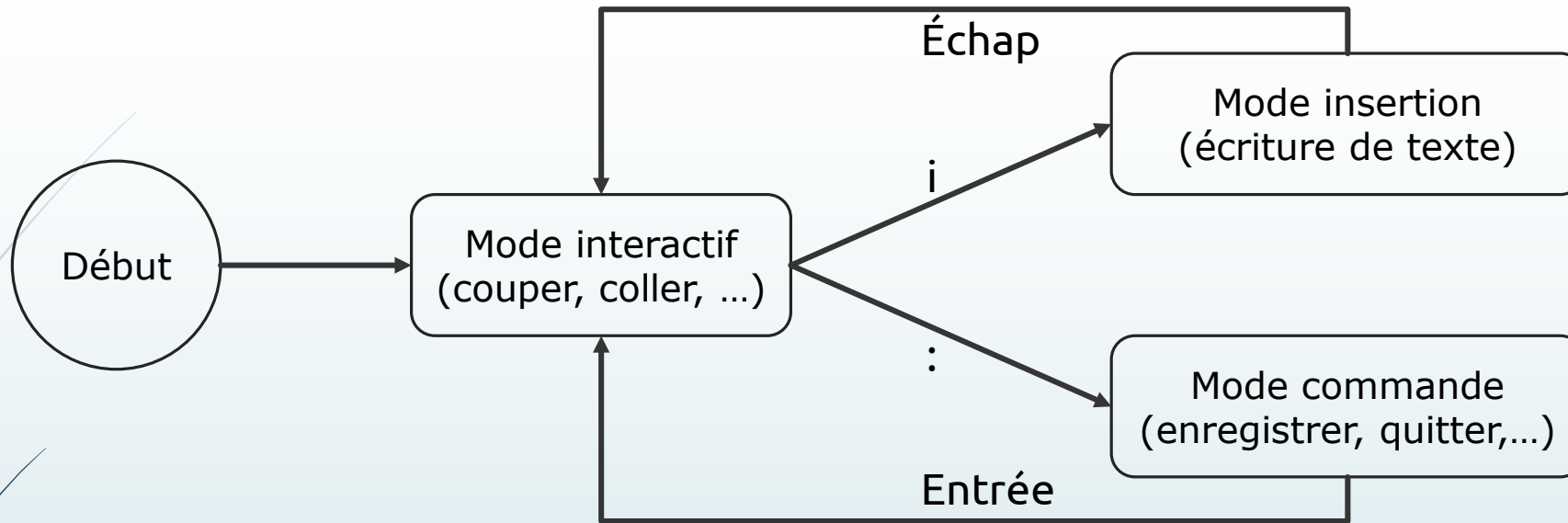
➤ Edition d'un fichier :

**sudo apt install vim** permet d'installer **vim** au cas où vous ne le trouvez pas sur votre machine.

**vimtutor** ouvre un tutoriel expliquant en détails l'utilisation de **vim**.

**vim file\_name** ouvre le fichier **file\_name** dans **vim** pour modifier son contenu.

# Modes de Vim



► Vim commence toujours en mode interactif

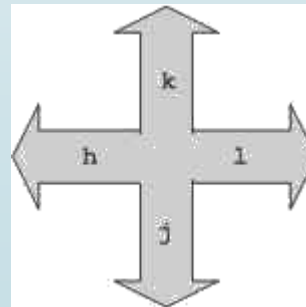
i permet d'insérer du texte, le message --INSERT-- apparaît en bas.

h se déplacer à gauche

l se déplacer à droite

j se déplacer en bas

k se déplacer en haut



Les flèches permettent aussi de se déplacer en mode commande.

## Options de Vim en mode commande

0	se déplacer en début de ligne. (la commande est le nombre zéro et non pas la lettre o)
\$	se déplacer en fin de ligne.
w	se déplacer de mot en mot.
:w	enregistrer le fichier
:q	quitter (on nous demande d'enregistrer avant de quitter).
:q!	forcer la fermeture si on ne veut pas enregistrer.
:wq	enregistrer le fichier et quitter
x	effacer la lettre colorée par le curseur. (même chose que supprimer en mode insertion).
dw	supprimer un mot.
dd	supprimer une ligne.
yy	copier une ligne
p	coller
R	chercher et remplacer du texte
u	annuler une modification faite

# Gestion des processus sous Linux

## Surveillance du système

- La commande « w » affiche les utilisateurs de la machine et affiche également ce qu'ils sont en train de faire.

```
ysf@ysf-virtual-machine: ~  
ysf@ysf-virtual-machine:~$ w  
08:40:01 up 55 min,  1 user,  load average: 0.03, 0.31, 0.39  
USER      TTY      FROM          LOGIN@   IDLE   JCPU   PCPU   WHAT  
ysf       tty2     tty2          07:45    54:56  0.08s  0.08s  /usr/libexec/gn  
ysf@ysf-virtual-machine:~$
```

**08:40:01** le moment où la commande **w** a été exécutée.

**Up 55 min** durée de fonctionnement de l'ordinateur.

**LOGIN@** heure où la session a été ouverte.

**IDLE** durée où l'utilisateur est inactif.

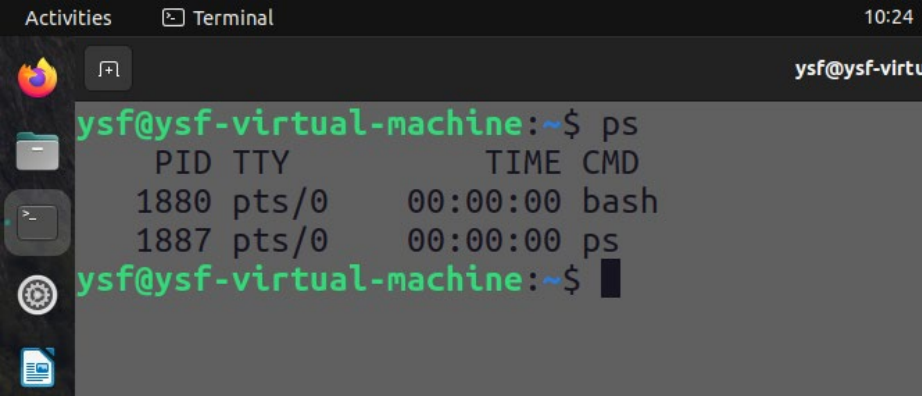
- La commande « who » affiche la liste des utilisateurs connectés sur la machine.

# Processus

► Un processus est un programme qui est en cours d'exécution.

► Commande ps : liste statique des processus

**PID :** c'est le numéro d'identification du processus. Il est utile lorsqu'on souhaite appliquer des opérations sur le processus.



```
ysf@ysf-virtual-machine:~$ ps
  PID TTY          TIME CMD
 1880 pts/0        00:00:00 bash
 1887 pts/0        00:00:00 ps
ysf@ysf-virtual-machine:~$
```

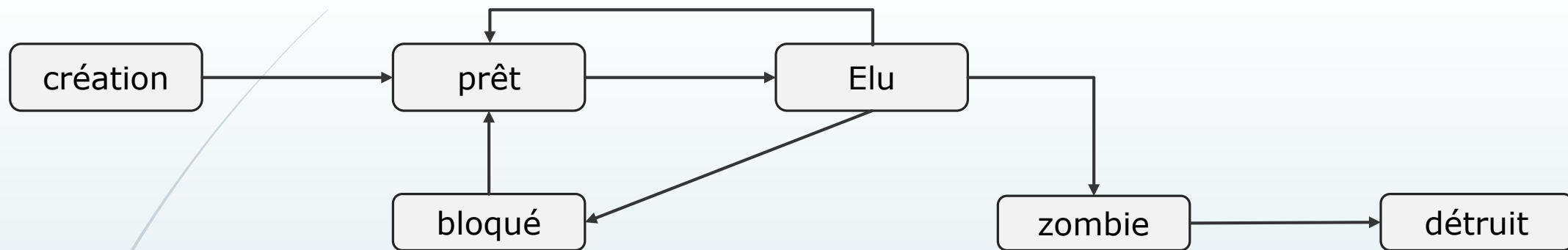
**TTY :** c'est le nom de la console depuis laquelle a été lancé le processus.

**TIME :** correspond à la durée pendant laquelle le processus a occupé le processeur depuis son lancement. (durée d'exécution du processus)

**CMD :** le programme qui a généré ce processus. Si vous voyez plusieurs fois le même programme, c'est que celui-ci s'est dupliqué en plusieurs processus.



## Cycle de vie d'un processus



**Création → prêt**

dans la file d'attente, attend la libération du processeur.

**Prêt → élu**

en cours d'exécution.

**élu → prêt**

épuisement du temps alloué (quantum).

**élu → bloqué**

attente d'une ressource nécessaire pour continuer l'exécution.

**bloqué → prêt**

ressource disponible.

**élu → zombie**

fin d'exécution.

**Zombie → détruit**

libération des ressources.

# Commande ps

```

ysf@ysf-virtual-machine:~$ ps -ef | head -30
UID          PID    PPID  C   STIME TTY          TIME CMD
root           1      0  0  10:22 ?        00:00:02 /sbin/init splash
root           2      0  0  10:22 ?        00:00:00 [kthreadd]
root           3      2  0  10:22 ?        00:00:00 [rcu_gp]
root           4      2  0  10:22 ?        00:00:00 [rcu_par_gp]
root           5      2  0  10:22 ?        00:00:00 [slub_flushwq]
root           6      2  0  10:22 ?        00:00:00 [netns]
root           8      2  0  10:22 ?        00:00:00 [kworker/0:0H-events_highpri]
root          10      2  0  10:22 ?        00:00:00 [mm_percpu_wq]
root          11      2  0  10:22 ?        00:00:00 [rcu_tasks_kthread]
root          12      2  0  10:22 ?        00:00:00 [rcu_tasks_rude_kthread]
root          13      2  0  10:22 ?        00:00:00 [rcu_tasks_trace_kthread]
root          14      2  0  10:22 ?        00:00:00 [ksoftirqd/0]
root          15      2  0  10:22 ?        00:00:00 [rcu_preempt]
root          16      2  0  10:22 ?        00:00:00 [migration/0]
root          17      2  0  10:22 ?        00:00:00 [idle_inject/0]
root          19      2  0  10:22 ?        00:00:00 [cpuhp/0]
root          20      2  0  10:22 ?        00:00:00 [cpuhp/1]
root          21      2  0  10:22 ?        00:00:00 [idle_inject/1]
root          22      2  0  10:22 ?        00:00:00 [migration/1]
root          23      2  0  10:22 ?        00:00:00 [ksoftirqd/1]
root          25      2  0  10:22 ?        00:00:00 [kworker/1:0H-events_highpri]
root          26      2  0  10:22 ?        00:00:00 [kdevtmpfs]
root          27      2  0  10:22 ?        00:00:00 [inet_frag_wq]
root          28      2  0  10:22 ?        00:00:00 [kauditd]
root          30      2  0  10:22 ?        00:00:00 [khungtaskd]
root          33      2  0  10:22 ?        00:00:00 [oom_reaper]
root          34      2  0  10:22 ?        00:00:00 [writeback]
root          35      2  0  10:22 ?        00:00:00 [kcompactd0]
root          36      2  0  10:22 ?        00:00:00 [ksmd]
ysf@ysf-virtual-machine:~$

```

**ps -ef** affiche **tous** les processus lancés par tous les utilisateurs.

**UID** Numéro identificateur de l'utilisateur.

**PPID** Numéro identifiant du processus parent (qui a lancé le processus actuel)

**C** facteur de priorité, plus la valeur est grande, plus le processus est prioritaire.

**STIME** l'heure de lancement du processus.

## Commandes ps et top

**ps -ejH** fait le même travail que **ps -ef** mais regroupe les processus sous forme d'arborescence, cela permet de visualiser quel processus a créé quels processus.

**ps -u user\_name** affiche tous les processus lancés par l'utilisateur **user\_name**.

### Commande top :

- la liste de ps ne bouge pas, elle affiche les processus au moment de lancement de ps
- la commande top permet d'afficher la liste des processus en temps réel
- La commande top considère les changements qui surviennent après l'avoir exécuté.
- Elle s'actualise chaque trois secondes.
- Pour fermer la commande top, on tape la lettre « q » (elle ne s'affiche pas).
- La lettre « h » affiche le « help » qui indique les options de top.

## Arrêt des processus

**Ctrl + C** arrête un processus qui est lancé sur la console.

► Commande kill :

**kill PID1 PID2 ...** arrête les processus identifiés par **PID1 PID2 ...**

**kill -9 PID** force l'arrêt du processus identifié par **PID** (au cas om il se plante).

**killall name** arrête tous les processus ayant le nom **name**. Elle est utile pour arrêter les processus qui se dupliquent plusieurs fois.

► Commande d'arrêt de redémarrage :

**sudo halt** arrête la machine.

**sudo reboot** redémarre la machine.

## Exécution des processus en arrière plan

► Commande &:

**Command\_to\_do &** exécute la **Command\_to\_do** en arrière plan afin de libérer la console.

**Command\_to\_do 2>&1 &** exécute la **Command\_to\_do** en arrière plan et renvoie les erreurs hors de la console.

La commande & renvoie deux paramètres à la console :

[nbr\_1] : c'est le numéro du processus en arrière-plan dans cette console.

nbr\_2 : c'est le numéro d'identification du processus (PID).

Défaut de la commande & : l'exécution de la commande est attaché à la console.

## Exécution des processus en arrière plan

- Nohup command\_to\_do** exécute **command\_to\_do** d'une manière indépendante de la console.
- Ctrl+Z** met en pause l'exécution de la commande.
- bg** si un processus est mis en pause, elle reprend son exécution mais en arrière plan pour libérer la console.
- jobs** affiche tous les processus qui tournent en arrière plan.
- fg** renvoie un processus en avant plan.
- Commande **at** : exécute une commande à une heure précise.
- at hour : minutes** affiche **>** et nous donne la main pour écrire la(es) commande(s) qu'on souhaite exécuter au moment **hour : minutes**.

## Contrôler l'instant d'exécution

**at hour : minutes month/day/year** précise le jour où on veut exécuter la commande.

**at hour : minutes tomorrow** l'exécution commence demain.

**at now +n minutes** l'exécution commence après **n** minutes.

**On peut remplacer minutes par : hours, days, weeks, months, years.**

Pour mémoriser les commandes devant s'exécuter au futur :

**atq** affiche la liste des jobs futurs.

**atrm** efface un job futur

# Fin Gestion de Processus