Module 2 : Gestion des Fichiers

Mohamed KHALED

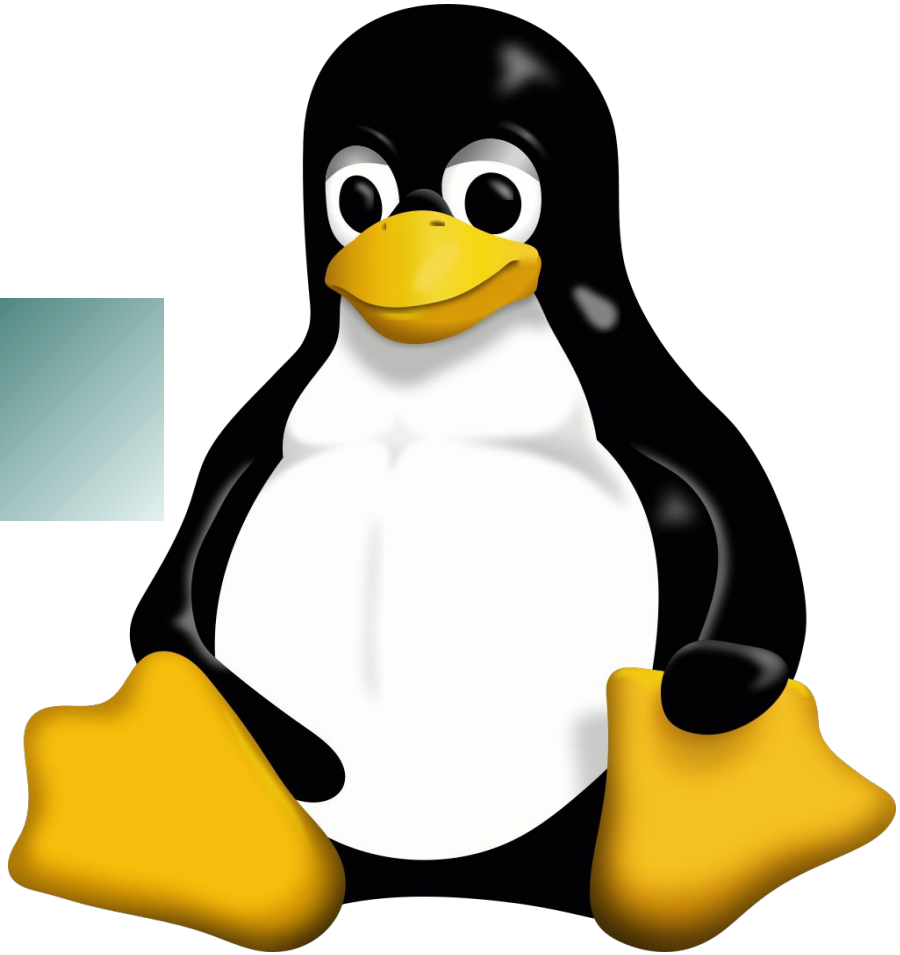
Objectifs

01 Navigation et Gestion de Fichiers

02 Flux, Redirections et Pipes

03 Consultation et filtrage de fichiers

04 Recherche de Fichiers et de Répertoires



Navigation et Gestion de Fichiers

Les Concepts Clés de la Navigation

La navigation repose sur une structure en arborescence unique, commençant à la **racine** (`/`).

1. **Chemin Absolu :** Un chemin d'accès complet qui commence à la racine (`/`). Il permet d'atteindre n'importe quel emplacement sans ambiguïté, quel que soit votre répertoire actuel.
    - *Exemple :* `/home/utilisateur/Documents`
2. **Chemin Relatif :** Un chemin d'accès qui commence à partir de votre répertoire actuel.
    - *Exemple :* Pour aller de `/home/utilisateur` à `/home/utilisateur/Documents`, vous utilisez simplement `Documents`.
3. **Répertoires Spéciaux :**
    - `.` : Représente le répertoire **actuel**.
    - `..` : Représente le répertoire **parent** (le niveau supérieur).
    - `~` : Représente votre **répertoire personnel** (`/home/utilisateur`).

Commandes de Navigation Essentielles

1. **`pwd:`** **P**rint **W**orking **D**irectory (Afficher le répertoire de travail)`:` Affiche le chemin complet de votre emplacement actuel.
2. **`cd:`** **C**hange **D**irectory (Changer de répertoire)

   Exemple: `cd /etc` : Déplace vers le répertoire de configuration (`/etc`).
   `cd ..` : Remonte d'un niveau (vers le répertoire parent).
   `cd ~` ou simplement `cd` : Revient à votre répertoire personnel.

3. **`ls:`** **Lis**t (Lister) Liste le contenu du répertoire actuel.
   - `ls -l` : Liste les détails (permissions, taille, date).

     Exemple:
     $ ls -l
     -rw-r--r-- 1 user staff 3606 Jan 13 2017 report2018.txt

   - `ls -a` : Liste tous les fichiers, y compris les fichiers cachés (qui commencent par un point `.`)
   - `ls -lh`: La combinaison d'un listage long avec des tailles de fichiers lisibles par l'homme (human readable) nous donnera des suffixes utiles tels que M pour mégaoctets ou K pour kilooctets.

Créer un Fichier Vide

La commande **`touch`** est la méthode la plus courante pour créer rapidement un fichier vide. Si le fichier existe déjà, `touch` met à jour son horodatage.

- **Syntaxe :** `touch [nom_du_fichier]`
- **Exemple :** Créer un fichier nommé `notes.txt`

Créer un Répertoire

La commande **`mkdir`** (**M**a**k**e **D**i**r**ectory) permet de créer un nouveau dossier.

- **Syntaxe :** `mkdir [nom_du_répertoire]`
- **Exemple :** Créer un dossier nommé `Rapports`.

  Option Utile : Utilisez l'option `-p` pour créer un répertoire ainsi que ses répertoires parents s'ils n'existent pas.

  mkdir -p Projet/Images/Sources

Note: Contrairement à Microsoft Windows, les noms de fichiers et de répertoires sur les systèmes Linux sont sensibles à la casse. Cela signifie que les noms /etc/ et /ETC/ sont des répertoires différents.

Copier des Fichiers et Répertoires

La commande **`cp`** (**C**o**p**y) sert à dupliquer des fichiers ou des répertoires.

> **Syntaxe :** `cp [source] [destination]`

Exemple 1 (Copier un fichier) : Copier `notes.txt` vers le répertoire `Rapports`.
cp notes.txt Rapports

Exemple 2 (Renommer en copiant) : Copier et renommer `notes.txt` en `notes_v2.txt` dans le même répertoire.
cp notes.txt notes_v2.txt

- **Option Importante pour les Répertoires :** Pour copier un répertoire et son contenu, vous devez utiliser l'option **récursive** (`-r` ou `-R`).
  cp -r Rapports/ Sauvegardes/

Déplacer et Renommer des Fichiers et Répertoires

La commande `mv` (**M**o**V**e) sert à la fois à **déplacer** des fichiers et à les **renommer**. C'est le même mécanisme pour les deux actions.

Déplacer

- **Syntaxe :** `mv [source] [destination]`
- **Exemple :** Déplacer `notes.txt` (et le retirer de l'emplacement actuel) vers le répertoire `Rapports`.

  mv notes.txt Rapports/

Renommer

Pour renommer, on déplace le fichier vers un nouvel emplacement *dans le même répertoire* sous un nouveau nom.

- **Syntaxe :** `mv [ancien_nom] [nouveau_nom]`
- **Exemple :** Renommer `notes_v2.txt` en `notes_final.txt`.

  mv notes_v2.txt notes_final.txt

Supprimer des Fichiers et Répertoires

Supprimer un Fichier

La commande `rm` (**R**e**M**ove) est utilisée pour supprimer des fichiers de manière permanente. Soyez **extrêmement prudent**, car cette suppression est définitive et ne passe pas par une corbeille.

- **Syntaxe :** `rm [nom_du_fichier]`
- **Exemple :** Supprimer `fichier_obsolete.txt`.

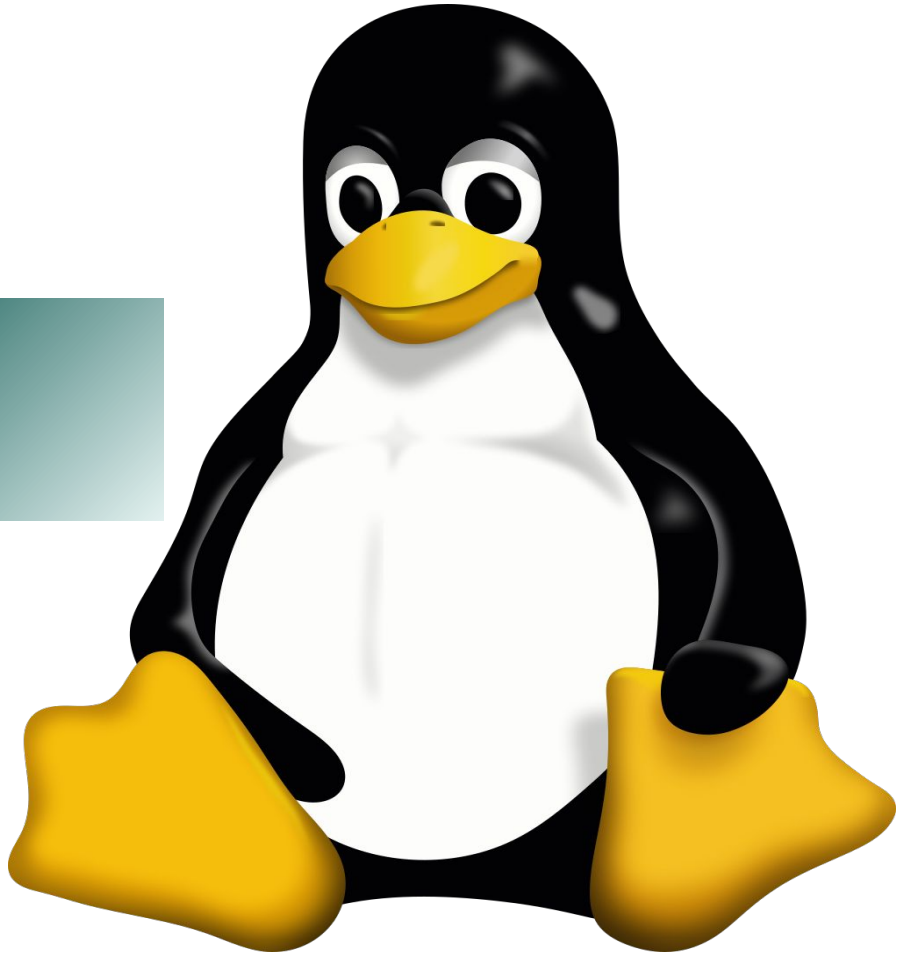
  rm fichier_obsolete.txt

Supprimer un Répertoire

`rmdir` (**R**e**M**ove **Dir**ectory) : Ne fonctionne **que** si le répertoire est **vide**.
rmdir Rep_Vide

`rm -r` : Pour supprimer un répertoire **non vide** et tout son contenu (fichiers et sous-répertoires). C'est l'outil le plus puissant, à utiliser avec une grande précaution.

> **Option de Sécurité :** L'option `-i` (interactive) demande une confirmation pour chaque fichier, ce qui est recommandé pour éviter les erreurs.



Flux, Redirections et Pipes

Les redirections:

Les **redirections** sous **Debian** (et plus généralement sous **Linux**, dans le **shell Bash**) permettent de contrôler où vont les **entrées** et **sorties** d'un programme — par exemple, envoyer la sortie d'une commande dans un fichier ou rediriger les erreurs ailleurs.

Les types de flux

Flux	Nom	Descripteur	Symbole	Description
stdin	Standard Input	0	<	Entrée standard (par défaut, le clavier)
stdout	Standard Output	1	> ou >>	Sortie standard (par défaut, l' écran)
stderr	Standard error	2	2> ou 2>>	Sortie des erreurs (par défaut, l'écran aussi)

Les redirections:

Le Concept Clé : Par défaut, votre terminal est à la fois la source d'entrée (votre clavier) et la destination de sortie (votre écran). Les redirections et les pipes permettent de modifier ces destinations/sources.

Les **redirections** permettent de changer la source d'entrée ou la destination de sortie et d'erreur.

Redirection de Sortie:

`>` (Redirection simple) : Redirige la **Stdout** vers un **fichier**, **écrasant** le contenu existant du fichier s'il existe.

- *Exemple :* `echo "Bonjour" > fichier.txt` (Crée ou écrase `fichier.txt` avec "Bonjour").

`>>` (Redirection avec ajout) : Redirige la **Stdout** vers un **fichier** en **ajoutant** le nouveau contenu à la fin.

- *Exemple :* `echo "Au revoir" >> fichier.txt` (Ajoute "Au revoir" à la fin de `fichier.txt`).

Les redirections:

Redirection de l'Erreur Standard (Stderr)

- `2>` : Redirige la **Stderr** vers un fichier, écrasant le contenu.
  - *Exemple :* `commande_erronnee 2> erreurs.log` (Envoie les messages d'erreur dans `erreurs.log`).
- `2>>` : Redirige la **Stderr** vers un fichier, en ajoutant au contenu.

Redirection de `stdout` et `stderr` vers des fichiers séparés :

commande > sortie_normale.txt 2> erreurs.log

Redirection Combinée (Stdout et Stderr)

commande > sortie_et_erreurs.txt 2>&1

ou:

commande &> sortie_et_erreurs.txt

Les redirections:

Redirection de l'Entrée Standard (stdin)

Ceci permet d'utiliser le contenu d'un fichier comme entrée pour une commande, au lieu de devoir la saisir au clavier.

Redirection de `stdin` (`<`) :
Bash
commande < fichier_entree.txt

- Par exemple, la commande `wc -l` (compter le nombre de lignes) peut lire l'entrée depuis un fichier :
  wc -l < liste.txt

Les redirections:

Les Pipes (Tubes)

Le **Pipe** (symbole : `|`) est un mécanisme qui permet de **connecter la sortie standard (stdout) d'une commande à l'entrée standard (stdin) d'une autre commande**.

C'est l'un des concepts les plus puissants du Shell, permettant de chaîner de petites commandes simples pour réaliser des tâches complexes.

Syntaxe et Fonctionnement

- commande1 | commande2 | commande3

1. `commande1` s'exécute.
2. Sa **Standard Output** est capturée.
3. Cette sortie capturée devient la **Standard Input** de `commande2`.
4. `commande2` traite ces données.
5. Sa sortie devient l'entrée de `commande3`, et ainsi de suite.

Les redirections:

Exemple Classique

Compter le nombre de processus en cours pour l'utilisateur `root` :

```
ps aux | grep root | wc -l
```

- **`ps aux`** : Affiche tous les processus du système (génère une longue liste).
- **`| grep root`** : Prend la sortie de `ps aux` et ne garde que les lignes contenant le mot "root".
- **`| wc -l`** : Prend la sortie filtrée et compte le nombre de lignes restantes.

Sans Pipe (Résultat Non Gérable)

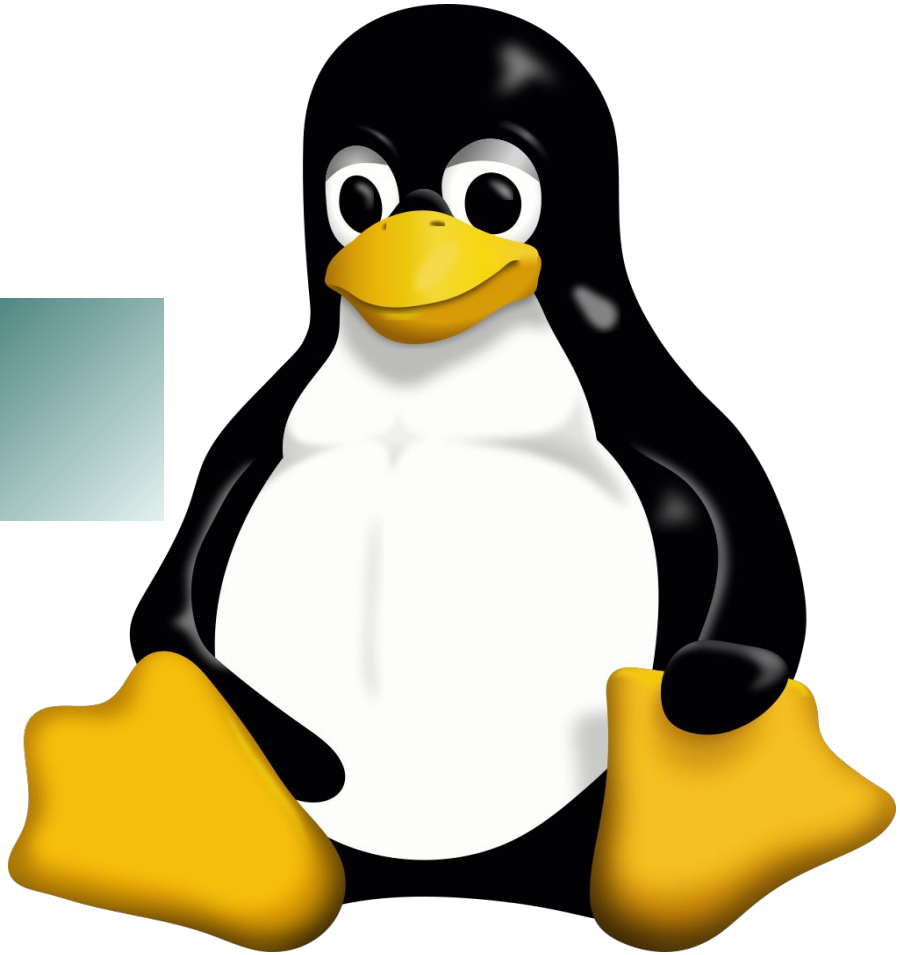
ls -l /usr/bin

La sortie (Standard Output) de `ls` va directement à l'écran et défile rapidement.

Avec Pipe (Résultat Filtré et Gérable)

ls -l /usr/bin | less

Le Principe du Pipe en une phrase: Le pipe (`|`) permet de dire : **"Prends le résultat de ceci, et donne-le à cela."**



Consultation et filtrage de texte

Afficher le contenu d'un fichier

Pour les Petits Fichiers

La commande cat (catenate) est la plus simple et affiche le contenu entier du fichier directement dans le terminal, sans pause.

Syntaxe : cat [nom_du_fichier]

Utilisation : Idéale pour les fichiers de configuration courts ou les petits scripts.

Options Utiles:

-n Affiche le contenu en numérotant toutes les lignes. cat -n fichier.txt

-b Affiche le contenu en numérotant les lignes non vides seulement. cat -b fichier.txt

Exemple : `cat /etc/os-release` (Affiche les informations de la distribution Linux)

Afficher le contenu d'un fichier

Pour les Fichiers Longs

> Lorsque vous travaillez avec des fichiers volumineux (journaux, longs documents), il est préférable d'utiliser des **paginators** qui affichent le contenu écran par écran, permettant de naviguer.

`more` (Le Paginator Simple)

`more` est un outil plus ancien qui ne permet de naviguer que vers l'avant.

- **Syntaxe :** `more [nom_du_fichier]`
- **Navigation :** Utilisez la `Espace` pour avancer.

`less` (Le Paginator Moderne)

`less` est l'outil privilégié car il permet de se déplacer à la fois **vers l'avant et vers l'arrière** dans le fichier.

- **Syntaxe :** `less [nom_du_fichier]`
- **Navigation dans `less` :**
  - `Espace` : Page suivante.
  - `b` : Page précédente.
  - `q` : Quitter l'affichage.

Afficher le contenu d'un fichier

Afficher le Début ou la Fin du Fichier

Ces commandes sont essentielles pour vérifier rapidement les entêtes de fichiers ou suivre les journaux d'activité en temps réel.

Début du Fichier (Commande `head`)

Affiche les **dix premières lignes** par défaut.

- **Syntaxe :** `head [nom_du_fichier]`
- **Option Utile :** Utilisez **`-n N`** pour spécifier le nombre `N` de lignes à afficher.
    - *Exemple :* `head -n 5 fichier.txt` (Affiche les 5 premières lignes)

Fin du Fichier (Commande `tail`)

Affiche les **dix dernières lignes** par défaut.

- **Syntaxe :** `tail [nom_du_fichier]`
- **Option Utile (`-n N`) :** Spécifie le nombre `N` de lignes à afficher depuis la fin.
    - *Exemple :* `tail -n 20 /var/log/syslog` (Affiche les 20 dernières lignes du journal système)
- **Option de Suivi (`-f` ou `--follow`) :** Indispensable pour surveiller les fichiers journaux. Le terminal reste ouvert et affiche les **nouvelles lignes en temps réel** dès qu'elles sont ajoutées au fichier.
    - *Exemple :* `tail -f /var/log/apache2/error.log` (Pour suivre les erreurs d'Apache en direct).

Traitement et de Filtrage de Texte

`grep` (Global Regular Expression Print)

Recherche des lignes qui correspondent à un motif (expression régulière) dans un fichier ou dans un flux de données (pipe). C'est le **filtre** essentiel.

- **Usage :** `grep "motif_a_rechercher" nom_du_fichier`
- **Exemples Clés :**
  - `ls -l /etc | grep 'conf'` : Affiche uniquement les fichiers et dossiers dans `/etc` dont le nom contient "conf".
  - `grep -i "error" log.txt` : Recherche le mot "error", ignorant la casse (`-i`).
  - `grep -v "info" log.txt` : Inverse la recherche (`-v`), affiche toutes les lignes SAUF celles contenant "info".

`sort`

Trie les lignes d'un fichier texte ou d'un flux de données.

- **Usage :** `sort nom_du_fichier`
- **Exemple:**
  - `cat liste.txt | sort -r` : Trie les lignes par ordre alphabétique inverse (`-r`).

Traitement et de Filtrage de Texte

`wc` (Word Count)

Compte le nombre de lignes, de mots et/ou de caractères.

- **Usage :** `wc nom_du_fichier`
- **Exemples Clés :**
  - `wc -l fichier.txt` : Compte uniquement le nombre de **lignes** (`-l`).
  - `wc -w fichier.txt` : Compte uniquement le nombre de **mots** (`-w`).
  - `ls -l | wc -l` : Compte le nombre de fichiers et dossiers dans le répertoire actuel.

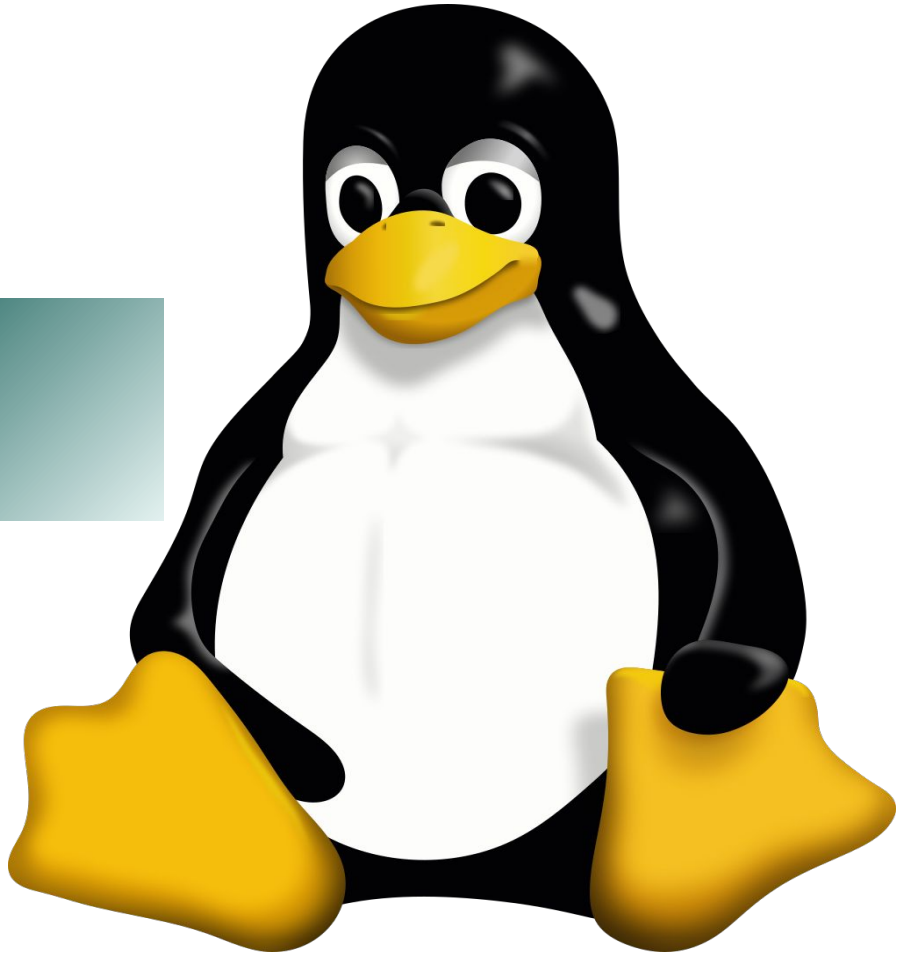
`cut`

Permet d'extraire des sections (champs ou colonnes) d'une ligne de texte. Très utile pour traiter des fichiers CSV ou des sorties de commandes structurées par des séparateurs.

- **Usage :** `cut -d "séparateur" -f numéro_de_champ nom_du_fichier`

Exemple : Extraire les noms d'utilisateur (premier champ) du fichier `/etc/passwd` où le séparateur est le deux-points (`:`).
Bash
cut -d ":" -f 1 /etc/passwd

  - `-d ":"` : Définit le deux-points comme délimiteur (séparateur).
  - `-f 1` : Spécifie d'afficher le premier champ.



Recherche de Fichiers et de Répertoires

Commande `find` (Recherche en temps réel)

La commande **`find`** parcourt l'arborescence des répertoires en temps réel, ce qui la rend très précise car elle reflète l'état actuel du système de fichiers.

Syntaxe de base

find [chemin_de_départ] [critères] [actions]

Recherche par Nom ou Type:

Rechercher un fichier par nom exact (sensible à la casse): `find /home -name "fichier.txt"`

Rechercher un fichier (insensible à la casse): `find / -iname "rapport.pdf"`

Rechercher des fichiers avec un motif (joker `*`): `find . -name "*.log"`

Rechercher seulement des dossiers (répertoires): `find /etc -type d -name "conf*"`

Rechercher seulement des fichiers: `find /var/log -type f -name "*.log"`

Commande `find` (Recherche en temps réel)

Recherche par Taille

Utilisez l'option `-size`. Les unités sont `c` (octets), `k` (kilo-octets), `M` (méga-octets), `G` (giga-octets). Utilisez **`+`** pour "plus grand que" et **`-`** pour "plus petit que".

Fichiers de plus de 100 Mo: `find / -type f -size +100M`

Fichiers de moins de 1 Mo: `find /home -type f -size -1M`

Fichiers exactement de 500 Ko: `find . -type f -size 500k`

Recherche par Date:

`-mtime -n`	modifié il y a moins de n jours	`-mtime -7`
`-mtime +n`	modifié il y a plus de n jours	`-mtime +30`
`-newermt "YYYY-MM-DD"`	modifié depuis une date	`-newermt "2025-10-01"`

Commande `locate` (Recherche par Index)

La commande **`locate`** est **beaucoup plus rapide** que `find` car elle ne parcourt pas l'arborescence en direct. Elle recherche dans une **base de données indexée** des fichiers (`/var/lib/mlocate/plocate.db`).

⚠️ **Inconvénient** : Les résultats sont basés sur la dernière mise à jour de la base de données. Les fichiers créés ou supprimés *après* la dernière indexation ne seront pas listés.

Installation et Mise à jour

1. **Installation** (si non installé) : `sudo apt install mlocate`
2. **Mise à jour de l'index** : `sudo updatedb` (nécessaire après l'installation et pour inclure les nouveaux fichiers).

locate [nom_du_fichier]

Rechercher le fichier hosts	locate hosts
Rechercher tous les fichiers .mp3	locate "*.mp3"
Rechercher (insensible à la casse)	locate -i apache