LINUX

Les fondations

Guide pratique pour technicien·ne·s

Code et Molécules

Édition 2026

Table des matières

CHAPITRE 0 : Introduction à Linux

0.1 Qu'est-ce que Linux ?

0.2 La philosophie Open Source

0.3 Les 3 piliers de la "Linux Way"

1. Tout est fichier

2. Petits outils, grandes tâches

3. La ligne de commande domine

0.4 La jungle des distributions

1. Les grandes familles :

0.5 Pourquoi apprendre Linux aujourd'hui ?

CHAPITRE 1 : Fondations et installation

1.1 Virtualisation et installation

1. Le concept de "Machine dans la machine"

2. Le processus d'installation

1.2 Arborescence et navigation

1. L'arbre Linux

2. Parler au Terminal

1.3 Gestion des paquets et édition

1. La fin des fichiers .exe (Les dépôts)

2. APT : Le standard Debian/Mint

3. L'éditeur VI / VIM

CHAPITRE 2 : L'Expansion des accolades

2.1. Le concept : Un générateur de texte

2.2. Les Listes (La virgule)

2.3. Les séquences (Les deux points)

1. Séquences numériques

2. Séquences alphabétiques

3. Le "Pas" (Sauter des nombres)

2.4. Combinaisons et "Nesting"

1. Préfixe et suffixe

2. Le produit cartésien (Multiplier les listes)

2.5. L'astuce des pros : La sauvegarde rapide

CHAPITRE 3 : Administration locale

3.1 Gestion des utilisateurs et groupes

1. L'identité numérique (UID & GID)

2. Les fichiers de l'annuaire

3. Commandes de gestion : Le cycle de vie

3.2 Permissions et sécurité

1. Le concept UGO et la trinité RWX

2. Les Maths de l'admin : La notation octale

3. Changer de propriétaire (chown)

3.3 Gestion des processus

1. Surveillance : L'art de savoir ce qui tourne

2. Arrêt des processus (Le nettoyeur)

3. Processus d'arrière-plan (multitâche)

4. Persistance (Survivre à la fermeture)

CHAPITRE 4 : Flux, filtres, stockage et réseau

4.1 Redirections, flux et filtres

1. Les canaux invisibles (flux)

2. Les redirections : Détourner la rivière

3. Le tube (Pipe |)

4. La boîte à outils (Les filtres)

4.2 Disques et partitionnement

1. Le matériel : Tout est fichier

2. Le cycle de vie d'un disque

3. La persistance (fstab)

4.3 Réseau et services

1. Interfaces et connexions (état des lieux)

2. Les fichiers texte importants

3. Commandes de gestion (NetworkManager)

4. Le chef d'orchestre : systemd

5. Sécurité et accès (SSH & UFW)

CHAPITRE 5 : Scripting et automatisation

5.1 Introduction au scripting

1. L'anatomie d'un script

2. Le rituel d'exécution

3. Les variables : des boîtes à mémoire

4. Rendre le script intelligent (Interactivité)

5.2 Contrôler le temps : sleep et wait

1. La sieste (sleep)

2. La synchronisation (wait)

5.3 Le flux de contrôle (logique)

1. Le code de retour ($?)

2. Les opérateurs de combinaison (&&, ||, ;)

3. La priorité et le groupement ( )

4. La structure IF / ELSE

5. Le "Cheat Sheet" des tests

5.4 Automatisation et boucles

1. La boucle FOR (Le traitement par lot)

2. La boucle WHILE (Le traitement par condition)

3. Danger : La boucle infinie

5.5 Planification (Cron)

1. L'édition (crontab -e)

GUIDE TECHNIQUE : Installation de Linux Mint (VirtualBox)

1. Prérequis et téléchargements
2. Création du "squelette" de la VM
3. Configuration optimisée
4. Installation du système
5. Post-Installation : Les Additions Invité

GUIDE TECHNIQUE : Création d'une clé USB d'installation

1. Prérequis matériels et logiciels
2. Méthode A : Depuis Windows (Recommandée)
3. Méthode B : Depuis Linux Mint (Natif)
4. Démarrage sur la clé (Boot)

CHAPITRE 0 : Introduction à Linux

Avant de taper votre première commande, vous devez comprendre où vous mettez les pieds. Linux n'est pas juste "un autre Windows". C'est une philosophie différente, une culture différente et le moteur qui propulse 90% d'Internet.

Objectifs du chapitre :

- Comprendre la différence entre le Noyau (Kernel) et le Système d'exploitation.
- Découvrir la philosophie du logiciel libre (Open Source).
- Naviguer dans la jungle des distributions (Pourquoi Mint ? Pourquoi Ubuntu ?).

0.1 Qu'est-ce que Linux ?

Strictement parlant, **Linux n'est pas un système d'exploitation**. Linux, c'est un noyau (Kernel).

ANALOGIE : LA VOITURE

Imaginez une voiture :

- **Le Moteur (Linux) :** C'est la pièce centrale qui fait tourner les roues. C'est complexe, puissant, mais inutilisable seul. Vous ne pouvez pas conduire un moteur posé sur la route.
- **La Carrosserie et le Tableau de bord (GNU) :** Ce sont les outils qui rendent le moteur utilisable (volant, pédales, fenêtres). C'est ce qu'on appelle les outils **GNU**.
- **Le Système Complet (GNU/Linux) :** C'est la voiture assemblée, prête à conduire.

C'est pour cela que les puristes disent souvent **"GNU/Linux"**. Linus Torvalds (le créateur finlandais) a créé le moteur en 1991, et Richard Stallman (le fondateur du mouvement libre) a fourni les outils pour construire la voiture autour.

0.2 La philosophie Open Source

Windows et macOS sont des systèmes **propriétaires** (fermés). C'est comme un restaurant : vous mangez le plat, mais le chef refuse de vous donner la recette. Vous n'avez pas le droit de modifier le plat ni de le revendre.

Linux est **Open Source** (libre). C'est comme une recette de grand-mère publiée dans le journal :

- **Liberté 0 :** Vous pouvez utiliser le logiciel pour n'importe quel usage.
- **Liberté 1 :** Vous pouvez étudier comment il fonctionne (le Code Source est ouvert).
- **Liberté 2 :** Vous pouvez le modifier pour l'adapter à vos besoins.
- **Liberté 3 :** Vous pouvez redistribuer vos modifications pour aider la communauté.

GRATUIT VS LIBRE

En anglais, "Free" veut dire *Gratuit* (Free Beer) et *Libre* (Free Speech).
Linux est gratuit, mais c'est surtout la **liberté** de contrôle qui intéresse les techniciens. Personne ne peut vous empêcher de réparer votre système Linux.

0.3 Les 3 piliers de la "Linux Way"

1. Tout est fichier

C'est le concept le plus important. Pour Linux, tout est un fichier texte.

- Votre disque dur ? C'est un fichier (`/dev/sda`).
- Votre carte réseau ? C'est un fichier.
- La configuration du serveur ? C'est un fichier texte.

Conséquence : Si vous savez éditer un fichier texte, vous savez tout configurer.

2. Petits outils, grandes tâches

Linux ne crée pas de "super-logiciels" qui font le café et la vaisselle. Il préfère des milliers de petits outils qui font **une seule chose parfaitement**.

On connecte ces outils entre eux (comme des Lego) pour créer des solutions complexes. C'est le principe du "Pipe" (`|`) que nous verrons au Chapitre 3.

3. La ligne de commande domine

L'interface graphique (souris/fenêtres) est considérée comme optionnelle. Un vrai serveur n'a pas d'écran. La ligne de commande est plus rapide, plus précise et consomme moins de ressources.

0.4 La jungle des distributions

Puisque la recette est publique, n'importe qui peut créer son propre Linux. C'est pourquoi il existe des centaines de versions différentes, appelées **Distributions** (ou "Distros").

1. Les grandes familles :

- **Famille Debian :** (Ubuntu, Linux Mint, Kali). Axée sur la facilité d'utilisation et la stabilité. C'est celle que nous utiliserons.
- **Famille Red Hat :** (RHEL, Fedora, AlmaLinux). Le standard dans les grandes entreprises américaines.
- **Famille Arch :** (Arch Linux, Manjaro). Pour les experts qui veulent tout construire eux-mêmes brique par brique.

ANALOGIE : LES SAVEURS DE CRÈME GLACÉE

Le noyau Linux est la crème et le sucre (la base).
Les distributions sont les saveurs :

- **Ubuntu/Mint :** Vanille/Chocolat (Populaire, tout le monde aime).
- **Kali Linux :** Piment fort (Spécialisé pour la sécurité/piratage).
- **Arch Linux :** "Faites votre propre mélange" (On vous donne les ingrédients bruts).

0.5 Pourquoi apprendre Linux aujourd'hui ?

Vous pourriez vous dire : "*Mais mon ordinateur de gaming est sous Windows !*". C'est vrai. Mais le monde professionnel tourne sous Linux.

- **Le Web :** 90% des serveurs web (Google, Facebook, Amazon) tournent sous Linux.
- **Le Cloud :** AWS, Azure et Google Cloud sont essentiellement des infrastructures Linux.
- **Le Mobile :** Android est basé sur un noyau Linux.
- **La Super-informatique :** 100% des 500 supercalculateurs les plus puissants du monde tournent sous Linux.

Apprendre Linux, ce n'est pas apprendre un vieux système des années 90. C'est apprendre le langage de l'infrastructure moderne.

CHAPITRE 1 : Fondations et installation

Bienvenue dans le monde de Linux. Ce chapitre n'est pas seulement technique, il est philosophique. Vous allez passer d'un utilisateur passif à un administrateur actif.

Objectifs du chapitre :

- Comprendre comment installer Linux sans toucher à votre Windows/Mac actuel (Virtualisation).
- Se repérer dans un système de fichiers qui ne possède pas de lettre `C:`.
- Apprendre à parler à l'ordinateur via le Terminal.

1.1 Virtualisation et installation

Beaucoup de débutants ont peur d'installer Linux car ils craignent d'effacer leurs données personnelles. C'est une peur légitime. La solution s'appelle la **Virtualisation**.

1. Le concept de "Machine dans la machine"

Imaginez que votre ordinateur actuel est une maison. Installer Linux directement reviendrait à raser la maison pour en construire une nouvelle. La virtualisation, c'est comme construire une maquette sécurisée à l'intérieur de votre salon.

- **L'Hôte (Host) :** C'est votre ordinateur physique (Windows ou macOS). Il fournit les muscles (CPU, RAM).
- **L'Hyperviseur :** C'est le logiciel chef d'orchestre (ex: VirtualBox, VMware). Il fait le lien.
- **L'Invité (Guest) :** C'est votre Linux Mint ou Ubuntu. Il croit être sur un vrai ordinateur, mais il vit dans une fenêtre.

> **AVANT DE COMMENCER**
>
> Si votre logiciel de virtualisation refuse de démarrer la machine, c'est souvent parce que l'option **VT-x** (Intel) ou **AMD-V** est désactivée dans le BIOS/UEFI de votre ordinateur physique. C'est la cause #1 des problèmes en début de session.

2. Le processus d'installation

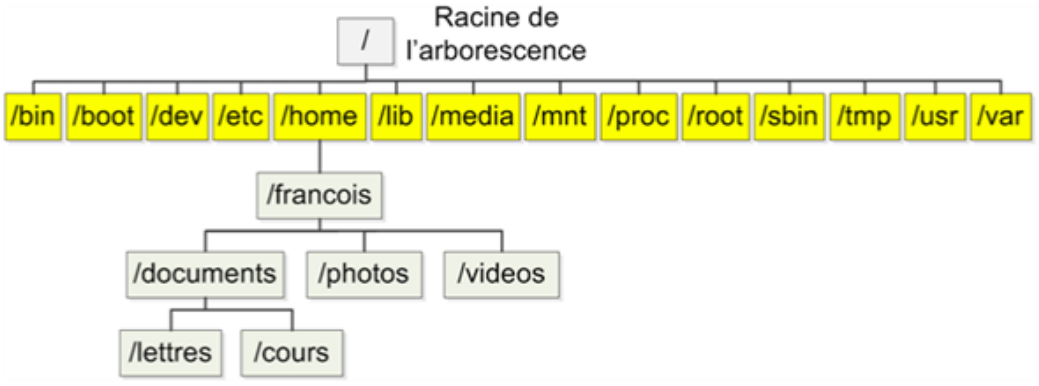
1. **L'image ISO :** C'est le disque d'installation virtuel (téléchargé gratuitement).
2. **Partitionnement :** Dans la VM, choisissez "Effacer le disque et installer Linux". Ne paniquez pas ! Cela n'efface que le disque *virtuel* de 20 Go, pas votre vrai disque dur.
3. **Création de compte :** Vous définirez un utilisateur (ex: `etudiant`) et un mot de passe. **Ne perdez pas ce mot de passe**, il est requis pour toutes les tâches administratives.

Guide d'installation de Linux Mint sur une Vm (VirtualBox)

1.2 Arborescence et navigation

Oubliez ce que vous savez de Windows. Ici, les lecteurs `C:`, `D:` ou `E:` n'existent pas. Sous Linux, tout part d'un point unique : la **Racine** (Root).

1. L'arbre Linux



- `/` **(La racine) :** Le début de tout. Seul l'administrateur peut écrire ici.
- `/home` **(Les maisons) :** Contient les dossiers personnels des utilisateurs.
  - `/home/françois` : Les dossiers de François.
  - `/home/françois/documents` : Les documents de François.
- `/etc` **(Etcaetera) :** Contient tous les fichiers de configuration du système.
- `/var` **(Variable) :** Pour les fichiers qui changent souvent (journaux système, sites web, emails).

2. Parler au Terminal

Se repérer : pwd et ls

```
etudiant@linux:~$ pwd
/home/etudiant
```

`pwd` (Print Working Directory) répond à la question "Où suis-je ?".

```
etudiant@linux:~$ ls -la
```

`ls` (List) affiche le contenu du dossier. L'option `-a` affiche les fichiers cachés.

Se déplacer : cd (Change Directory)

Pour comprendre le déplacement, il faut d'abord visualiser le territoire. Voici à quoi ressemble votre système de fichiers :

```
/ (Racine)
├── etc
├── var
└── home
    └── etudiant  <-- VOUS ÊTES ICI
        ├── Images
        │   └── vacances.jpg
        └── Documents
            └── projet.txt
```

ANALOGIE : LE GPS VS LA BOUSSOLE

Imaginez que vous êtes dans le dossier **etudiant** et vous voulez aller dans **Images**.

- **Chemin Absolu (Le GPS) :**
  Vous donnez l'adresse postale complète et exacte depuis la racine du pays. C'est long, mais infaillible. Ça commence **toujours** par un slash `/`.
  `cd /home/etudiant/Images`
- **Chemin Relatif (La Boussole) :**
  Vous donnez la direction par rapport à votre position actuelle. C'est rapide, mais ça dépend d'où vous êtes.
  `cd Images` *(Traduction : "Entre dans le dossier Images qui est juste devant moi").*

Les raccourcis de survie

Les administrateurs sont paresseux. Voici les symboles pour aller plus vite :

Symbole	Signification	Exemple
`..`	Le dossier parent (Reculer d'un cran)	`cd ..` (Je sors de *Images* pour revenir à *etudiant*)
`.`	Le dossier actuel (Ici même)	`./script.sh` (Lancer le script qui est ici)
`~`	Votre dossier maison (/home/etudiant)	`cd ~` (Retour à la maison, peu importe où je suis)
`-`	Le dossier précédent	`cd -` (C'est le bouton "Précédent" de la télécommande TV)

L'ASTUCE DU PRO

Vous êtes perdu dans les sous-dossiers ? Tapez simplement `cd` et appuyez sur Entrée.
Par défaut, cela vous ramène directement à la maison (`~`).

1.3 Gestion des paquets et édition

1. La fin des fichiers .exe (Les dépôts)

Sous Windows, quand vous voulez un logiciel, vous ouvrez le navigateur, vous cherchez un site (parfois douteux), vous téléchargez un `setup.exe` et vous cliquez sur "Suivant, Suivant, Suivant".

Sous Linux, cette méthode est archaïque. Nous utilisons des **Gestionnaires de Paquets** connectés à des **Dépôts** (Repositories). C'est une immense bibliothèque centralisée, validée et sécurisée par les créateurs de votre distribution.

2. APT : Le standard Debian/Mint

Sur Linux Mint, le chef d'orchestre s'appelle **APT**. Voici les 3 commandes que vous taperez tous les jours :

ANALOGIE : LE MENU DU RESTAURANT

Beaucoup de débutants confondent `update` et `upgrade`. Imaginez que vous êtes au restaurant :

- `sudo apt update` : **Vous demandez la carte du jour.**
  "*Qu'est-ce qu'il y a de nouveau ?*" (Cela ne vous donne rien à manger, cela met juste à jour votre connaissance des plats disponibles).
- `sudo apt upgrade` : **Vous commandez les plats améliorés.**
  "*Remplacez mes vieux plats par les nouvelles versions de la carte.*" (C'est là que l'installation réelle se fait).

```
# 1. ACTUALISER (Toujours faire ceci en premier !)
# Télécharge la liste des dernières versions disponibles.
sudo apt update

# 2. INSTALLER
# Télécharge et installe le logiciel (et ses dépendances).
sudo apt install vim

# 3. SUPPRIMER
# Désinstalle le logiciel.
sudo apt remove vim
```

CULTURE GÉNÉRALE : LE COUSIN DNF

Linux est une grande famille. Mint et Ubuntu utilisent **APT** (format `.deb`). Mais dans le monde professionnel, vous croiserez souvent la famille Red Hat (Fedora, RHEL, AlmaLinux) qui utilise **DNF** (format `.rpm`).

C'est exactement la même logique, seul le mot change :

Action	Mint / Ubuntu	Red Hat / Fedora
Mettre à jour la liste	`apt update`	`dnf check-update`
Installer	`apt install`	`dnf install`

3. L'éditeur VI / VIM

Maintenant que vous savez installer des logiciels, installons **VIM**. C'est l'outil indispensable pour modifier les fichiers de configuration directement dans le terminal.

VIM est un éditeur modal avec deux modes principaux :

1. **Mode Commande (Défaut) :** Le clavier sert à donner des ordres (`dd` pour supprimer, `x` pour couper).
2. **Mode Insertion (Édition) :** Appuyez sur `i` pour écrire. `Esc` pour sortir.

KIT DE SURVIE VIM

Pour sortir : Appuyez sur `Esc`, puis tapez `:wq` (Sauvegarder et Quitter) ou `:q!` (Quitter sans sauver).

CHAPITRE 2 : L'Expansion des accolades

Si la ligne de commande a une magie, c'est bien celle-ci. L'expansion des accolades `{}` permet de générer des listes de texte arbitraires. C'est un outil de productivité redoutable pour éviter de taper des choses répétitives.

2.1. Le concept : Un générateur de texte

L'expansion des accolades n'est pas une commande. C'est un mécanisme du Shell qui se produit **avant** même que la commande ne soit exécutée. Bash voit les accolades, génère tous les mots possibles, et les donne à la commande.

```
# Ce que vous tapez :
echo Je mange une {pomme,banane,poire}

# Ce que Bash "voit" et exécute réellement :
echo Je mange une pomme Je mange une banane Je mange une poire
```

2.2. Les Listes (La virgule)

La forme la plus simple utilise des virgules pour séparer les éléments.

```
# Créer trois dossiers d'un coup
mkdir {dossier1,dossier2,dossier3}

# Créer une arborescence de projet complète en une ligne
mkdir -p Projet/{images,css,js,docs}
```

INTERDIT AUX ESPACES !

C'est l'erreur numéro 1. Il ne faut **JAMAIS** mettre d'espace à l'intérieur des accolades.

- ❌ `{pomme, poire}` : Bash ne le reconnaît pas à cause de l'espace. Il affichera "{pomme," et "poire}".
- ✅ `{pomme,poire}` : Bash génère "pomme" "poire".

2.3. Les séquences (Les deux points)

Bash sait compter. Si vous lui donnez un début et une fin séparés par deux points `..`, il remplit le vide.

1. Séquences numériques

```
echo {1..10}
# Résultat : 1 2 3 4 5 6 7 8 9 10

# Compte à rebours
echo {5..1}
# Résultat : 5 4 3 2 1
```

2. Séquences alphabétiques

```
echo {a..e}
# Résultat : a b c d e
```

ASTUCE : LE REMPLISSAGE PAR ZÉRO

Pour que vos fichiers soient bien triés, utilisez un zéro devant les chiffres.
`echo {01..10}` générera `01 02 03 ... 09 10`.

3. Le "Pas" (Sauter des nombres)

Depuis la version 4 de Bash, on peut ajouter un troisième argument pour définir le saut (step).

```
# Compter de 0 à 10 par tranches de 2 (Nombres pairs)
echo {0..10..2}
# Résultat : 0 2 4 6 8 10
```

2.4. Combinaisons et "Nesting"

C'est ici que ça devient puissant. On peut "coller" du texte autour des accolades.

1. Préfixe et suffixe

```
# Générer des noms de fichiers images
echo img_{01..03}.jpg
# Résultat : img_01.jpg img_02.jpg img_03.jpg
```

2. Le produit cartésien (Multiplier les listes)

Si vous mettez deux accolades côte à côte, Bash combine tout avec tout.

```
echo {a,b}{1,2}
# Résultat : a1 a2 b1 b2

# Exemple concret : Créer une structure d'années et de mois
mkdir 202{4,5,6}/{jan,fev,mar}
# Crée :
# 2024/jan 2024/fev 2024/mar
# 2025/jan 2025/fev 2025/mar...
```

2.5. L'astuce des pros : La sauvegarde rapide

Les administrateurs système utilisent souvent une accolade vide pour faire une copie de sauvegarde d'un fichier sans retaper son nom.

```
# Au lieu de taper :
cp /etc/ssh/sshd_config /etc/ssh/sshd_config.bak

# Tapez ceci :
cp /etc/ssh/sshd_config{,.bak}
```

Explication : `{,.bak}` se déploie en "rien" (vide) puis ".bak".
Bash transforme donc la commande en : `cp fichier fichier.bak`.

Exercices rapides

- Créez 100 fichiers nommés `file1.txt` à `file100.txt` en une commande.
- Affichez toutes les lettres de l'alphabet.
- Supprimez uniquement les fichiers `image_1.jpg`, `image_3.jpg` et `image_5.jpg` en une commande.

CHAPITRE 3 : Administration locale

Maintenant que vous savez vous déplacer, il est temps de devenir propriétaire. Nous allons gérer les utilisateurs, les permissions et les processus.

3.1 Gestion des utilisateurs et groupes

Linux est un système multi-utilisateurs strict. Pour le noyau (le cœur du système), votre nom n'est pas "Alfred" ou "François". Vous êtes un numéro.

1. L'identité numérique (UID & GID)

Chaque utilisateur possède un **UID** (User ID) et appartient à un groupe principal identifié par un **GID** (Group ID). C'est comme votre numéro d'assurance sociale ou de matricule étudiant.

- **UID 0 (Root) :** L'administrateur suprême. C'est le "Dieu" de la machine. Il a tous les droits, tout le temps.
- **UID 1 à 999 (Système) :** Ce sont les "ouvriers invisibles". Ces comptes servent à faire tourner les services (comme le serveur web `www-data` ou le mail) sans donner les droits de root. Ils ne peuvent généralement pas se connecter.
- **UID 1000+ (Les Humains) :** Les utilisateurs réels. Le premier compte créé (le vôtre) a souvent l'UID 1000. Le suivant aura 1001, etc.

2. Les fichiers de l'annuaire

Contrairement à Windows qui cache ces infos, Linux les stocke dans des fichiers texte lisibles (au moins en partie).

> **ANALOGIE : L'ANNUAIRE ET LE COFFRE-FORT**
>
> - `/etc/passwd` est comme l'annuaire de l'immeuble dans le hall d'entrée. Tout le monde peut le consulter pour savoir qui habite où (quel est son dossier home, son UID).
> - `/etc/shadow` est le coffre-fort de la banque. Il contient les signatures des mots de passe. Seul le directeur (Root) a la clé pour l'ouvrir.

L'annuaire public : /etc/passwd

Si vous ouvrez ce fichier (`cat /etc/passwd`), vous verrez des lignes comme celle-ci :

```
etudiant:x:1000:1000:Etudiant,,,:/home/etudiant:/bin/bash
```

Déchiffrons cette ligne (les champs sont séparés par `:`) :

1. **etudiant :** Le nom de connexion (Login).
2. **x :** Le mot de passe ? Non ! Le `x` signifie "Le mot de passe est caché dans */etc/shadow*".
3. **1000 :** L'UID (Numéro d'utilisateur).
4. **1000 :** Le GID (Numéro de son groupe principal).
5. **Etudiant,,, :** Le commentaire (Nom complet, bureau, tél...).
6. **/home/etudiant :** Son dossier personnel.
7. **/bin/bash :** Son interpréteur de commandes (Shell).

Le coffre-fort : /etc/shadow

Ce fichier contient les hashs (mots de passe chiffrés). Si un utilisateur a un `!` ou un `*` ici, son compte est désactivé.

Les groupes : /etc/group

Définit les groupes secondaires (ex: développeurs, sudo, docker) et qui en fait partie.

3. Commandes de gestion : Le cycle de vie

Gérer un utilisateur se fait en trois temps : on le crée, on définit son secret, et on lui donne des pouvoirs.

Création (useradd)

La commande de base est `useradd`. Attention, sans option, elle crée un utilisateur "SDF" (sans dossier personnel) !

```
# L'option -m (Make home) est OBLIGATOIRE pour créer /home/alfred
# L'option -s (Shell) définit le programme par défaut (/bin/bash)
sudo useradd -m -s /bin/bash alfred
```

Sécurisation (passwd)

Un nouvel utilisateur est verrouillé par défaut (pas de mot de passe). Il faut l'activer.

LE "BLIND TYPE" (FRAPPE AVEUGLE)

Quand vous tapez la commande ci-dessous, Linux vous demandera le mot de passe.
Rien ne s'affichera à l'écran (ni étoiles, ni points) pendant que vous tapez le mot de passe. C'est une sécurité. Tapez à l'aveugle et faites Entrée.

```
# Syntaxe : passwd [Nom_Utilisateur]
sudo passwd alfred
```

Modification des groupes (usermod)

Alfred est un simple utilisateur. Pour qu'il devienne administrateur, il faut l'ajouter au groupe `sudo` (ou `wheel` sur d'autres Linux).

DANGER : L'OPTION -AG VS -G

C'est l'erreur classique qui supprime des accès :

- ❌ `usermod -G sudo alfred` : "Alfred fait **UNIQUEMENT** partie du groupe sudo." (Cela l'enlève de tous ses autres groupes !).
- ✅ `usermod -aG sudo alfred` : "**A**joute (Append) le groupe sudo à la liste des groupes d'Alfred."

```
# Ajouter Alfred au groupe des admins (sudo)
sudo usermod -aG sudo alfred
```

Suppression (userdel)

Alfred quitte l'entreprise ? Il faut supprimer son compte ET ses fichiers.

```
# L'option -r (Remove) supprime aussi le dossier /home/alfred
sudo userdel -r alfred
```

3.2 Permissions et sécurité

1. Le concept UGO et la trinité RWX

Chaque fichier a des droits pour : **U**ser (Propriétaire), **G**roup (Groupe), **O**thers (Autres).

2. Les Maths de l'admin : La notation octale

Au lieu d'écrire `rwxr-xr--`, les administrateurs utilisent des chiffres. C'est plus rapide et moins sujet aux fautes de frappe. Mais d'où viennent les chiffres **4, 2 et 1** ?

L'origine binaire (Sous le capot)

Imaginez que chaque groupe de permissions est composé de **3 interrupteurs** (Bits). Si le bit est à 1, l'option est activée. Si c'est 0, c'est désactivé.

Position	Permission	Binaire	Valeur Décimale
1 (Gauche)	**R**ead (Lecture)	1 0 0	**4**
2 (Milieu)	**W**rite (Écriture)	0 1 0	**2**
3 (Droite)	e**X**ecute (Exécution)	0 0 1	**1**

Pour obtenir le droit final, on additionne simplement les valeurs actives :

7 : TOUT AUTORISÉ

Lecture (4) + Écriture (2) + Exéc (1)
Total = 7

5 : LIRE ET LANCER

Lecture (4) + Rien (0) + Exéc (1)
Total = 5

6 : LIRE ET MODIFIER

Lecture (4) + Écriture (2) + Rien (0)
Total = 6

EXEMPLE CONCRET : CHMOD 750

Si je tape `chmod 750 document.txt`, je définis trois règles d'un coup :

- **7 (Propriétaire) :** 4+2+1. Je fais ce que je veux (Lecture, Écriture, Exécution).
- **5 (Groupe) :** 4+0+1. Mes collègues peuvent lire le fichier et l'exécuter (script), mais pas le modifier.
- **0 (Les Autres) :** Accès totalement interdit. Le fichier n'existe même pas pour eux.

3. Changer de propriétaire (chown)

Le nom de la commande vient de **CH**ange **OWN**er. Elle est indispensable quand vous copiez des fichiers en tant qu'administrateur (root) et que vous voulez redonner les droits à un utilisateur normal.

LA SYNTAXE : LES DEUX POINTS

La force de `chown` est qu'il peut changer le propriétaire ET le groupe en même temps en les séparant par deux points `:`.
Syntaxe : `chown [NouveauProprio]:[NouveauGroupe] [Fichier]`

```
# Cas 1 : Changer seulement le propriétaire
# "Ce rapport appartient maintenant à Robert"
sudo chown robert rapport.txt

# Cas 2 : Changer le propriétaire ET le groupe
# "Ce fichier appartient à Robert et à l'équipe Developers"
sudo chown robert:developers script.py

# Cas 3 : La Récursivité (-R)
# "Ce dossier, et TOUT ce qu'il contient (sous-dossiers, fichiers), est à Robert"
sudo chown -R robert:developers /var/www/mon_site
```

ATTENTION AU -R

L'option `-R` (Majuscule) signifie **Récursif**.
Soyez très prudent ! Si vous faites `chown -R robert /`, vous donnez tout le système d'exploitation à Robert, ce qui brisera Linux instantanément. Vérifiez toujours le dossier cible.

3.3 Gestion des processus

Chaque programme lancé sur Linux devient un **Processus**. Pour le système, ce processus est identifié par un numéro unique : le **PID** (Process ID). Si un programme plante, Linux ne plante pas : il suffit de tuer le PID coupable.

1. Surveillance : L'art de savoir ce qui tourne

La méthode visuelle : pstree

Les processus ne naissent pas de nulle part. Ils sont lancés par des parents. Pour voir cette hiérarchie (qui a lancé qui), utilisez l'arbre :

```
pstree
```

Vous verrez que **systemd** (PID 1) est l'ancêtre de tous les processus.

La méthode statique : ps (Process Status)

C'est la commande standard. Elle a deux syntaxes populaires (Unix vs BSD) :

- `ps aux` (Style BSD, le plus courant) :
  - **a** : Tous les utilisateurs.
  - **u** : Affiche le propriétaire et la consommation RAM/CPU.
  - **x** : Affiche aussi les processus sans terminal (services).
- `ps -ef` (Style Standard) : Donne un affichage légèrement différent, souvent utilisé pour voir le PPID (Parent PID).

```
# Chercher un processus spécifique
ps aux | grep firefox
```

La méthode temps réel : htop

C'est le "Gestionnaire des tâches" interactif. (`F10` ou `q` pour quitter).

2. Arrêt des processus (Le nettoyeur)

Quand un programme ne répond plus, on envoie un **Signal**.

KILL VS KILLALL

- **kill [PID] :** Le sniper.
  Il tue un processus précis via son numéro.
  Ex : `kill 1234`
- **killall [Nom] :** La grenade.
  Il tue **tous** les processus qui portent ce nom.
  Ex : `killall firefox` (Ferme toutes les fenêtres Firefox d'un coup).

Les niveaux de force :

- `kill -15` (SIGTERM) : Demande polie de fermeture (Défaut).
- `kill -9` (SIGKILL) : Arrêt forcé immédiat (Dernier recours).

3. Processus d'arrière-plan (multitâche)

Par défaut, une commande longue bloque votre terminal. Pour récupérer la main, on utilise l'arrière-plan (Background).

EXERCICE PRATIQUE : LA SIESTE

Pour tester, nous allons utiliser la commande `sleep` qui ne fait rien d'autre qu'attendre.

Scénario A : Lancer directement en fond (&)

Ajoutez l'esperluette `&` à la fin de la commande.

```
# Je demande au terminal de dormir 1000 secondes en arrière-plan
sleep 1000 &
# Le terminal me répond : [1] 4567 (Numéro du Job + PID)
```

Scénario B : J'ai oublié le & (Ctrl+Z, bg, fg)

Vous avez lancé une commande qui bloque tout. Pas de panique.

1. **Ctrl + Z :** Met le processus en **Pause** (Stopped).
2. `bg` (BackGround) : Relance le processus, mais en arrière-plan.
3. `jobs` : Affiche la liste de vos tâches de fond.
4. `fg` (ForeGround) : Ramène le processus devant vous (le bloque à nouveau).

```
# Exemple complet
sleep 500       # Zut, je suis bloqué !
(Ctrl+Z)        # [1]+  Stopped    sleep 500
bg              # [1]+ sleep 500 & (Il tourne maintenant en fond)
jobs            # Affiche la liste
fg 1            # Je le ramène au premier plan
```

4. Persistance (Survivre à la fermeture)

Si vous fermez votre terminal (ou si SSH coupe), tous vos processus enfants (y compris ceux en arrière-plan) meurent instantanément. C'est la chaîne de commandement.

La solution préventive : nohup

`nohup` (No Hang Up) détache le processus du terminal. Il continuera de tourner même si vous vous déconnectez.

```
# Lancer une sauvegarde qui prendra toute la nuit
nohup cp -R /source /dest &

# La sortie (textes) sera écrite dans un fichier 'nohup.out' par défaut.
```

La solution curative : disown

Vous avez lancé une tâche en `bg` mais vous avez oublié `nohup` ?

```
sleep 1000 &
disown -h %1    # Détache le job n°1 du terminal. Vous pouvez fermer la fenêtre.
```

CHAPITRE 4 : Flux, filtres, stockage et réseau

Une machine isolée ne sert à rien. Connectons-la au monde et gérons son stockage.

4.1 Redirections, flux et filtres

La philosophie de Linux est de créer de petits outils qui font une seule chose parfaitement. La puissance vient de notre capacité à connecter ces outils entre eux.

1. Les canaux invisibles (flux)

Chaque fois que vous lancez une commande, trois canaux s'ouvrent automatiquement :

- **Entrée Standard (Stdin - 0) :** Ce qui rentre (votre Clavier).
- **Sortie Standard (Stdout - 1) :** Le résultat normal (l'Écran).
- **Erreur Standard (Stderr - 2) :** Les messages d'erreur (l'Écran aussi).

2. Les redirections : Détourner la rivière

Par défaut, le résultat s'affiche à l'écran. Une redirection permet d'envoyer ce texte **dans un fichier**.

EXEMPLE : LA LISTE DE COURSES

Utilisons la commande `echo` (qui répète ce qu'on dit) pour comprendre la différence entre **Créer (>)** et **Ajouter (>>)**.

1. **Le destructeur (>) :**
   `echo "Pommes" > liste.txt`
   Résultat : Crée le fichier. S'il existait déjà, **il est vidé et remplacé** par "Pommes".

2. **Le constructeur (>>) :**
   `echo "Bananes" >> liste.txt`
   Résultat : Ajoute "Bananes" à la suite, sans effacer les Pommes.

```
# Pour vérifier le contenu du fichier
cat liste.txt
# Affiche :
# Pommes
# Bananes
```

3. Le tube (Pipe |)

C'est l'outil le plus puissant de la ligne de commande. Le caractère `|` (Alt Gr + 6 ou Shift + \) permet de connecter deux commandes.

ANALOGIE : LA CHAÎNE D'USINE

Le Pipe connecte la **Sortie** de la commande A directement sur l'**Entrée** de la commande B. Pas de fichier intermédiaire, tout se passe en mémoire.

`Machine A (Produit) | Machine B (Emballe) | Machine C (Étiquette)`

4. La boîte à outils (Les filtres)

Le Pipe est inutile sans outils pour traiter les données. Voici les filtres indispensables :

Commande	Action	Exemple
`grep`	Chercher un mot (Filtrer les lignes)	`cat fichier | grep "erreur"`
`sort`	Trier par ordre alphabétique	`cat noms.txt | sort`
`uniq`	Supprimer les doublons (doit être trié avant)	`sort noms.txt | uniq`
`wc -l`	Compter le nombre de lignes (Word Count)	`cat liste.txt | wc -l`
`head / tail`	Voir le début / la fin	`tail -n 5 journal.log`

Exemple combiné (Le "Combo")

Objectif : Savoir combien d'utilisateurs peuvent utiliser le shell "bash".

```
cat /etc/passwd | grep "/bin/bash" | wc -l
```

Traduction : "Lis l'annuaire" -> "Garde seulement les lignes contenant bash" -> "Compte les lignes restantes".

4.2 Disques et partitionnement

Contrairement à Windows qui assigne une lettre (D:, E:) à chaque clé USB, Linux intègre tout dans son arborescence unique. Mais avant d'utiliser un disque, il faut le préparer.

1. Le matériel : Tout est fichier

Vos périphériques sont listés dans le dossier `/dev` (Devices). La nomenclature est logique :

- `/dev/sda` : **S**CSI **D**isk **A** (1er disque physique).
- `/dev/sdb` : 2e disque physique.
- `/dev/sda1` : 1ère partition du 1er disque.

Commande vitale : `lsblk` (List Block Devices) permet de voir l'arbre de vos disques pour ne pas se tromper de cible.

2. Le cycle de vie d'un disque

ANALOGIE : L'AMÉNAGEMENT DE L'ENTREPÔT

Imaginez que vous achetez un immense entrepôt vide (Le Disque Dur). Vous ne pouvez pas encore y stocker des dossiers.

1. **Partitionner (Les murs) :**
   Vous construisez des murs pour diviser l'entrepôt en pièces distinctes.
   Outil : `fdisk`
2. **Formater (Les étagères) :**
   Vous installez des étagères et un système de classement. Sans cela, les dossiers seraient jetés par terre en vrac. C'est le **Système de Fichiers** (Filesystem).
   Outil : `mkfs` (Make FileSystem)
3. **Monter (La porte) :**
   Vous installez une porte avec une étiquette (ex: "Archives"). Si la porte est fermée, la pièce existe, mais on ne peut pas y entrer.
   Outil : `mount`

Étape 1 : Partitionner

```
# Ouvre l'utilitaire interactif sur le disque B
sudo fdisk /dev/sdb
# (Tapez 'n' pour New, 'p' pour Primary, 'w' pour Write/Save)
```

Étape 2 : Formater

Le standard sous Linux est **ext4** (comme NTFS pour Windows).

```
# Formate la partition 1 du disque B
sudo mkfs.ext4 /dev/sdb1
```

Étape 3 : Monter (temporaire)

Linux ne "voit" pas le disque tant qu'il n'est pas attaché à un dossier vide (le point de montage).

```
# 1. Créer le point d'ancrage
sudo mkdir /mnt/data

# 2. Accrocher le disque
sudo mount /dev/sdb1 /mnt/data
```

3. La persistance (fstab)

La commande `mount` est temporaire. Au redémarrage, tout disparaît. Pour rendre le montage permanent, il faut éditer le fichier `/etc/fstab`.

On utilise l'**UUID** (Universally Unique IDentifier), une empreinte digitale unique du disque, plutôt que son nom `/dev/sdb1` (qui peut changer si on débranche des câbles).

```
# Trouver l'UUID de votre disque
sudo blkid
```

ZONE DE DANGER FSTAB

Le fichier `/etc/fstab` est lu au démarrage du noyau. S'il contient une faute de frappe, **le système ne démarrera pas** (Mode Urgence).

Règle d'or : Après avoir modifié ce fichier, lancez TOUJOURS :
`sudo mount -a`
Si cette commande ne dit rien, c'est gagné. Si elle affiche une erreur, corrigez le fichier **avant** d'éteindre l'ordinateur !

4.3 Réseau et services

Un serveur Linux sans réseau est une île déserte. Dans cette section, nous allons apprendre à identifier les cartes réseaux, configurer les adresses IP et gérer les programmes qui écoutent le monde extérieur.

1. Interfaces et connexions (état des lieux)

La vieille commande `ifconfig` est obsolète (même si elle fonctionne encore). Le standard moderne est la suite **iproute2**.

Identifier ses cartes (ip link)

Avant de configurer, il faut connaître le nom de son matériel.

```
ip link show
```

- **lo (Loopback) :** La boucle locale (127.0.0.1). C'est l'ordinateur qui se parle à lui-même. Ne jamais y toucher !
- **eth0 / enp3s0 :** Votre carte Ethernet (câble).
- **wlan0 / wlp2s0 :** Votre carte Wi-Fi.

Voir les adresses IP (ip addr)

```
ip addr show
```

Cherchez la ligne `inet` pour voir votre IPv4 (ex: 192.168.1.50).

Activer / Désactiver le réseau

Parfois, il faut "débrancher le câble" virtuellement pour réinitialiser une connexion.

```
# Éteindre l'interface
sudo ip link set enp3s0 down

# Rallumer l'interface
sudo ip link set enp3s0 up
```

2. Les fichiers texte importants

Sous Linux, la configuration réseau se cache dans des fichiers texte. Voici les deux plus importants à connaître pour le dépannage :

ANALOGIE : LE CARNET D'ADRESSES

- **/etc/hosts (Contacts favoris) :** C'est votre petit carnet personnel. Vous y dites : "L'adresse 192.168.1.10 s'appelle 'serveur-web'". Linux regarde ici *avant* d'aller sur Internet.
- **/etc/resolv.conf (L'opérateur) :** C'est ici qu'on définit les serveurs DNS (ceux qui traduisent google.com en adresse IP).

3. Commandes de gestion (NetworkManager)

Sur Linux Mint et la plupart des postes de travail, c'est **NetworkManager** qui gère le réseau. L'outil en ligne de commande est `nmcli`.

```
# Voir l'état général
nmcli general status

# Lister les connexions disponibles
nmcli connection show

# Activer une connexion spécifique
nmcli connection up "Connexion filaire 1"
```

ASTUCE : L'OUTIL VISUEL

Si vous êtes en console pure et que vous détestez taper des commandes réseau compliquées, tapez :
`nmtui` (NetworkManager Text User Interface).
C'est une interface graphique en texte (avec des menus bleus) très facile à utiliser.

4. Le chef d'orchestre : systemd

Un serveur Web (Apache) ou un serveur de fichiers qui tourne en arrière-plan s'appelle un **Service** (ou Démon). Le programme qui les gère tous s'appelle **Systemd**.

ANALOGIE : LE CONCIERGE D'HÔTEL

Imaginez Systemd comme le concierge. Vous ne parlez pas aux employés (Apache, SSH, MySQL), vous parlez au concierge.

- "Concierge, réveille le cuisinier !" (start)
- "Concierge, assure-toi que le cuisinier soit là tous les matins à l'ouverture." (enable)

```
# 1. DÉMARRER (Immédiat)
sudo systemctl start apache2

# 2. ACTIVER (Au prochain redémarrage)
sudo systemctl enable apache2

# 3. STATUT (Le plus important pour le dépannage !)
sudo systemctl status apache2
```

5. Sécurité et accès (SSH & UFW)

SSH : Le cordon ombilical

SSH (Secure Shell) permet de se connecter à distance de manière chiffrée.

```
ssh nom_utilisateur@adresse_ip_du_serveur
```

UFW : Le pare-feu simpliste

Par défaut, on devrait tout bloquer et ouvrir seulement le nécessaire. UFW (Uncomplicated Firewall) simplifie la gestion.

```
sudo ufw enable        # Activer le pare-feu
sudo ufw allow ssh     # Laisser entrer le SSH (Port 22)
sudo ufw allow http    # Laisser entrer le Web (Port 80)
sudo ufw status        # Vérifier les règles
```

CHAPITRE 5 : Scripting et automatisation

5.1 Introduction au scripting

Un script Bash est un chef d'orchestre. Il lance des commandes, leur donne des paramètres et attend qu'elles finissent. Parfois, il doit aussi faire des pauses.

1. L'anatomie d'un script

Un script Bash n'est rien de magique. C'est simplement un fichier texte contenant une liste de commandes que vous auriez pu taper à la main, l'une après l'autre. L'avantage ? Vous l'écrivez une fois, vous l'exécutez mille fois.

Créez un fichier `mon_script.sh`. Voici de quoi il a l'air :

```
#!/bin/bash

# Ceci est un commentaire (ignoré par l'ordinateur)
echo "Démarrage du script..."
ls -la
```

Le Shebang (#!)

La toute première ligne est cruciale. Elle s'appelle le **Shebang** (Hash-Bang). Elle indique au système quel langage utiliser pour lire le fichier.

- `#!/bin/bash` : "Utilise le langage Bash standard".
- `#!/usr/bin/python3` : "Utilise le langage Python".

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

Les commentaires (#)

Les commentaires sont précédés du symbole `#`.

La commande `echo` permet d'afficher des résultats à l'écran.

```
#!/bin/bash
# J'affiche que le script démarre
echo "Début du script"
```

2. Le rituel d'exécution

Pour lancer votre chef-d'œuvre, il y a deux étapes obligatoires.

Étape 1 : Donner la permission (chmod)

Par sécurité, un fichier texte ne peut pas être exécuté. Il faut le transformer en programme.

```
chmod +x mon_script.sh
```

Étape 2 : Lancer le script (Le ./)

Si vous tapez juste `mon_script.sh`, Linux dira "Commande introuvable". Pourquoi ? Parce que Linux ne cherche pas dans le dossier actuel par sécurité.

Il faut lui dire : "Lance le script qui est **ici** (`.`) dans ce dossier (`/`)".

```
./mon_script.sh
```

3. Les variables : des boîtes à mémoire

En programmation, on stocke l'information dans des variables. En Bash, la syntaxe est **très stricte**.

LA RÈGLE D'OR DES ESPACES

Il ne faut JAMAIS mettre d'espaces autour du signe égal !

- ❌ `NOM = "Toto"` : L'ordinateur cherche la commande "NOM" (qui n'existe pas).
- ✅ `NOM="Toto"` : L'ordinateur stocke "Toto" dans la variable NOM.

Pour **remplir** la boîte, on utilise le nom seul. Pour **lire** le contenu, on ajoute un dollar `$`.

```
#!/bin/bash

COURS="Linux"            # Je remplis
echo "J'aime $COURS"     # Je lis (Affiche : J'aime Linux)
```

4. Rendre le script intelligent (Interactivité)

Un script peut poser des questions à l'utilisateur ou recevoir des paramètres au lancement.

Méthode A : Poser une question (read)

```
read -p "Quel est ton nom ? " UTILISATEUR
echo "Bonjour $UTILISATEUR, bienvenue dans le système."
```

Méthode B : Les arguments ($1, $2...)

On peut donner des infos directement au lancement : `./mon_script.sh Toto`

- `$#` : Le nombre de paramètres.
- `$*` : La liste des paramètres.
- `$0` : Le nom du script lui-même.
- `$1` : Le premier mot après le script (Toto).
- `$2` : Le deuxième mot, etc.

Exemple concret : Le script `infos_script.sh`

```
#!/bin/bash

echo "Nom du script: $0"
echo "Nombre d'arguments: $#"
echo "1er argument: $1"
echo "2e argument: $2"
echo "Tous les arguments (\$*): $*"
```

Si on exécute ce script

```
./infos_script.sh Hello World "Linux Script"
```

Le résultat sera :

```
Nom du script: ./info_script.sh
Nombre d'arguments: 3
1er argument: Hello
2e argument: World
Tous les arguments ($*): Hello World Linux Script
```

5.2 Contrôler le temps : sleep et wait

Parfois, il faut ralentir la cadence ou synchroniser des tâches.

1. La sieste (sleep)

Suspend l'exécution pendant X secondes. Utile pour attendre qu'un service démarre.

```
#!/bin/bash

echo "Démarrage du serveur..."
sleep 5   # Pause de 5 secondes
echo "Serveur prêt."
```

2. La synchronisation (wait)

Si vous lancez des tâches en arrière-plan (avec `&`), votre script risque de se terminer avant elles. La commande `wait` dit au script : "Attends que tes enfants aient fini avant de quitter".

EXEMPLE : LE CHEF DE CHANTIER

```
#!/bin/bash

echo "Je lance la peinture..."
./peindre_mur_nord.sh &   # Tâche de fond 1
./peindre_mur_sud.sh &    # Tâche de fond 2

echo "Je ne peux pas poser le sol tant que la peinture n'est pas finie !"
wait                      # Le script bloque ici tant que les 2 tâches tournent
echo "Peinture finie. Je pose le sol."
```

5.3 Le flux de contrôle (logique)

Comment un script sait-il si une copie a réussi ou échoué ? Grâce à un petit numéro invisible appelé le **Code de Retour**.

1. Le code de retour ($?)

Chaque fois qu'une commande se termine, elle laisse un message au système, stocké dans la variable `$?`.

LA LOGIQUE INVERSÉE DE LINUX

Contrairement à l'intuition humaine (où 1 = 1ère place = Bravo) :

- **0 (Zéro) = SUCCÈS** ("Zéro erreur", tout va bien).
- **1 à 255 = ÉCHEC** (Il y a eu un problème).

```
#!/bin/bash

ls /home
echo $?   # Affiche 0 (Car le dossier existe)

ls /dossier_imaginaire
echo $?   # Affiche 2 (Erreur : Introuvable)
```

2. Les opérateurs de combinaison (&&, ||, ;)

On peut enchaîner des commandes selon le succès ou l'échec de la précédente. C'est la base de l'automatisation.

Opérateur	Nom	Logique	Exemple parlé
;	Point-virgule	**FAIS A PUIS FAIS B** (Peu importe si A plante)	"Fais la vaisselle ; ensuite sors les poubelles."
&&	ET (AND)	**SI A RÉUSSIT, FAIS B** (Arrêt si A échoue)	"Démarre la voiture **ET** roule." *(Si elle ne démarre pas, tu ne roules pas).*
||	OU (OR)	**SI A ÉCHOUE, FAIS B** (Plan de secours)	"Prends le bus **OU** appelle un taxi." *(Si le bus arrive, pas besoin de taxi).*

Exemple pratique

```
#!/bin/bash

# Créer un dossier ET entrer dedans
mkdir projet && cd projet

# Essayer de copier, OU afficher une alerte si ça rate
cp rapport.txt /backup/ || echo "Erreur critique : Sauvegarde échouée !"
```

3. La priorité et le groupement ( )

Comme en mathématiques, on peut utiliser des parenthèses pour grouper des commandes.

```
# Si le dossier existe (A) ET qu'on peut entrer dedans (B)...
# ... ALORS on liste le contenu (C) SINON on pleure (D).
( [ -d "data" ] && cd "data" ) && ls -la || echo "Impossible d'accéder"
```

4. La structure IF / ELSE

Les opérateurs `&&` sont parfaits pour des actions courtes. Pour des blocs complexes, on utilise `if`.

Jusqu'à présent, vos scripts étaient des robots bêtes qui exécutaient des ordres à la suite. Avec la logique conditionnelle, vous leur donnez un cerveau. Ils peuvent dire : "*SI le dossier existe, ALORS je copie, SINON je crée le dossier.*"

La syntaxe

La structure commence par `if` (si) et se termine obligatoirement par `fi` (if à l'envers). Le point-virgule `;` après le crochet fermant est indispensable si vous mettez le `then` sur la même ligne.

```
if [ condition ]; then
    # Commandes si c'est VRAI
    echo "C'est vrai !"
else
    # Commandes si c'est FAUX
    echo "C'est faux !"
fi
```

Exemple

```
if [ -d "/home/etudiant" ]; then
    echo "Dossier trouvé."
else
    echo "Introuvable."
fi
```

LE PIÈGE MORTEL DES ESPACES

En Bash, le crochet `[` n'est pas une décoration, c'est une **commande**. Comme toute commande, elle a besoin d'espaces pour séparer ses arguments.

- ❌ `if [$a="non"]` : ERREUR (Bash cherche la commande `[$a="non"]` qui n'existe pas).
- ✅ `if [ "$a" = "non" ]` : SUCCÈS (Espace après `[` et avant `]`).

Les opérateurs logiques (dans les tests)

On peut combiner plusieurs tests dans un seul `if`.

- `!` : **NON** (Inverse le résultat).
- `-a` : **ET** (And) -> obsolète, préférez `[[ A && B ]]`.
- `-o` : **OU** (Or).

```
# Si le fichier existe ET n'est pas vide (-s)
if [ -f "log.txt" ] && [ -s "log.txt" ]; then
    echo "Le fichier de log contient des données."
fi
```

5. Le "Cheat Sheet" des tests

Bash ne teste pas un fichier comme il teste un nombre. Voici les codes à connaître par cœur :

Type de Test	Code	Signification
📂 **Fichiers**	`-e`	**E**xiste (Fichier ou Dossier)
	`-f`	Est un **F**ichier (Pas un dossier)
	`-d`	Est un **D**ossier (Directory)
🔢 **Nombres**	`-eq`	**Eq**ual (Égal à)
	`-gt`	**G**reater **T**han (Plus grand que)
	`-lt`	**L**ess **T**han (Plus petit que)
🔤 **Texte**	`=`	Est identique à
	`!=`	Est différent de

Exemple concret : Le script de sauvegarde

Créons un script qui vérifie si un dossier de sauvegarde existe avant de copier des fichiers.

```
#!/bin/bash

DOSSIER_BACKUP="/home/etudiant/backup"

# On teste si le dossier (-d) existe
if [ -d "$DOSSIER_BACKUP" ]; then
    echo "Le dossier existe. Copie en cours..."
    cp *.txt "$DOSSIER_BACKUP"
else
    echo "Le dossier n'existe pas. Je le crée pour toi."
    mkdir "$DOSSIER_BACKUP"
    cp *.txt "$DOSSIER_BACKUP"
fi
```

L'INVERSE (!)

On peut inverser une condition avec le point d'exclamation (le "NOT").
`if [ ! -d "/tmp" ]; then` signifie "Si le dossier /tmp **N'EXISTE PAS**".

5.4 Automatisation et boucles

La paresse est la première qualité de l'administrateur. Si vous devez répéter une tâche plus de trois fois, ne le faites pas à la main. Laissez la machine travailler pour vous grâce aux boucles.

1. La boucle FOR (Le traitement par lot)

La boucle `for` est utilisée quand on connaît la liste des éléments à traiter (une liste de fichiers, une liste de prénoms, une série de chiffres).

ANALOGIE : LA CAISSIÈRE

La boucle FOR fonctionne comme un tapis roulant de supermarché.
"POUR chaque article sur le tapis, scanne-le et bip-le."
Quand il n'y a plus d'articles, la boucle s'arrête naturellement.

Exemple : Le renommage de masse

Imaginez que vous avez 100 fichiers `.txt` et que vous voulez leur ajouter l'extension `.bak`.

```
# Syntaxe : for VARIABLE in LISTE; do COMMANDES; done

for fichier in *.txt; do
    echo "Traitement du fichier : $fichier"
    mv "$fichier" "$fichier.bak"
done
```

Explication : La variable `$fichier` va prendre tour à tour le nom de chaque fichier texte trouvé. La commande `mv` sera exécutée 100 fois automatiquement.

2. La boucle WHILE (Le traitement par condition)

La boucle `while` (Tant que) est utilisée quand on ne sait pas combien de temps cela va durer. Elle continue **tant que** la condition est vraie.

ANALOGIE : LE THERMOSTAT

La boucle WHILE fonctionne comme un chauffage.
"TANT QUE la température est inférieure à 20°C, chauffe."
On ne sait pas si ça prendra 5 minutes ou 2 heures. On s'arrête quand la condition change.

Exemple : Le compte à rebours

```
#!/bin/bash
COMPTEUR=5

# Tant que le compteur est plus grand que 0
while [ $COMPTEUR -gt 0 ]; do
    echo "Décollage dans $COMPTEUR..."
    sleep 1             # Attendre 1 seconde
    ((COMPTEUR--))      # Enlever 1 au compteur (Syntaxe mathématique)
done

echo "DÉCOLLAGE !"
```

3. Danger : La boucle infinie

Que se passe-t-il si la condition reste toujours VRAIE ? Le script ne s'arrêtera jamais. C'est souvent une erreur, mais c'est parfois voulu (pour un serveur qui doit tourner 24h/24).

```
# Attention : Ce script ne s'arrêtera jamais tout seul !
while true; do
    echo "Je tourne en rond..."
    sleep 1
done
```

KIT DE SURVIE

Si vous lancez un script et qu'il part en boucle infinie :
Appuyez sur **Ctrl + C** dans le terminal pour tuer le processus immédiatement.

5.5 Planification (Cron)

Le démon **Cron** est le réveil-matin de Linux. Il permet d'exécuter des scripts automatiquement à une heure précise, même si vous n'êtes pas connecté.

1. L'édition (crontab -e)

Chaque utilisateur a son propre agenda. Pour l'ouvrir :

```
crontab -e
```

*(Si on vous demande de choisir un éditeur, choisissez **Nano**, c'est le plus simple).*

La syntaxe des 5 Étoiles

Une ligne Cron se lit de gauche à droite. Voici le diagramme indispensable à mémoriser :

```
.---------------- minute (0 - 59)
|  .------------- heure (0 - 23)
|  |  .---------- jour du mois (1 - 31)
|  |  |  .------- mois (1 - 12)
|  |  |  |  .---- jour de la semaine (0 - 6) (Dimanche=0)
|  |  |  |  |
* * * * * commande à exécuter
```

EXEMPLES FRÉQUENTS

- `0 3 * * *` : "À la minute 0 de la 3ème heure, tous les jours." (Tous les jours à 03h00).
- `0 0 * * 1` : "À minuit, seulement si on est le 1er jour de la semaine." (Tous les lundis matins).
- `*/5 * * * *` : "Toutes les tranches de 5 minutes." (00:05, 00:10, 00:15...).

Le piège mortel : Les chemins absolus

C'est l'erreur qui fait échouer 99% des scripts planifiés. **Cron ne connaît pas vos raccourcis.**

RÈGLE D'OR

Dans un Cron, utilisez TOUJOURS le chemin complet :
❌ `0 3 * * * backup.sh` (Cron ne sait pas où est ce fichier).
✅ `0 3 * * * /home/etudiant/scripts/backup.sh` (Chemin absolu).

Où va la sortie ? (Logs)

Si votre script affiche du texte (echo), Cron ne peut pas vous le montrer car vous dormez. Par défaut, il essaie de vous envoyer un email système.

Pour garder une trace, on utilise une redirection :

```
# Enregistrer le résultat dans un journal
0 3 * * * /home/etudiant/backup.sh >> /home/etudiant/backup.log 2>&1
```

GUIDE TECHNIQUE : Installation de Linux Mint (VirtualBox)

Ce guide détaille pas à pas la création d'une machine virtuelle (VM) pour le cours. Nous utiliserons **VirtualBox** comme hyperviseur et **Linux Mint (Édition Cinnamon)** comme système invité.

1. Prérequis et téléchargements

- **VirtualBox :** Téléchargez et installez la dernière version depuis virtualbox.org.
- **Extension Pack :** Sur la même page, téléchargez le "Oracle VM VirtualBox Extension Pack" et installez-le (double-cliquez dessus).
- **Image ISO :** Téléchargez l'ISO de Linux Mint (Édition Cinnamon 64-bit) depuis linuxmint.com.

POINT CRITIQUE : VIRTUALISATION BIOS

Avant de commencer, assurez-vous que la virtualisation matérielle est activée dans le BIOS de votre ordinateur physique. Cherchez une option nommée **Intel VT-x** ou **AMD-V**. Sans cela, VirtualBox ne fonctionnera pas ou sera très lent.

2. Création du "squelette" de la VM

Nous allons d'abord assembler le matériel virtuel.

1. Ouvrez VirtualBox et cliquez sur le bouton bleu **Nouvelle**.
2. **Nom et Système :**
   - Nom : `Linux Mint` (ou votre Nom).
   - Dossier : Laissez par défaut.
   - Image ISO : Sélectionnez le fichier `.iso` téléchargé précédemment.
   - *Astuce : Si une case "Skip Unattended Installation" apparaît, cochez-la pour avoir le contrôle total.*
3. **Matériel (Hardware) :**
   - **Mémoire vive (RAM) :** 4096 Mo (4 Go) recommandé. Ne dépassez jamais 50% de la RAM de votre vrai PC (Hôte).
   - **Processeurs (CPU) :** 2 CPU recommandés.
4. **Disque Dur :**
   - Créer un disque dur virtuel maintenant.
   - Taille : **25 Go minimum** (30 Go est confortable).
5. Cliquez sur **Finish**.

3. Configuration optimisée

Avant de démarrer, ajustons quelques réglages pour la fluidité.

1. Sélectionnez votre VM et cliquez sur **Configuration** (Roue dentée).
2. Allez dans **Affichage** (Display) :
   - Mémoire vidéo : Poussez le curseur au maximum (128 Mo).
   - Contrôleur graphique : Laissez sur **VMSVGA**.
   - Activer l'accélération 3D : **Cocher** (Optionnel mais recommandé).
3. Allez dans **Système** > Onglet **Processeur** :
   - Vérifiez que "Activer PAE/NX" est coché.
4. Cliquez sur **OK**.

4. Installation du système

C'est le moment de vérité. Allumons la machine.

1. Cliquez sur **Démarrer** (Flèche verte).
2. La VM va démarrer sur l'ISO. Vous verrez un menu, appuyez sur **Entrée** pour "Start Linux Mint".
3. Vous arrivez sur un bureau "Live". Ce n'est pas encore installé !
4. Double-cliquez sur l'icône de CD **Install Linux Mint** sur le bureau.

ÉTAPES DE L'ASSISTANT D'INSTALLATION

1. **Langue :** Français.
2. **Clavier :** French (Canada) ou French (Azerty) selon votre clavier physique. Testez dans la zone de texte !
3. **Codecs multimédia :** Cochez la case "Installer les codecs multimédia".
4. **Type d'installation :** Choisissez "*Effacer le disque et installer Linux Mint*".

PEUR D'EFFACER LE DISQUE ?

Quand l'installateur dit "Effacer le disque", il parle du **disque virtuel** de 25 Go que nous avons créé à l'étape 2.
Il ne peut **PAS** toucher à votre Windows ni à vos vrais fichiers. Vous êtes dans une bulle étanche. **Cliquez sur "Installer maintenant" sans crainte.**

5. **Fuseau horaire :** Cliquez sur votre ville (ex: Montréal/Paris).
6. **Qui êtes-vous ?**
   - Votre nom : `Étudiant`
   - Nom de l'ordinateur : `vm-linux` (Gardez-le court).
   - Nom d'utilisateur : `etudiant` (Minuscules, pas d'espaces).
   - Mot de passe : Choisissez un code simple pour le cours (ex: `linux123`).
   - **Important :** Cochez "Demander mon mot de passe pour ouvrir la session".
7. L'installation commence. Cela prendra 5 à 10 minutes.
8. À la fin, cliquez sur **Redémarrer maintenant**.
9. VirtualBox vous demandera de retirer le média d'installation : Appuyez simplement sur **Entrée**.

5. Post-Installation : Les Additions Invité

Une fois redémarré, vous remarquerez que l'écran est petit et ne s'adapte pas si vous agrandissez la fenêtre. Il faut installer les pilotes VirtualBox.

1. Dans la fenêtre de la VM (pas dans Linux, mais le cadre VirtualBox autour), allez dans le menu : **Périphériques** > **Insérer l'image CD des Additions Invité**.
2. Linux Mint va détecter le CD et demander "Voulez-vous lancer ce logiciel ?". Cliquez sur **Lancer**.
3. Entrez votre mot de passe (celui créé à l'installation).
4. Un terminal noir va s'ouvrir et installer des modules. Attendez de voir la ligne : "*Press Return to close this window*".
5. Appuyez sur Entrée.
6. **Redémarrez la VM** (Menu Mint > Éteindre > Redémarrer).

Félicitations ! Vous pouvez maintenant redimensionner la fenêtre, copier-coller entre Windows et Linux et profiter de votre laboratoire.

GUIDE TECHNIQUE : Création d'une clé USB d'installation

Ce guide explique comment transformer une clé USB standard en support d'installation (Live USB) pour **Linux Mint 22.2 (Cinnamon)**.

Objectif : Démarrer un ordinateur sur Linux pour l'installer ou le réparer.

1. Prérequis matériels et logiciels

- **Une clé USB :** Capacité minimale de 8 Go.
- **Le fichier ISO :** L'image disque `linuxmint-22.2-cinnamon-64bit.iso`.
- **Un logiciel de gravure :** *BalenaEtcher* (Windows/macOS) ou *Créateur de clé USB* (Linux).

ATTENTION : PERTE DE DONNÉES

L'opération va **effacer irréversiblement** tout le contenu de la clé USB. Assurez-vous d'avoir sauvegardé vos documents personnels présents sur la clé avant de commencer.

2. Méthode A : Depuis Windows (Recommandée)

Nous utilisons **BalenaEtcher** car il est simple et évite les erreurs de formatage fréquentes avec d'autres outils.

1. **Téléchargement :** Récupérez le logiciel sur balena.io/etcher.
2. **Préparation :** Insérez votre clé USB et lancez Etcher.
3. **Configuration :**
   - Cliquez sur **Flash from file** et choisissez l'ISO de Linux Mint.
   - Cliquez sur **Select target** et cochez votre clé USB.
4. **Gravure :** Cliquez sur **Flash!**. Cela prendra quelques minutes.

POURQUOI WINDOWS PANIQUE-T-IL ?

À la fin de la gravure, Windows va probablement ouvrir plusieurs fenêtres vous demandant de **"Formater le disque"**.

NE LE FAITES SURTOUT PAS ! Fermez simplement les fenêtres (Annuler).
Explication : Windows ne sait pas lire le système de fichiers Linux (ext4/iso9660) que nous venons de créer. Il croit que la clé est cassée, mais elle est parfaitement fonctionnelle pour Linux.

3. Méthode B : Depuis Linux Mint (Natif)

Si vous êtes déjà sur un poste au laboratoire, c'est la méthode la plus rapide.

1. Insérez la clé USB.
2. Ouvrez le menu principal (Touche Super/Windows) et tapez **Créateur de clé USB**.
3. **Écrire l'image :** Sélectionnez votre fichier `.iso`.
4. **Vers :** Sélectionnez votre clé USB dans la liste.
5. Cliquez sur **Écrire** et entrez votre mot de passe administrateur.

4. Démarrage sur la clé (Boot)

Une fois la clé prête, il faut dire à l'ordinateur de démarrer dessus au lieu de démarrer sur le disque dur interne (Windows).

1. Éteignez complètement l'ordinateur.
2. Insérez la clé USB.
3. Appuyez sur le bouton d'allumage.
4. **Immédiatement** (dès que l'écran s'allume), tapotez de façon répétée sur la touche du **Boot Menu** correspondant à la marque de votre ordinateur :

Marque	Touche de Boot (Tapoter au démarrage)
Acer / Dell / Lenovo	F12
HP	F9 ou Echap (Esc)
Asus	F8 ou Echap (Esc)
MSI	F11

L'ÉCRAN DE SÉLECTION (GRUB)

1. Dans le menu de démarrage, sélectionnez votre clé USB. Elle est souvent préfixée par **UEFI : [Nom de la clé]**.
2. Un écran noir avec du texte blanc va apparaître (c'est GRUB).
3. Validez la première option : **Start Linux Mint 22.2 Cinnamon 64-bit**.