2nde BAC Pro CIEL		
CIEL	**LINUX COMMANDE**	COURS Année 2024/2025

Table des matières

Attention ce document est principalement utilisé pour les activités de découverte de Linux dans le cadre du BAC PRO CIEL avec M. Béger.

Il ne fait pas office de référence officiel de commande Linux et n'est qu'une synthèse.

La seule référence officielle de l'utilisation des commandes Linux est l'outil manuel intégré via la commande `man` tel que :

```
$ man [commande]
```

avec [commande], le nom de la commande dont on souhaite obtenir le manuel d'utilisation.

A. Présentation de Linux

1) Qu'est-ce que Linux ?

Linux est un système d'exploitation libre et open source, créé en 1991 par Linus Torvalds. Il repose sur un noyau, appelé « kernel », qui assure la communication entre les programmes et le matériel de l'ordinateur. Contrairement à Windows ou macOS, Linux n'appartient à aucune entreprise : son code source est ouvert et peut être librement utilisé, étudié, modifié et redistribué par quiconque.

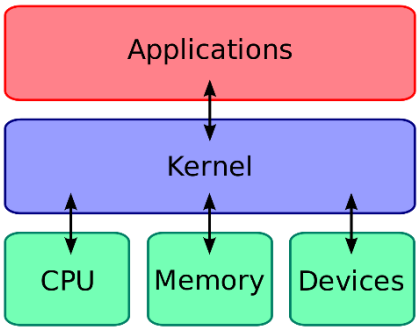
Cette philosophie du partage et de la transparence a permis à Linux de devenir l'un des systèmes les plus utilisés au monde. On le retrouve aujourd'hui dans les serveurs, les objets connectés, les routeurs, les téléphones Android, les supercalculateurs et même certaines voitures modernes.

2) La philosophie du logiciel libre

Le cœur de l'esprit Linux repose sur les principes du logiciel libre. Un logiciel libre n'est pas forcément gratuit : il est surtout libre dans le sens de la liberté. L'utilisateur a le droit de l'utiliser pour n'importe quel usage, d'en étudier le fonctionnement, de le modifier pour l'adapter à ses besoins et enfin de le redistribuer.

Cette approche a favorisé la création d'une immense communauté mondiale de développeurs et d'utilisateurs qui collaborent ensemble pour améliorer le système. C'est ce modèle communautaire qui explique la fiabilité et la rapidité d'évolution de Linux. Chacun peut contribuer, corriger des erreurs, proposer de nouvelles fonctionnalités ou simplement aider à la documentation.

3) La structure générale du système



Un système Linux est composé de plusieurs couches.

Au cœur se trouve le noyau (kernel), véritable chef d'orchestre du système. Il gère la mémoire, le processeur, les périphériques et la communication entre les différentes parties du système.

Autour du noyau, on trouve les utilitaires GNU, qui constituent l'ensemble des outils nécessaires à la gestion des fichiers, des processus, des utilisateurs ou du réseau. C'est l'association de ces deux éléments — le noyau Linux et les outils GNU — qui a donné naissance à l'univers GNU/Linux.

Enfin, l'utilisateur interagit avec le système par l'intermédiaire d'une interface. Il peut s'agir d'une interface graphique, semblable à celle de Windows, ou d'une interface en ligne de commande appelée « shell », où l'on saisit directement les instructions au clavier. Le shell le plus courant est Bash, mais il en existe d'autres comme Zsh ou Fish.

4) Les distributions Linux

Contrairement à d'autres systèmes d'exploitation, Linux n'existe pas sous une seule forme. Il en existe des centaines de variantes, appelées distributions. Chacune d'elles repose sur le même noyau, mais propose une sélection différente d'outils, de logiciels et de configurations.

Parmi les plus connues, on peut citer :

- **Debian**, réputée pour sa stabilité

- **Ubuntu**, dérivée de Debian, qui vise la simplicité d'utilisation

- **Fedora**

- **Red Hat Enterprise Linux**, utilisées dans le monde professionnel

- **Arch Linux**, destinée aux utilisateurs avancés

Certaines distributions sont spécialisées dans un domaine précis. Par exemple, Kali Linux est conçue pour la cybersécurité et le test d'intrusion, tandis que Raspberry Pi OS est adaptée aux petits ordinateurs embarqués. Dans le cadre du Bac Pro CIEL, on utilise souvent Debian ou Ubuntu Server, car elles offrent un environnement stable, complet et libre de droits.

5) Les principes fondamentaux de Linux

Linux repose sur quelques principes simples mais puissants. Le plus célèbre est : « Tout est fichier ». Dans ce système, tout — qu'il s'agisse d'un document, d'un périphérique, d'un processus ou d'un répertoire — est considéré comme un fichier. Cette logique uniforme simplifie la gestion et l'automatisation du système.

Un autre principe essentiel est la séparation des droits entre utilisateurs. Le système distingue le super-utilisateur, appelé « root », qui possède tous les privilèges, et les utilisateurs classiques, dont les permissions sont limitées pour éviter les erreurs ou les failles de sécurité. Chaque fichier ou dossier est associé à des droits d'accès : lecture, écriture et exécution. Ces permissions sont attribuées à trois catégories d'utilisateurs : le propriétaire, le groupe et les autres. Ce mécanisme contribue à la sécurité et à la stabilité du système.

6) Les avantages de Linux

L'un des atouts majeurs de Linux est sa stabilité. Il peut fonctionner sans redémarrage pendant des mois, voire des années, ce qui explique son succès dans les serveurs. Il est également sécurisé : la gestion fine des droits utilisateurs, l'ouverture du code source et la rapidité des mises à jour limitent les risques de virus et de failles.

Linux est aussi libre et gratuit, ce qui en fait un choix économique pour les entreprises et les établissements scolaires. Il est personnalisable à tous les niveaux : de l'apparence graphique à la configuration du noyau. Enfin, c'est un système réseau par nature. De nombreux services comme SSH, Apache, DNS, DHCP ou FTP ont été conçus pour fonctionner d'abord sous Linux.

7) Les modes d'utilisation

Linux peut être utilisé de plusieurs manières selon le contexte. Sur un poste de travail, il est souvent accompagné d'une interface graphique conviviale, proche de ce que l'on retrouve sur Windows ou macOS. En revanche, dans les environnements professionnels, on privilégie l'utilisation en ligne de commande, via un terminal. Ce mode peut sembler austère au début, mais il offre une rapidité et une précision incomparables pour l'administration et l'automatisation.

Dans le cas d'un serveur, l'environnement graphique est généralement absent : le système fonctionne en arrière-plan et se gère à distance, par exemple via le protocole SSH. C'est ce mode qui est le plus utilisé dans les réseaux d'entreprise et dans les datacenters.

8) Linux dans le monde professionnel

Aujourd'hui, Linux est omniprésent dans les infrastructures informatiques. La majorité des serveurs Internet tournent sous une distribution Linux, tout comme les routeurs, les pare-feux, les systèmes de virtualisation et les équipements réseaux. Dans le domaine de la cybersécurité, il est incontournable : de nombreux outils de test, d'analyse ou de surveillance sont conçus pour fonctionner sur ce système.

On le retrouve aussi dans les systèmes embarqués : téléphones Android, voitures connectées, drones, objets domotiques... Tous reposent sur une base Linux adaptée à leurs besoins.

B. L'arborescence et la structure du système de fichiers Linux

1) Un système organisé comme un arbre

Sous Linux, les fichiers sont organisés sous la forme d'un arbre hiérarchique unique qui prend racine dans un répertoire appelé «**/**» (racine).

Contrairement à Windows, où chaque disque ou partition possède sa propre lettre (C:\, D:\, E:\,...), Linux considère que tout fait partie d'un même système de fichiers, quel que soit le support de stockage.

Ainsi, un disque externe, une clé USB ou une partition réseau ne créent pas de nouvelle racine, mais sont simplement montés (rattachés) dans un dossier de cette arborescence.

On dit souvent que tout commence à la racine "**/**", puis se ramifie en répertoires spécialisés selon leur usage.

Chaque fichier, dossier ou périphérique possède donc un chemin absolu permettant de le localiser, par exemple :

```
/home/alain/Documents/notes.txt
```

2) Les répertoires principaux

L'arborescence Linux suit une structure normalisée appelée **FHS** (Filesystem Hierarchy Standard).

Chaque dossier à la racine a un rôle précis et contribue au bon fonctionnement du système.

Voici les plus importants :

/ : c'est la racine du système. Tous les autres dossiers y sont rattachés.

/bin : contient les commandes de base nécessaires au fonctionnement du système, comme ls, cp, mv, cat ou mkdir.

/sbin : similaire à /bin, mais réservé aux outils d'administration du système, comme shutdown, ifconfig ou fdisk.

/etc : regroupe les fichiers de configuration du système et des services. Chaque logiciel y stocke ses paramètres sous forme de fichiers texte.

/home : c'est le dossier personnel de chaque utilisateur.

Par exemple, l'utilisateur alain aura son répertoire `/home/alain`, où seront stockés ses documents, téléchargements, configurations personnelles, etc.

/root : dossier personnel du super-utilisateur root.

/lib : contient les bibliothèques partagées nécessaires à l'exécution des programmes, l'équivalent des DLL sous Windows.

/dev : regroupe les périphériques matériels (disques, clés USB, ports série, etc.), représentés sous forme de fichiers spéciaux.

/mnt et **/media** : servent de points de montage temporaires pour les supports externes.

/var : contient les données variables, c'est-à-dire celles qui évoluent souvent : journaux du système (/var/log), files d'attente, bases de données, etc.

/tmp : espace temporaire utilisé pour les fichiers créés de manière provisoire par les programmes.

/usr : abréviation de "user system resources", il contient la majorité des logiciels installés et leurs fichiers associés.

/boot : renferme les fichiers nécessaires au démarrage du système, dont le noyau Linux et le chargeur d'amorçage (GRUB).

/proc et **/sys** : répertoires virtuels qui donnent des informations en temps réel sur le système et les processus en cours. Ils ne contiennent pas de vrais fichiers, mais des représentations de l'état interne du système.

3) Les chemins absolus et relatifs

Pour accéder à un fichier ou un dossier, Linux utilise la notion de chemin.

Un chemin absolu part toujours de la racine **`/`** et décrit la position complète d'un élément. Par exemple :

`/home/alain/images/photo.png.`

C'est une adresse unique qui reste valable quel que soit l'endroit où l'on se trouve dans le système.

Un chemin relatif, au contraire, est exprimé à partir du dossier courant.

Si vous êtes déjà dans `/home/alain`, vous pouvez simplement écrire `images/photo.png` pour accéder au même fichier.

Deux symboles particuliers facilitent la navigation :

`.` désigne le répertoire courant.

`..` désigne le répertoire parent.

Ces deux notions sont essentielles pour comprendre le fonctionnement des commandes de navigation, que l'on verra dans la partie suivante.

4) Le rôle du dossier personnel

Chaque utilisateur dispose de son propre espace dans le répertoire `/home`.

Ce dossier contient ses documents, ses paramètres et parfois des dossiers cachés (dont le nom commence par un point) contenant les configurations des logiciels.

Par exemple, **`.bashrc`** est un fichier caché qui définit le comportement du shell Bash pour cet utilisateur.

Cette organisation garantit la séparation entre les comptes, ce qui évite qu'un utilisateur modifie les fichiers d'un autre ou du système.

Lorsque l'on se connecte à une session, le terminal s'ouvre automatiquement dans ce dossier personnel.

Ainsi, un simple **`cd`** sans argument permet de revenir directement à son répertoire `/home/utilisateur`.

5) Le système de fichier et le montage

Sous Linux, un système de fichiers (filesystem) correspond à la manière dont les données sont organisées sur un support de stockage.

Les formats les plus utilisés sont ext4, xfs, btrfs ou encore fat32 pour les périphériques amovibles.

Chaque partition ou disque est accessible uniquement lorsqu'il est monté, c'est-à-dire intégré dans l'arborescence.

Le montage se fait souvent automatiquement, mais il peut aussi être effectué manuellement avec la commande `mount`

Par exemple, si une clé USB est montée dans `/media/usb`, tout son contenu sera visible dans ce dossier.

Cette flexibilité permet à Linux de gérer un grand nombre de périphériques sans multiplier les lettres de lecteur, comme c'est le cas sous Windows.

6) Les fichiers spéciaux et virtuels

Une particularité de Linux est de représenter les périphériques et les informations système sous forme de fichiers.

Dans le dossier `/dev`, chaque périphérique matériel est vu comme un fichier : le disque principal peut s'appeler `/dev/sda`, une clé USB `/dev/sdb`, et ainsi de suite.

Cette représentation unifiée permet d'interagir avec le matériel à l'aide des commandes habituelles : on peut lire, écrire ou interroger un périphérique comme s'il s'agissait d'un simple fichier.

Les répertoires `/proc` et `/sys` fonctionnent de manière similaire, mais de façon virtuelle. Ils n'existent pas réellement sur le disque : le système y génère à la volée des fichiers contenant des informations sur les processus, la mémoire, les interfaces réseau, ou encore la configuration du noyau.

Par exemple, lire le fichier `/proc/cpuinfo` permet d'obtenir les caractéristiques du processeur en temps réel.

7) Importance de comprendre l'arborescence

Pour un utilisateur débutant, cette organisation peut sembler complexe, mais elle repose sur une logique très cohérente.

Chaque élément du système a une place bien définie, ce qui rend Linux prévisible et structuré.

Cette arborescence normalisée permet à un administrateur de s'y retrouver immédiatement, quel que soit le type de distribution.

Comprendre cette structure est indispensable avant d'apprendre les commandes de navigation, de manipulation de fichiers et de gestion du système.

C. Les commandes de navigation dans l'arborescence

1) Introduction

Dans un système Linux, la plupart des opérations d'administration s'effectuent dans un terminal, c'est-à-dire une interface en ligne de commande.

Pour manipuler des fichiers, il faut donc savoir se déplacer dans les répertoires, afficher leur contenu et identifier l'endroit où l'on se trouve.

Ces actions, très simples en apparence, constituent les fondations de toute activité sur le système.

2) La commande **pwd** : connaître sa position

Lorsqu'on ouvre un terminal, il est essentiel de savoir où l'on se trouve dans l'arborescence.

La commande **pwd** (Print Working Directory) affiche le chemin absolu du répertoire courant.

Autrement dit, elle indique la "branche" de l'arbre sur laquelle on se trouve.

Par exemple :

```
$ pwd
/home/alain/Documents
```

Cette commande permet de vérifier à tout moment que l'on travaille au bon endroit, avant de copier, supprimer ou exécuter des fichiers.

Par défaut, lors de la connexion, un utilisateur est placé dans son dossier personnel (`/home/nom_utilisateur`).

L'utilisateur root, quant à lui, se trouve dans `/root`.

3) La commande `ls` : Lister le contenu d'un dossier

Pour connaître le contenu d'un répertoire, on utilise la commande **`ls`** (LiSt).

Employée seule, elle affiche simplement les noms des fichiers et dossiers présents dans le répertoire courant :

```
$ ls

Bureau  Documents  Images  Téléchargements
```

`ls` dispose de nombreuses options qui modifient son affichage :

- ⇨ **`ls -l`** affiche le contenu sous forme de liste détaillée (permissions, propriétaire, taille, date, etc.)
- ⇨ **`ls -a`** montre également les fichiers cachés (ceux dont le nom commence par un point)
- ⇨ **`ls -lh`** combine les deux précédentes options et affiche les tailles en unités lisibles (Ko, Mo, Go).

Ainsi :

```
$ ls -lha
```

affiche tous les fichiers, y compris cachés, avec des détails complets et lisibles.

C'est la forme la plus couramment utilisée par les administrateurs.

4) La commande `cd` : se déplacer dans l'arborescence

La commande **`cd`** (Change Directory) permet de changer de dossier dans lequel on travaille.

Pour aller dans le répertoire `/etc`, il suffit d'écrire :

```
$ cd /etc
```

Si l'on veut se déplacer vers un dossier à l'intérieur du répertoire courant, on peut indiquer un chemin relatif :

```
$ cd Documents
```

ou encore revenir au dossier parent :

```
$ cd ..
```

Quelques raccourcis sont très utiles :

`cd` seul ramène toujours au dossier personnel de l'utilisateur

`cd -` permet de revenir au dossier précédent ;

`cd /` vous place directement à la racine du système.

La commande **`pwd`** permet ensuite de vérifier le nouveau répertoire dans lequel on se trouve.

5) La commande `tree` : visualiser la structure hiérarchique

La commande **`tree`** affiche le contenu d'un répertoire sous forme d'arborescence.

Elle n'est pas toujours installée par défaut, mais peut l'être facilement via le gestionnaire de paquets.

Son affichage est très parlant pour comprendre la structure d'un dossier :

```
$ tree
.
├── Documents
│   ├── rapport.txt
│   └── images
│       └── photo.jpg
└── Téléchargements
```

Cette commande est particulièrement utile à des fins pédagogiques, car elle permet de visualiser l'organisation logique du système de fichiers.

6) La commande `lsblk` : explorer les supports de stockage

Pour aller un peu plus loin, la commande **`lsblk`** (LiSt BLocK devices) affiche les disques et partitions montées sur le système :

```
$ lsblk
```

Elle indique la hiérarchie des disques (sda, sdb), les partitions associées et les points de montage (`/`, `/home`, `/media/usb`, etc.).

C'est une commande précieuse pour comprendre où se trouve physiquement chaque partie de l'arborescence.

D. Manipulation de fichiers dans l'arborescence

1) Introduction

Après avoir appris à se déplacer dans l'arborescence et à manipuler les dossiers, il est temps d'apprendre à créer, consulter et modifier des fichiers.

Sous Linux, ces opérations se font principalement en ligne de commande, à l'aide de quelques outils simples mais puissants.

2) La commande **touch** : créer un fichier vide

La commande **touch** permet de créer un nouveau fichier sans contenu.

C'est souvent la première étape avant de l'éditer.

Par exemple :

```
$ touch notes.txt
```

créera un fichier nommé « notes.txt » dans le répertoire courant.

Si le fichier existe déjà, **touch** met simplement à jour sa date de dernière modification, sans effacer son contenu.

Cette commande est très pratique pour préparer une structure de dossiers ou pour créer rapidement plusieurs fichiers à la fois :

```
$ touch lundi.txt mardi.txt mercredi.txt
```

créera 3 fichiers nommés "lundi.txt", "mardi.txt" et "mercredi.txt".

3) La commande **echo** : écrire du texte rapidement

echo permet d'afficher un texte dans le terminal, ou de l'écrire directement dans un fichier.

Sans redirection, elle affiche simplement le message à l'écran :

```
$ echo "Bonjour Linux !"

Bonjour Linux !
```

Pour enregistrer ce message dans un fichier, on utilise le symbole **>** :

```
$ echo "Bonjour Linux !" > message.txt
```

Le fichier message.txt sera créé (ou remplacé s'il existait déjà).

Si l'on souhaite ajouter du texte sans effacer le contenu existant, on utilise **>>** :

```
$ echo "Deuxième ligne" >> message.txt
```

Cette commande simple permet donc de créer et compléter des fichiers texte depuis le terminal sans passer par un éditeur.

4) La commande `cat` : afficher le contenu d'un fichier

cat (pour concatenate) sert à afficher le contenu d'un fichier texte directement dans le terminal :

```
$ cat message.txt
Bonjour Linux !
Deuxième ligne
```

Elle peut également être utilisée pour fusionner plusieurs fichiers :

```
$ cat fichier1.txt fichier2.txt > total.txt
```

Le fichier total.txt contiendra alors la suite des deux premiers.

Enfin, utilisée sans argument, **cat** lit le texte que l'on tape au clavier et l'affiche immédiatement. C'est un bon moyen d'observer le comportement du terminal.

5) La commande `tail` : lire les dernières lignes d'un fichier

tail est très utile pour consulter la fin d'un fichier, notamment dans le cas de journaux système ou de fichiers qui se mettent à jour en continu.

Par défaut, elle affiche les 10 dernières lignes :

```
$ tail /var/log/syslog
```

On peut préciser un autre nombre avec l'option **-n** :

```
$ tail -n 20 fichier.txt
```

Et surtout, l'option **-f** permet de suivre un fichier en temps réel :

```
$ tail -f /var/log/auth.log
```

Dans ce mode, le terminal affiche les nouvelles lignes dès qu'elles sont écrites dans le fichier — une fonction très utilisée par les administrateurs pour observer en direct l'activité du système.

6) La commande `head` : afficher le début d'un fichier

Complémentaire à **tail**, la commande **head** affiche les premières lignes d'un fichier.

Par défaut, elle en montre 10, mais on peut préciser un nombre :

```
$ head -n 5 /etc/passwd
```

permet de lire les 5 premières lignes d'un fichier de configuration ou d'un journal.

7) L'éditeur de texte nano

Pour modifier réellement le contenu d'un fichier, il faut un éditeur de texte.

Sous Linux, plusieurs existent, mais `nano` est le plus simple et le plus adapté pour débuter.

Il s'utilise directement dans le terminal :

```
$ nano notes.txt
```

Si le fichier notes.txt n'existe pas, il sera automatiquement créé.

L'écran de nano affiche le texte du fichier et une série de raccourcis en bas.

Les commandes les plus importantes sont :

- ⇨ **Ctrl + O** pour enregistrer le fichier
- ⇨ **Ctrl + X** pour quitter l'éditeur
- ⇨ **Ctrl + G** pour afficher l'aide
- ⇨ **Ctrl + K** pour couper une ligne et **Ctrl + U** pour la coller.

nano ne nécessite pas de connaissances particulières et permet d'éditer rapidement des fichiers de configuration, des scripts ou des notes.

Il est particulièrement utile lorsqu'on administre un système sans interface graphique.

8) Les commandes mkdir et rmdir : créer et supprimer un répertoire

Pour créer un répertoire, on utilise `mkdir` (MaKe DIRectory).

Par exemple :

```
$ mkdir sauvegardes
```

crée un répertoire nommé « sauvegardes » dans le répertoire courant.

On peut également créer un répertoire à un emplacement précis en indiquant un chemin absolu :

```
$ mkdir /home/alain/Documents/projets
```

La commande inverse, `rmdir`, permet de supprimer un répertoire vide :

```
$ rmdir sauvegardes
```

Si le répertoire contient des fichiers, il faudra utiliser `rm -r`.

9) La commande `cp` : copier un fichier ou un dossier

La commande `cp` (CoPy) permet de dupliquer un fichier d'un endroit à un autre.

```
$ cp fichier.txt /home/alain/Documents/
```

L'option `-r` permet de copier un dossier entier (avec son contenu) :

```
$ cp -r /etc /home/alain/sauvegarde/
```

10) La commande `mv` : Déplacer ou renommer un fichier ou un dossier

La commande `mv` (MoVe) permet de déplacer un fichier ou un répertoire vers un autre emplacement.

```
$ mv fichier.txt /tmp/
```

Si la destination est le même répertoire, `mv` agit comme un renommage :

```
$ mv fichier.txt ancien_fichier.txt
```

11) La commande `rm` : Supprimer un fichier

La commande `rm` (ReMove) supprime un fichier de manière définitive (pas de corbeille sous Linux) :

```
$ rm notes.txt
```

L'option `-r` permet de supprimer un répertoire et tout son contenu :

```
$ rm -r dossier/
```

⚠ Commande à manier avec prudence, surtout en tant que super-utilisateur.

Pour plus de sécurité, tu peux présenter l'option `-i` (interactive) qui demande une confirmation :

```
$ rm -ri dossier/
```

12) La commande `file` : Information sur les fichiers

La commande `file` permet de savoir si un fichier est un texte, une image, un exécutable, etc... :

```
$ file /bin/bash
```

13) La commande `du` : Estimer l'espace utilisé

La commande `du` affiche la taille d'un répertoire ou d'un fichier.

```
$ du -h /home/alain/Documents
```

L'option `-h` rend les tailles lisibles (en ko, Mo, Go).

14) La commande `find` : Rechercher des fichiers selon des critères

La commande **`find`** permet de rechercher des fichiers ou des dossiers dans l'arborescence selon une grande variété de critères : leur nom, leur type, leur taille, leur date de modification ou encore leurs permissions.

```
$ find /home/alain -name note.txt
```

permet de parcourir tout le répertoire /home/alain et afficher le chemin complet de tous les fichiers nommés « note.txt »

```
$ find /home -name "*.txt"
```

permet de parcourir tout le répertoire /home et afficher le chemin complet de tous les fichier dont le nom se termine par « .txt ».

```
$ find /home -iname "rapport.doc"
```

permet de parcourir tout le repertoire /home et afficher le chemin complet de tous les fichiers nommés « rapport.doc » sans tenir compte de la casse (rapport.doc, RaPpOrT.doc, rapport.DOC, RAPPORT.doc, ...).

```
$ find /etc -type d
```

permet de parcourir tout le répertoire /etc et afficher tous les répertoires présents à l'intérieur.

```
$ find /etc -type f
```

permet de parcourir tout le répertoire /etc et afficher tous les fichiers présents à l'intérieur.

```
$ find /var/log -type f -name "*.log"
```

permet de parcourir tout le répertoire /var/log et afficher tous les fichiers dont le nom se termine par .log.

```
$ find /home -size +10M
```

permet de parcourir tout le répertoire /home et afficher tous les fichiers supérieur à 10 Mo.

```
$ find . -size -100k
```

permet de parcourir tout le répertoire courant et afficher tous les fichiers inférieur à 100 ko.

```
$ find /home/alain/Documents -mtime -2
```

permet de parcourir tout le répertoire /home/alain/Documents et afficher les fichiers modifiés dans les deux derniers jours.

```
$ find /var/log -mtime +7 -name "*.log"
```

permet de parcourir tout le répertoire /var/log et afficher tous les fichiers dont le nom se termine par « .log » et qui a été modifié il y a plus de 7 jours.

```
$ find /home -user alain
```

permet de parcourir tout le répertoire /home et afficher tous les fichiers appartenant à l'utilisateur alain.

15) La commande `clear` : Nettoyer l'écran du terminal

La commande `clear` permet d'effacer l'affichage actuel pour repartir sur un écran vide :

```
$ clear
```

E. Gestion des utilisateurs et des droits

1) Introduction

Sous Linux, chaque action est réalisée par un utilisateur.

Qu'il s'agisse d'un administrateur, d'un élève, d'un programme ou même d'un service système, tout ce qui s'exécute sur la machine possède une identité.

Cette logique garantit la sécurité : les droits d'accès sont limités à chaque utilisateur, empêchant ainsi un programme ou une personne non autorisée de modifier des fichiers critiques du système.

Linux repose sur un modèle simple et efficace :

- ⇨ un super-utilisateur, appelé root, qui a tous les droits
- ⇨ des utilisateurs standards, disposant de permissions limitées
- ⇨ des groupes, qui rassemblent plusieurs utilisateurs partageant les mêmes privilèges.

2) Le super-utilisateur : root

L'utilisateur root est l'administrateur du système.

Il peut tout faire : installer ou supprimer des logiciels, créer ou effacer des comptes, modifier les fichiers système, changer les droits d'accès, etc.

Son pouvoir est total, ce qui signifie aussi qu'une erreur de manipulation peut avoir des conséquences graves.

C'est pourquoi, dans la pratique, on évite de se connecter directement en tant que root.

À la place, on utilise un compte standard et on exécute temporairement une commande en mode administrateur à l'aide de **sudo**.

Par défaut, sudo n'est pas forcément installé, on va donc temporairement se connecter en root grâce à la commande :

```
$ su -
```

Après l'exécution de cette commande, Linux demandera le mot de passe du compte root et vous passerez en mode super-utilisateur root. Le '$' est remplacé par '#'.

Lorsque sudo sera installé, un utilisateur pourra utiliser les commandes root grâce à sudo :

```
$ sudo [cmd]
```

permet d'exécuter la commande cmd en super-utilisateur root.

3) Les comptes utilisateurs

Chaque personne ou service qui utilise la machine dispose d'un compte utilisateur.

Lorsqu'un utilisateur est créé, le système lui attribue automatiquement :

- ⇨ un identifiant (login) et un mot de passe
- ⇨ un UID (User ID), numéro unique dans le système
- ⇨ un répertoire personnel, situé dans `/home` (ex. `/home/alain`)
- ⇨ un groupe principal, souvent du même nom que l'utilisateur

L'ensemble des comptes est répertorié dans le fichier `/etc/passwd`, qui contient pour chaque utilisateur des informations comme son nom, son UID, son répertoire personnel et son shell (souvent `/bin/bash`).

Le fichier `/etc/passwd` est organisé afin que chaque ligne correspond à un utilisateur.

Chaque ligne est organisé en champ séparé par ':' tel que :

- ⇨ Champ 1 : Nom de l'utilisateur utilisé pour la connexion
- ⇨ Champ 2 : Ancien emplacement du mot de passe aujourd'hui remplacé par « X »
- ⇨ Champ 3 : UID (User ID), le numéro d'identification unique de l'utilisateur
- ⇨ Champ 4 : GID (Group ID), le numéro d'identification du groupe principal de l'utilisateur
- ⇨ Champ 5 : Commentaire : Description ou nom complet de l'utilisateur
- ⇨ Champ 6 : Répertoire personnel (HOME), emplacement du dossier personnel de l'utilisateur dans l'arborescence
- ⇨ Champ 7 : Shell, programme lancé à la connexion de l'utilisateur

Les mots de passe, eux, sont stockés chiffrés dans `/etc/shadow`, fichier accessible uniquement par le super-utilisateur.

4) Se connecter avec un utilisateur

Pour se connecteur avec un utilisateur, la commande à utiliser est **`su -`** tel que :

```
$ su - etienne
```

Linux va demander le mot de passe et vous serez connecter avec l'utilisateur « etienne » dans notre exemple.

Pour se déconnecter de l'utilisateur, la commande à utiliser est **`exit`**.

5) Créer, modifier et supprimer un utilisateur

Pour créer, modifier et supprimer des utilisateurs sous Linux, il faut réaliser ces actions en tant que super-utilisateur.

Pour créer un utilisateur, la commande est **`adduser`** (ADD USER) :

```
# adduser paul
```

Cette commande vient de créer l'utilisateur « paul », son dossier personnel `/home/paul` et configure automatiquement les permissions de base. Cette commande va également déclencher plusieurs questions pour compléter les informations de l'utilisateur (mot de passe, numéro téléphone, prénom et nom, adresse, etc...)

Pour supprimer un utilisateur, la commande est **`deluser`** (DELete USER)

```
# deluser paul
```

Cette commande vient de supprimer l'utilisateur « paul ». Attention, elle ne supprime que l'utilisateur, son environnement de travail reste présent.

Pour supprimer l'utilisateur ET son environnement de travail personnel, il faut ajouter l'option `--remove-home` tel que :

```
# deluser --remove-home paul
```

6) Les groupes d'utilisateur

Un groupe permet de rassembler plusieurs utilisateurs partageant les mêmes besoins d'accès à certains fichiers ou dossiers.

Chaque utilisateur appartient au minimum à un groupe principal, mais peut aussi être membre de groupes secondaires.

L'ensemble des groupes du système est répertorié dans le fichier `/etc/group`.

Chaque ligne de ce fichier correspond à un groupe.

Chaque ligne est organisé en champ séparé par ':' tel que :

- ⇨ Champ 1 : Nom du groupe
- ⇨ Champ 2 : Mot de passe du groupe vide ou X
- ⇨ Champ 3 : GID (Group ID) Numéro unique d'identification du groupe
- ⇨ Champ 4 : Membres du groupe, liste des utilisateurs appartenant à ce groupe séparé par des virgules

7) Créer, modifier et supprimer un groupe

Pour créer, modifier et supprimer des groupes sous Linux, il faut réaliser ces actions en tant que super-utilisateur.

Pour créer un groupe, la commande est **addgroup** (ADD GROUP) :

```
# addgroup ingenieur
```

Cette commande vient de créer le groupe « ingenieur ».

Pour ajouter un utilisateur dans un groupe, la commande est **usermod** avec l'option **-aG** tel que :

```
# usermod -aG ingenieur paul
```

Cette commande vient d'ajouter l'utilisateur « paul » dans le groupe « ingenieur ».

Pour connaître les groupes secondaires d'un utilisateur, la commande est **groups** :

```
$ groups paul
```

Cette commande va afficher les groupes auxquels appartient l'utilisateur « paul »

Pour supprimer un groupe, la commande est **delgroup** (DELete GROUP) :

```
# delgroup ingenieur
```

Cette commande va supprimer le groupe « ingenieur »

8) Les droits d'accès aux fichiers

Chaque fichier et répertoire sous Linux possède trois catégories de permissions :

- ⇨ Le propriétaire (souvent l'utilisateur qui a créé le fichier)
- ⇨ Le groupe (ensemble d'utilisateurs)
- ⇨ Les autres (tous les autres utilisateurs du système)

Et pour chacune de ces catégories, trois types de droits sont possibles :

- ⇨ Lecture (r) : autorise la lecture du fichier ou le listage d'un dossier
- ⇨ Écriture (w) : autorise la modification du contenu ou la création/suppression de fichiers dans un dossier
- ⇨ Exécution (x) : autorise l'exécution d'un fichier ou l'accès à un dossier

La commande `ls -l` permet d'afficher ces droits avec :

Première colonne : Les permissions et types de fichiers

Ce sont 10 caractères les uns derrière les autres tels que :

Le premier caractère :

- - : C'est un fichier régulier (standard)
- d : C'est un répertoire (dossier – Directory)
- l : C'est un lien symbolique
- b : C'est un périphérique de bloc
- c : C'est un périphérique de caractère
- p : C'est un pipe (tunnel) nommé
- s : C'est un socket

Les 9 caractères suivants sont les permissions organisées en 3 groupes de 3. Le premier groupe de 3 caractères concerne le propriétaire, le second groupe de 3 caractères concerne le groupe d'utilisateur et le dernier groupe de 3 caractères concerne les autres utilisateurs.

- - : permission refusée
- r : permission de lecture
- w : permission d'écriture
- x : permission d'exécuter

Deuxième colonne : Le nombre de lien

Pour un fichier, cela indique le nombre de lien physiques vers ce fichier.

Pour un répertoire, cela indique le nombre de sous-répertoire et de fichiers qu'il contient.

Troisième colonne : le propriétaire (user)

L'identifiant du propriétaire du fichier ou de répertoire

Quatrième colonne : Le groupe (group)

L'identifiant du groupe propriétaire du fichier

Cinquième colonne : La taille

La taille du fichier ou du répertoire en octets.

Sixième colonne : L'horodatage

La date et l'heure de la dernière modification du fichier ou du répertoire

Dernière colonne : le nom

Nom du fichier ou du répertoire

Par exemple :

```
drwxr-xr--  2  alain  travail  4096  1 janv. 2025  manuel
```

d	indique que manuel est un répertoire
rwx	indique que le propriétaire du répertoire (alain) peut tout faire (lire, écrire et exécuter)
r-x	indique que les membres du groupe travail peuvent lire et exécuter le répertoire manuel
r--	indique que tous les autres utilisateurs peuvent seulement lire
alain	le nom d'utilisateur propriétaire de manuel
travail	le groupe d'utilisateur propriétaire de manuel
manuel	le nom du répertoire

La commande **chmod** (CHange MODe) permet de modifier les droits d'accès d'un répertoire ou d'un fichier.

Il existe deux méthodes pour changer les droits d'accès :

- ⇨ **Mode symbolique**

Permission	
r	Lire
w	Écrire
x	Éxécuter

Catégorie d'utilisateur	
u	user - propriétaire
g	group – groupe
o	others – autres
a	all – tous

L'opérateur '+' permet d'ajouter des droits.

L'opérateur '-' permet de retirer des droits.

```
$ chmod u+x script.sh
```

Cette commande ajouter le droit d'exécution au propriétaire du fichier « script.sh »

```
$ chmod go-w note.txt
```

Cette commande retire le droit d'écriture au groupe et aux autres utilisateurs du fichier « note.txt »

```
$ chmod ug+rw readme.txt
```

Cette commande va ajouter les droits d'écritures et de lecture au propriétaire et au groupe du fichier « readme.txt ».

- ⇨ **Mode numérique**

Propriétaire Owner (user – U)			Groupe Group (G)			Autres Others (o)		
r	w	x	r	w	x	r	w	x
4	2	1	4	2	1	4	2	1

Il suffit d'additionner les règles rwx par groupe de 3.

Par exemple :

7 : (4+2+1) = lecture, écriture et exécution

6 : (4+2) = lecture et écriture

4 : (4) = lecture seule

```
$ chmod 750 game.py
```

Cette commande donne tous les droits au propriétaire (7), la lecture et l'exécution au groupe (5) et rien aux autres (0).

La commande **chown** (CHange OWNer) permet de modifier le propriétaire d'un fichier :

```
# chown pierre notes.txt
```

Cette commande vient de changer le propriétaire du fichier « notes.txt » afin que le propriétaire soit « pierre ».

La commande **chgrp** (CHanGe GRouP) permet de modifier le groupe propriétaire d'un fichier :

```
# chgrp ingenieur system.log
```

Cette commande vient de changer le groupe propriétaire du fichier « system.log » afin que le groupe soit « ingenieur ».

Il est également possible de changer le propriétaire et le groupe simultanément :

```
# chown pierre:ingenieur system.log
```

Cette commande vient de changer le propriétaire du fichier « system.log » par « pierre » et le groupe par « ingenieur »

F. Gestion des paquets et mises à jour

1) Introduction

Sous Linux, tous les logiciels — du plus petit outil jusqu'à l'environnement graphique complet — sont organisés sous forme de paquets.

Un paquet (ou package) est une archive contenant un programme, sa description, ses dépendances, et parfois des scripts d'installation.

Ce système permet d'installer, de mettre à jour ou de supprimer des logiciels facilement, tout en assurant la cohérence du système.

Chaque distribution (Debian, Ubuntu, Fedora, etc.) dispose de son propre gestionnaire de paquets :

- ⇨ Debian et Ubuntu utilisent APT (Advanced Package Tool) avec des fichiers .deb
- ⇨ Fedora, Red Hat et CentOS utilisent DNF/YUM avec des fichiers .rpm

Dans le cadre du Bac Pro CIEL, on travaille le plus souvent sur Debian ou Ubuntu, donc sur la base du gestionnaire APT.

2) Les dépôts de logiciels

Les paquets ne sont pas téléchargés directement depuis un site Web, mais depuis des dépôts officiels (repositories).

Un dépôt est un serveur contenant des milliers de paquets classés et maintenus par la communauté ou par l'éditeur de la distribution.

Les dépôts utilisés par le système sont listés dans le fichier `/etc/apt/sources.list`

Structure typique :

```
deb [options] <URL> <suite> <composants>

deb-src [options] <URL> <suite> <composants>
```

On va lire la ligne telle que :

« Je veux télécharger les paquets binaires Debian, depuis le miroir <URL>, pour la version <suite>, en incluant les logiciels <composants>, <composants>, ... »

- ⇨ **Type :**
  - deb : binaires
  - deb-src : sources
- ⇨ **URL (mirroir) :**
  - L'adresse du miroir Debian (où se trouvent les paquets)
- ⇨ **Suite :**
  - stable : version actuelle stable recommandée pour les serveurs et la production. Ce sont des paquets bien testés très stable
  - testing : version de test (future stable), des paquets après un premier tri, moins stable mais plus à jour
  - unstable : version de développement (sid), intègre des nouveautés en premier potentiellement instable
  - oldstable : ancienne version stable toujours maintenue un certain temps pour transition
  - bookworm : nom de code de la version Debian, permet de cibler une version précise plutôt qu'une branche
- ⇨ **Composants :**
  - main : tous les paquets 100% libres selon la charte Debian
  - contrib : Logiciels libres nécessitant un paquet « non-free » pour fonctionner. Le paquet est libre mais utilise des dépendances non libres
  - non-free : Logiciels distribués sans code source ou sous licence restrictive (propriétaire)
  - non-free-firmware : Micro-logiciel pour périphériques (Wifi, GPU, Bluetooth,...) non libre. (Depuis Debian 12 ce contenu est séparé du reste pour plus de clarté)

3) Mettre à jour la liste des paquets

Avant toute installation ou mise à jour, il faut rafraîchir la base locale des paquets.

Cette base contient la liste de tous les logiciels disponibles et leurs versions.

La commande est :

```
# apt update
```

Elle interroge tous les dépôts configurés et télécharge la liste la plus récente.

C'est une étape indispensable avant d'installer un nouveau paquet ou de mettre le système à jour.

4) Mettre à jour le système

Une fois la liste actualisée, on peut procéder à la mise à jour des logiciels installés :

```
# apt upgrade
```

Cette commande télécharge et installe les nouvelles versions disponibles pour les paquets déjà présents sur le système.

Elle ne supprime rien et ne modifie pas les dépendances.

Pour effectuer une mise à jour plus complète (ajout/suppression de dépendances selon les besoins), on peut utiliser :

```
# apt full-upgrade
```

Ces mises à jour permettent de corriger des failles de sécurité, d'améliorer les performances et de maintenir un système stable et à jour.

Sur un serveur ou un poste sensible, elles sont indispensables à la sécurité.

5) Rechercher un paquet

Pour savoir si un logiciel est disponible dans les dépôts, on utilise :

```
$ apt search [nom_du_logiciel]
```

Exemple :

```
$ apt search nmap
```

Cette commande va afficher les paquets correspondant à "nmap", leur description et leur version.

On peut aussi obtenir des informations plus précises sur un paquet particulier :

```
$ apt show nmap
```

6) Installer un paquet

L'installation d'un logiciel se fait en une seule commande :

```
# apt install [nom_du_paquet]
```

Exemple :

```
# apt install net-tools
```

APT se charge de télécharger le paquet et toutes les dépendances nécessaires à son fonctionnement.

C'est un des grands avantages du système Linux : les installations sont automatisées et cohérentes.

7) Supprimer un paquet

Pour désinstaller un logiciel tout en conservant ses fichiers de configuration :

```
# apt remove [nom_du_paquet]
```

Pour tout supprimer, y compris les fichiers de configuration :

```
# apt purge [nom_du_paquet]
```

Enfin, pour supprimer les dépendances inutiles (anciens paquets devenus orphelins) :

```
# apt autoremove
```

8) Nettoyer et vérifier le système

Avec le temps, des fichiers temporaires et des caches peuvent occuper de la place.

On peut les supprimer pour libérer de l'espace :

```
# apt clean
```

Cette commande vide le cache local de téléchargement des paquets.

Pour vérifier l'intégrité d'un paquet déjà installé :

```
# debsums nom_du_paquet
```

(s'il n'est pas installé, on peut le récupérer via `apt install debsums`)

G. Gestion des services et processus

1) Introduction

Sous Linux, tout ce qui s'exécute est un processus.

Un processus est une instance d'un programme en cours d'exécution : qu'il s'agisse d'un navigateur Web, d'un script, d'un serveur ou d'un démon système, chacun fonctionne dans son propre espace mémoire et possède son propre identifiant.

Savoir gérer ces processus, les observer, les suspendre ou les arrêter, est une compétence essentielle pour maintenir un système stable et performant.

De la même façon, certains processus sont dits services (services système ou daemons) : ce sont des programmes qui fonctionnent en arrière-plan dès le démarrage du système.

Exemples : le serveur SSH, le service réseau, ou encore le serveur Web Apache.

2) Identifier les processus en cours

Pour visualiser la liste des processus actifs, on peut utiliser plusieurs commandes.

La commande **ps** :

```
$ ps
```

qui affiche les processus liés à la session courante.

Pour obtenir une vue plus complète :

```
$ ps aux
```

a : affiche les processus de tous les utilisateurs

u : montre le nom de l'utilisateur associé

x : inclut les processus sans terminal associé

Exemple :

```
$ ps aux | head
USER   PID %CPU %MEM COMMAND
root     1  0.0  0.2 /sbin/init
alain  132  0.3  1.1 /usr/lib/firefox/firefox
```

Chaque processus possède un identifiant unique appelé PID (Process ID), utilisé pour le manipuler.

3) Observer les processus en temps réel

Pour suivre en direct l'activité du système, on utilise la commande :

```
$ top
```

Elle affiche en temps réel la liste des processus classés par utilisation CPU ou mémoire.

L'interface se met à jour automatiquement toutes les quelques secondes.

La commande **htop**, plus moderne et interactive, peut être installée avec :

```
# apt install htop
```

Elle permet d'utiliser les flèches pour naviguer, trier les processus, et même en terminer un en appuyant sur F9.

C'est un outil idéal pour une visualisation de l'activité du système.

4) Terminer un processus

Lorsqu'un programme se bloque ou consomme trop de ressources, il peut être nécessaire de le forcer à se fermer.

On utilise pour cela la commande kill suivie du PID :

```
$ kill 132
```

Si le processus ne répond toujours pas, on peut utiliser le signal -9 (SIGKILL) :

```
$ kill -9 132
```

⚠ Ce signal interrompt le processus sans lui laisser le temps de se fermer proprement — à n'utiliser qu'en dernier recours.

On peut aussi rechercher un processus par son nom et le tuer directement avec :

```
$ pkill firefox
```

5) Lancer un processus en arrière-plan

Par défaut, lorsqu'on exécute un programme depuis le terminal, celui-ci bloque la session tant qu'il n'est pas terminé.

Pour exécuter un processus en arrière-plan, il suffit d'ajouter le symbole **&** à la fin de la commande :

```
$ firefox &
```

La commande s'exécute alors sans bloquer le terminal.

Si un processus est déjà lancé, on peut le suspendre avec Ctrl + Z puis le reprendre en arrière-plan avec :

```
$ bg
```

ou au premier plan avec :

```
$ fg
```

Ces manipulations sont utiles pour gérer plusieurs tâches simultanément.

6) Les services systèmes et systemd

Depuis plusieurs années, la plupart des distributions modernes (comme Debian ou Ubuntu) utilisent **systemd** pour gérer les services.

Un service est un processus particulier qui démarre automatiquement avec le système et reste actif en arrière-plan.

Pour gérer les services, la commande de base est :

```
$ systemctl
```

Quelques exemples essentiels :

Action	Commande	Description
Démarrer un service	`systemctl start ssh`	Lance le service ssh
Arrêter un service	`systemctl stop ssh`	Interrompt le service ssh
Redémarrer un service	`systemctl restart ssh`	Relance un service après modification
Vérifier l'état	`systemctl status ssh`	Indique si le service est actif
Activer au démarrage	`systemctl enable ssh`	Active le démarrage automatique
Désactiver au démarrage	`systemctl disable ssh`	Supprime l'auto-démarrage

Exemple :

```
# systemctl status ssh
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; enabled)
   Active: active (running) since Fri 2025-11-16 10:12:44 CET
```

7) Démarrage et arrêt du système

Les commandes suivantes permettent de contrôler le système lui-même :

Action	Commande
Redémarrer le système	`reboot`
Éteindre la machine	`poweroff`
Passer en veille	`systemctl suspend`

Lors de l'arrêt, systemd s'assure que les services sont correctement arrêtés dans le bon ordre, ce qui évite les corruptions de données.

8) Surveillance de journaux de service

Lorsqu'un service ne démarre pas ou rencontre une erreur, il est essentiel de consulter les journaux système.

systemd stocke ces informations dans le journal accessible avec :

```
$ journalctl -u ssh
```

Cette commande affiche les logs du service SSH.

Pour voir les dernières lignes :

```
$ journalctl -u apache2 -n 20
```

Et pour suivre les journaux en direct :

```
$ journalctl -f
```

Ces commandes sont très utilisées en cybersécurité et en maintenance pour diagnostiquer un dysfonctionnement ou surveiller une intrusion.

H. Gestion réseau

1) Introduction

Linux est très largement utilisé dans les infrastructures réseau : serveurs, routeurs, équipements IoT, systèmes de supervision...

Pour comprendre, configurer et diagnostiquer un réseau, il existe sous Linux une série de commandes essentielles.

Elles permettent de visualiser la configuration IP, les interfaces, les routes, les ports ouverts, ou encore les connexions en cours.

2) Identifier le nom de la machine

La première information réseau de base est le nom de la machine (hostname).

Il permet d'identifier un poste dans un réseau local ou un domaine.

Pour l'afficher, la commande est :

```
$ hostname
```

Pour afficher les informations détaillées (dont le nom de domaine si configuré) :

```
$ hostnamectl
```

3) Visualiser les interfaces réseau avec **ip**

La commande **ip** est la référence moderne pour afficher et configurer les interfaces réseau.

Cette commande liste les interfaces (ex : eth0, wlan0, lo), leurs adresses IP, leur statut (UP/DOWN), et leurs adresses MAC.

Par exemple :

```
$ ip a
2: enp0s3: <UP,BROADCAST,RUNNING>
    link/ether 08:00:27:ae:57:18 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.42/24 brd 192.168.1.255 scope global dynamic
```

4) Afficher la table de routage avec ip route

La table de routage indique comment la machine décide d'atteindre les autres réseaux, notamment la passerelle par défaut :

```
$ ip route

default via 192.168.1.1 dev enp0s3
```

5) Tester la connectivité avec ping

La commande **ping** envoie des paquets ICMP pour tester la communication avec une machine distante.

```
$ ping 192.168.1.1

PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data
64 bytes from 192.168.1.42: icmp_seq1 ttl=64 time=3.92 ms
64 bytes from 192.168.1.42: icmp_seq2 ttl=64 time=3.40 ms
```

Ctrl + X permet d'arrêter le test de ping.

```
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
```

On peut également limiter le test à un certain nombre paquets (4 dans l'exemple suivant) :

```
$ ping -c 4 www.google.fr
```

6) Test de chemin avec traceroute

Pour voir le chemin que prennent les paquets sur Internet :

```
$ traceroute www.google.fr
```

Cette commande indique chaque routeur traversé, utile pour diagnostiquer les coupures ou les lenteurs réseau.

7) Analyse des ports et connexions avec ss

La commande **ss** (Socket Statistics) permet d'afficher toutes les connexions réseau et tous les ports qui écoutent sur un système Linux.

C'est l'un des outils essentiels pour vérifier si un service réseau fonctionne réellement.

```
$ ss -ltn
```

Avec les options :

- l : pour listening afin d'afficher uniquement les ports qui attendent une connexion

- t : pour TCP afin de n'afficher que les ports TCP

- n : pour numeric afin d'afficher le numéro des ports et non leur nom

Par exemple, la réponse suivante :

```
LISTEN     0     128   0.0.0.0:32542          0.0.0.0:*
```

Un service est en écoute sur le port 32542.

I. Index