Linux — Rappel : fichiers, chemins et permissions

1.1 — Qu'est-ce que Linux ?

Linux est le **noyau** (le cœur du système, celui qui gère le matériel : processeur, mémoire, disques, réseau). Ce qu'on appelle "une distribution Linux" (Ubuntu, Debian, Fedora, etc.) est un ensemble composé de ce noyau, plus des outils, des bibliothèques et des logiciels autour.

Un principe central à retenir : **sous Linux, presque tout est représenté comme un fichier** — les fichiers classiques bien sûr, mais aussi les dossiers, les périphériques matériels (un disque dur, une imprimante), et même certaines informations sur l'état du système. C'est ce qui permet d'utiliser les mêmes commandes de base pour manipuler des choses très différentes.

Linux est aussi un système **multi-utilisateur** et **multitâche** : plusieurs personnes peuvent l'utiliser en même temps, et plusieurs programmes tournent simultanément, chacun isolé des autres par des permissions.

1.2 — Le système de fichiers

Contrairement à Windows (qui a un `C:`, un `D:`, etc.), Linux n'a qu'**une seule arborescence**, qui part d'un unique point appelé la **racine**, notée `/`. Tout le reste — y compris d'autres disques ou clés USB — vient se "brancher" (on dit se *monter*) quelque part dans cette même arborescence.

Quelques dossiers importants à connaître dès le début :

Dossier	Rôle
`/`	La racine, le point de départ de tout
`/home/`	Contient le répertoire personnel de chaque utilisateur (ex : `/home/alice`)
`/etc/`	Fichiers de configuration du système
`/var/`	Données qui changent souvent (journaux, files d'attente...)
`/tmp/`	Fichiers temporaires, effacés régulièrement
`/bin/`, `/usr/bin/`	Programmes exécutables du système
`/root/`	Répertoire personnel de l'utilisateur administrateur (`root`)

Ton **répertoire personnel** (souvent noté `~`) est l'endroit qui t'appartient et où tu as normalement tous les droits.

1.3 — Structure d'une commande

Une commande Linux suit presque toujours ce schéma :

```
commande [options] [arguments]
```

- La **commande** est le nom du programme à exécuter.
- Les **options** modifient son comportement. Elles commencent par un tiret : une lettre seule (`-a`) ou un mot complet (`--all`). Plusieurs options courtes peuvent souvent se combiner (`-l -a -h` devient `-lah`).
- Les **arguments** sont ce sur quoi la commande travaille (un nom de fichier, un chemin...).

Exemple : dans `ls -lah /home`, `ls` est la commande, `-lah` regroupe trois options, `/home` est l'argument (le dossier à lister).

Réflexe à toujours avoir en cas de doute : la plupart des commandes ont une aide intégrée, accessible en ajoutant `--help` après la commande, ou en consultant son manuel complet. C'est plus fiable que de deviner ou de chercher au hasard sur internet.

1.4 — Gestion des utilisateurs

Chaque personne qui utilise le système a un **compte utilisateur**, identifié par un nom et, en interne, par un numéro (UID). Chaque utilisateur appartient à un ou plusieurs **groupes** (identifiés eux aussi par un numéro, le GID), ce qui permet de partager des droits entre plusieurs personnes sans les gérer une par une.

Le système garde la liste des utilisateurs et des groupes dans des fichiers de configuration dédiés, que tu croiseras tôt ou tard.

Il existe un utilisateur particulier, `root` (parfois appelé le "superutilisateur"), qui a tous les droits sur le système, sans restriction. Par sécurité, on ne travaille jamais directement en tant que `root` au quotidien : on utilise un mécanisme d'élévation temporaire de privilèges quand c'est nécessaire, pour une seule commande à la fois.

1.5 — Chemins absolus et chemins relatifs

Un **chemin** décrit l'emplacement d'un fichier ou d'un dossier dans l'arborescence.

- Un **chemin absolu** part toujours de la racine `/`. Il fonctionne **peu importe l'endroit où tu te trouves** au moment où tu le tapes. Exemple : `/home/alice/projet/notes.txt`.
- Un **chemin relatif** part de **l'endroit où tu es actuellement**. Il change de sens selon ta position. Exemple : si tu es dans `/home/alice`, le chemin relatif `projet/notes.txt` désigne la même chose que l'absolu ci-dessus — mais si tu es ailleurs, ce même chemin relatif désignera autre chose, ou n'existera pas.

Quelques notations à connaître :

- `.` désigne le dossier courant (celui où tu es).
- `..` désigne le dossier parent (un niveau au-dessus).
- `~` désigne ton répertoire personnel, où que tu sois.

La règle d'or à retenir pour toute la suite de ce document : un chemin relatif n'a de sens que si tu sais avec certitude où tu te trouves au moment où tu l'écris. Dans le doute, vérifie avant d'agir — ne suppose jamais.

1.6 — Les permissions

Chaque fichier et chaque dossier appartient à **un utilisateur** (le propriétaire) et à **un groupe**. Trois catégories de personnes peuvent avoir des droits différents sur un même élément :

- le **propriétaire**,
- les membres du **groupe** associé,
- **tous les autres** utilisateurs du système.

Pour chacune de ces trois catégories, trois droits peuvent être accordés ou non :

- **r** (read / lecture) : voir le contenu d'un fichier, ou lister le contenu d'un dossier.
- **w** (write / écriture) : modifier un fichier, ou créer/supprimer des éléments dans un dossier.
- **x** (execute / exécution) : lancer un fichier comme un programme, ou "entrer" dans un dossier (s'y déplacer, y accéder).

Ces droits peuvent s'écrire de deux façons :

- **Symbolique** : une suite de lettres, ex. `rwxr-xr--`.
- **Numérique (octale)** : chaque droit vaut un chiffre (r = 4, w = 2, x = 1), on additionne pour chaque catégorie. `rwx` = 7, `r-x` = 5, `r--` = 4 → la ligne ci-dessus se note `754`.

Point souvent mal compris : sur un **dossier**, le droit d'exécution (`x`) ne veut pas dire "lancer le dossier comme un programme" (ça n'a pas de sens), mais **pouvoir y accéder/le traverser**. Un

dossier lisible (`r`) mais sans exécution (`x`) permet de voir les noms des fichiers qu'il contient, mais pas d'y entrer ni d'accéder à leur contenu.

1.7 — Décomposer une ligne de `ls -lah`

Voici un exemple de ligne obtenue avec `ls -lah`, et ce que chaque partie signifie :

```
-rw-r--r-- 1 alice devteam 1.2K sept. 10 10:00 rapport.txt
```

Portion	Signification
`-` (premier caractère)	Type d'élément : `-` = fichier normal, `d` = dossier, `l` = lien symbolique
`rw-`	Droits du **propriétaire** : lecture + écriture, pas d'exécution
`r--`	Droits du **groupe** : lecture seule
`r--`	Droits des **autres** : lecture seule
`1`	Nombre de liens physiques vers cet élément (notion avancée, à ne pas creuser pour l'instant)
`alice`	Le **propriétaire** du fichier
`devteam`	Le **groupe** associé au fichier
`1.2K`	La taille (ici en kilo-octets, grâce à l'option qui rend la taille lisible)
`sept. 10 10:00`	Date et heure de dernière modification
`rapport.txt`	Le nom de l'élément

Entraîne-toi à relire cette ligne plusieurs fois jusqu'à ce que tu puisses la décomposer sans hésiter, dans les deux notations (symbolique et octale).