

1 - Découvrir Linux et le terminal

Comprendre le shell, la syntaxe des commandes, l'aide intégrée et l'élévation de privilèges.

- [1.1 - Premiers repères dans Linux](#)
- [1.2 - Syntaxe d'une commande](#)
- [1.3 - Obtenir de l'aide](#)
- [1.4 - Utilisateur, root, su et sudo](#)

1.1 - Premiers repères dans Linux

Linux se pilote avec une interface graphique, mais aussi avec un **terminal**. Dans ce terminal, un programme appelé **shell** interprète les commandes saisies. Bash est très répandu ; Zsh, Fish et Dash sont d'autres shells possibles.

Identifier sa session

```
whoami          # utilisateur courant
hostname        # nom de la machine
pwd             # dossier de travail actuel
echo "$SHELL"   # shell configuré pour le compte
printf '%s\n' "$0" # shell ou processus de la session actuelle
```

Le symbole affiché avant la saisie est l'**invite de commande**. Par convention, `$` représente un utilisateur normal et `#` le compte `root`. Il ne faut pas recopier ces symboles lorsqu'ils apparaissent dans une documentation.

Comprendre l'arborescence

Linux utilise une arborescence unique dont la racine est `/`.

Chemin	Rôle habituel
<code>/home</code>	dossiers personnels des utilisateurs
<code>/root</code>	dossier personnel de l'administrateur <code>root</code>
<code>/etc</code>	configuration du système et des services
<code>/var</code>	données variables : journaux, caches, files d'attente
<code>/tmp</code>	fichiers temporaires, souvent nettoyés automatiquement
<code>/usr</code>	programmes, bibliothèques et données partagées
<code>/dev</code>	périphériques représentés sous forme de fichiers

Chemin	Rôle habituel
<code>/proc</code> et <code>/sys</code>	informations virtuelles sur le noyau et le matériel
<code>/mnt</code> et <code>/media</code>	points de montage des stockages

Afficher les principaux dossiers :

```
ls -la /
```

Chemins absolus et relatifs

- `/etc/hosts` est un **chemin absolu** : il part de `/`.
- `Documents/notes.txt` est un **chemin relatif** : il part du dossier actuel.
- `.` représente le dossier actuel.
- `..` représente le dossier parent.
- `~` représente le dossier personnel de l'utilisateur.

```
cd /etc          # chemin absolu
cd ../var        # chemin relatif
cd ~             # retour au dossier personnel
```

Linux distingue généralement les majuscules des minuscules : `Rapport.txt` et `rapport.txt` sont donc deux noms différents.

Historique et confort de saisie

```
history          # afficher l'historique
history | tail -n 20 # afficher les 20 dernières commandes
clear            # nettoyer l'affichage du terminal
```

- Flèche `↑` : rappeler une ancienne commande.
- `Tab` : compléter un chemin ou une commande.
- `Ctrl` + `C` : interrompre le programme au premier plan.
- `Ctrl` + `L` : nettoyer l'écran sans effacer l'historique.
- `Ctrl` + `D` : fermer le shell courant ou signaler la fin de saisie.

1.2 - Syntaxe d'une commande

Une commande Linux suit habituellement cette forme :

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

Exemple :

```
ls -l --human-readable /var/log
```

Élément	Valeur	Rôle
commande	ls	programme exécuté
option courte	-l	affichage détaillé
option longue	--human-readable	tailles lisibles
argument	/var/log	dossier traité

Options courtes et longues

Certaines options courtes peuvent être regroupées :

```
ls -l -a -h  
ls -lah
```

Toutes les commandes ne suivent pas exactement les mêmes conventions. Il faut consulter `--help` ou la page de manuel avant de supposer qu'une option existe.

Espaces, guillemets et caractères spéciaux

Un espace sépare les arguments. Un chemin contenant des espaces doit être protégé :

```
cd "Mes documents"
cp 'rapport final.txt' sauvegarde/
cd Mes\ documents
```

Les guillemets simples conservent littéralement leur contenu. Les guillemets doubles permettent encore le remplacement des variables :

```
nom="Alex"
echo 'Bonjour $nom'      # affiche : Bonjour $nom
echo "Bonjour $nom"     # affiche : Bonjour Alex
```

Motifs génériques

Le shell développe certains motifs avant d'exécuter la commande :

Motif	Signification	Exemple
*	zéro ou plusieurs caractères	<code>ls *.log</code>
?	exactement un caractère	<code>ls image?.jpg</code>
[abc]	un caractère de la liste	<code>ls fichier[123].txt</code>
{a,b}	plusieurs valeurs générées	<code>mkdir -p projet/{docs,scripts}</code>

Toujours vérifier un motif avant une suppression :

```
printf '%s\n' *.tmp
# Puis seulement si la liste est correcte :
rm -- *.tmp
```

Codes de retour

Une commande renvoie un code : 0 signifie généralement réussite, une autre valeur signale une erreur ou un résultat particulier.

```
ls /etc/hosts
echo "$?"
```

Enchaînements utiles :

```
commande1 && commande2    # commande2 seulement si commande1 réussit  
commande1 || commande2    # commande2 seulement si commande1 échoue  
commande1 ; commande2     # commande2 dans tous les cas
```

Exemple prudent :

```
mkdir sauvegarde && cp -a configuration/ sauvegarde/
```

Commandes internes et exécutables

```
type cd  
type ls  
command -v python3  
which python3          # courant, mais command -v est plus portable
```

`type` indique si le nom correspond à une commande interne, un alias, une fonction ou un exécutable.

1.3 - Obtenir de l'aide

Linux fournit l'essentiel de sa documentation directement dans le système. Il est préférable de la consulter, car les options peuvent varier selon la distribution et la version installée.

Aide rapide

```
ls --help
ip --help
systemctl --help
```

Pour une commande interne au shell Bash :

```
help cd
help history
```

Pages de manuel

```
man ls
man chmod
man systemctl
```

Dans `man` :

Touche	Action
<code>Espace</code> ou <code>Page↓</code>	page suivante
<code>b</code> ou <code>Page↑</code>	page précédente
<code>/mot</code>	rechercher vers le bas
<code>n</code> / <code>N</code>	occurrence suivante/précédente
<code>g</code> / <code>G</code>	début/fin du manuel
<code>q</code>	quitter

Les manuels sont répartis en sections. Les plus courantes sont :

Section	Contenu
1	commandes utilisateur
2	appels système
5	formats de fichiers et configurations
8	commandes d'administration

Si un même nom existe dans plusieurs sections :

```
man 1 passwd      # commande passwd
man 5 passwd      # format du fichier /etc/passwd
```

Rechercher une commande

```
apropos "espace disque"
man -k "network interface"
whatis rsync
```

La base de données des manuels doit parfois être générée avec `sudo mandb`.

Documentation Info et fichiers installés

```
info coreutils
info bash
```

Selon la distribution, les exemples et notes complémentaires se trouvent dans :

```
ls /usr/share/doc
```

Rechercher le paquet auquel appartient une commande :

```
# Debian et Ubuntu
dpkg -S /usr/bin/rsync
```

```
# Arch Linux  
pacman -Qo /usr/bin/rsync
```

```
# Fedora  
rpm -qf /usr/bin/rsync
```

Bonne méthode

1. Identifier précisément la commande avec `type` ou `command -v`.
2. Lire son résumé avec `--help`.
3. Consulter les exemples et avertissements dans `man`.
4. Tester d'abord sur un fichier ou un dossier non critique.
5. Vérifier le code de retour et le résultat obtenu.

Pour les commandes sensibles, ne copiez jamais une ligne trouvée en ligne sans comprendre les chemins, les variables et les droits qu'elle utilise.

1.4 - Utilisateur, root, su et sudo

Linux sépare les tâches courantes des opérations d'administration. Un utilisateur normal limite l'impact d'une erreur ; `root` possède presque tous les droits sur le système.

Identifier l'utilisateur et ses groupes

```
whoami  
id  
groups
```

Afficher les sessions ouvertes :

```
who  
w
```

Exécuter une commande avec sudo

```
sudo systemctl restart ssh  
sudo apt update
```

`sudo` exécute uniquement la commande qui suit avec des privilèges élevés. Le mot de passe demandé est normalement celui de l'utilisateur courant.

Tester ses autorisations :

```
sudo -v      # valider ou renouveler l'authentification
sudo -l      # lister les commandes autorisées
sudo -k      # oublier immédiatement l'authentification en cache
```

Ouvrir un shell d'administration seulement lorsque cela est nécessaire :

```
sudo -i
exit
```

Différence entre su et sudo

```
su - utilisateur
su -
```

- `su - utilisateur` ouvre une session de connexion pour cet utilisateur.
- `su -` tente d'ouvrir une session `root` et demande généralement le mot de passe de `root`.
- `sudo -i` utilise les règles `sudo` et conserve une trace des commandes lancées.

Sur certaines distributions, le compte `root` n'a volontairement pas de mot de passe utilisable ; l'administration passe alors par `sudo`.

Redirections et privilèges

Cette commande peut échouer, car la redirection `>` est réalisée par le shell normal avant `sudo` :

```
sudo echo "127.0.0.1 exemple.local" > /etc/hosts
```

Utiliser `tee` :

```
echo "127.0.0.1 exemple.local" | sudo tee -a /etc/hosts
```

Pour modifier un fichier système de manière contrôlée :

```
sudoedit /etc/hosts
```

`sudoedit` copie temporairement le fichier, ouvre l'éditeur avec l'identité normale, puis remet le résultat en place avec les droits requis.

Règles de prudence

- Rester utilisateur normal pour les tâches habituelles.
- Vérifier les chemins avant toute commande précédée de `sudo`.
- Ne jamais lancer un script inconnu avec `curl ... | sudo sh`.
- Ne pas modifier `/etc/sudoers` directement ; utiliser `sudo visudo`.
- Quitter un shell `root` dès la tâche terminée.
- Préférer une commande précise à une session administrateur prolongée.