

Linux Commandes de base

Guide pratique pour découvrir le terminal Linux, manipuler les fichiers, gérer les permissions, les paquets, les services, le réseau et diagnostiquer les problèmes courants.

- [1 - Découvrir Linux et le terminal](#)
 - [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](#)
- [2 - Naviguer et gérer les fichiers](#)
 - [2.1 - Se repérer avec pwd, ls et cd](#)
 - [2.2 - Créer, copier, déplacer et renommer](#)
 - [2.3 - Supprimer sans se tromper](#)
 - [2.4 - Types de fichiers et liens](#)
- [3 - Lire, rechercher et traiter du texte](#)
 - [3.1 - Lire un fichier avec cat, less, head et tail](#)
 - [3.2 - Rechercher avec grep, rg, find et locate](#)
 - [3.3 - Redirections, tubes et tee](#)
 - [3.4 - Trier et transformer avec sort, uniq, cut, sed et awk](#)
- [4 - Permissions, propriétaires et comptes](#)
 - [4.1 - Comprendre les permissions Linux](#)
 - [4.2 - Modifier les droits avec chmod](#)
 - [4.3 - Propriétaires, groupes, chown et chgrp](#)

- [4.4 - Gérer les utilisateurs et les groupes](#)
- [5 - Logiciels, processus, services et journaux](#)
 - [5.1 - Installer des logiciels avec apt, pacman et dnf](#)
 - [5.2 - Surveiller et arrêter les processus](#)
 - [5.3 - Gérer les services avec systemctl](#)
 - [5.4 - Consulter les journaux avec journalctl et dmesg](#)
- [6 - Réseau, Web et accès distant](#)
 - [6.1 - Adresses IP et tests de connectivité](#)
 - [6.2 - DNS, routes et ports réseau](#)
 - [6.3 - Requêtes et téléchargements avec curl et wget](#)
 - [6.4 - SSH, SCP et rsync](#)
- [7 - Disques, système et dépannage](#)
 - [7.1 - Espace disque avec lsblk, df et du](#)
 - [7.2 - Monter, démonter et archiver](#)
 - [7.3 - Informations système et matériel](#)
 - [7.4 - Méthode de dépannage et mémo des commandes](#)

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

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

Chemin	Rôle habituel
<code>/dev</code>	périphériques représentés sous forme de fichiers
<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	<code>ls</code>	programme exécuté
option courte	<code>-l</code>	affichage détaillé
option longue	<code>--human-readable</code>	tailles lisibles
argument	<code>/var/log</code>	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.

2 - Naviguer et gérer les fichiers

Se déplacer dans l'arborescence et manipuler fichiers, dossiers et liens sans risque.

2.1 - Se repérer avec pwd, ls et cd

Le shell exécute les commandes dans un **dossier de travail courant**. Bien savoir où l'on se trouve évite de modifier le mauvais fichier.

Afficher le dossier courant

```
pwd
pwd -P      # résoudre les liens symboliques dans le chemin
```

Lister le contenu

```
ls
ls -l      # détails : droits, propriétaire, taille et date
ls -a      # inclure les noms commençant par un point
ls -lah    # détails, fichiers cachés et tailles lisibles
ls -lt     # trier par date de modification
ls -ltr    # plus ancien en premier
ls -ld /etc # afficher le dossier lui-même, pas son contenu
```

Dans une sortie `ls -l`, le premier caractère indique le type : `d` pour un dossier, `-` pour un fichier ordinaire et `l` pour un lien symbolique.

Changer de dossier

```
cd /var/log      # chemin absolu
cd ../lib        # dossier parent, puis lib
cd ~             # dossier personnel
```

```
cd          # identique à cd ~
cd -        # revenir au dossier précédent
```

Les raccourcis essentiels :

Chemin	Signification
<code>.</code>	dossier courant
<code>..</code>	dossier parent
<code>~</code>	dossier personnel
<code>-</code> avec <code>cd</code>	dossier visité précédemment

Afficher une arborescence

Si la commande `tree` est installée :

```
tree
tree -L 2
tree -a -L 2 /etc
```

Sans `tree` :

```
find . -maxdepth 2 -print
```

Noms cachés et complétion

Sous Linux, un nom commençant par `.` est caché par convention : `.bashrc`, `.config`, `.ssh`, etc. Il n'est pas secret et reste visible avec `ls -a`.

Utilisez `Tab` pour compléter les chemins. Deux pressions peuvent afficher les possibilités. Cette méthode réduit fortement les erreurs de frappe.

Exemple de navigation contrôlée

```
cd /var/log
pwd
```

```
ls -lah
```

```
cd -
```

Avant une commande sensible, associer `pwd` et `ls` constitue un bon réflexe :

```
pwd
```

```
ls -la -- .
```

Le séparateur `--` indique la fin des options. Il protège notamment contre un fichier dont le nom commence par `-`.

2.2 - Créer, copier, déplacer et renommer

Créer des dossiers et des fichiers

```
mkdir projet
mkdir -p projet/{docs,scripts,sauvegardes}
touch notes.txt
touch "compte rendu.txt"
```

`mkdir -p` crée aussi les dossiers parents manquants et ne signale pas d'erreur si le dossier existe déjà. `touch` crée un fichier vide ou actualise sa date de modification.

Copier

```
cp source.txt copie.txt
cp source.txt sauvegardes/
cp -i source.txt copie.txt      # demander avant d'écraser
cp -a dossier/ sauvegarde/      # copie récursive en préservant les attributs
cp -av dossier/ sauvegarde/     # mode détaillé
```

Attention à la barre finale :

```
cp -a dossier sauvegarde/      # crée sauvegarde/dossier si sauvegarde existe
cp -a dossier/. sauvegarde/    # copie le contenu de dossier dans sauvegarde
```

Déplacer ou renommer

La même commande réalise les deux opérations :

```
mv ancien.txt nouveau.txt
mv rapport.pdf Documents/
mv -i fichier.txt destination/
mv -n fichier.txt destination/    # ne jamais écraser
```

Créer rapidement du contenu

```
printf '%s\n' "Première ligne" "Deuxième ligne" > notes.txt
echo "Ligne ajoutée" >> notes.txt
```

> remplace le contenu existant. >> ajoute à la fin.

Comparer avant de remplacer

```
cmp fichier1.bin fichier2.bin
diff -u ancienne.conf nouvelle.conf
```

Pour synchroniser des dossiers, `rsync` offre un mode de simulation très utile :

```
rsync -avhn source/ destination/ # simulation
rsync -avh source/ destination/  # exécution
```

Gérer les noms particuliers

Protéger les espaces :

```
cp -- "rapport final.txt" "rapport final - copie.txt"
```

Traiter un nom commençant par un tiret :

```
mv -- -ancien-fichier nouveau-fichier
```

Vérifications recommandées

```
file sauvegarde.tar.gz      # identifier le type réel
stat fichier.txt            # métadonnées détaillées
ls -lah destination/
```

Pour une donnée importante, une copie locale n'est pas forcément une sauvegarde. Une vraie sauvegarde doit idéalement être stockée sur un autre support et testée par une restauration.

2.3 - Supprimer sans se tromper

Sous Linux, `rm` ne place généralement pas les éléments dans une corbeille. Une suppression en ligne de commande peut donc être immédiate et difficile à récupérer.

Supprimer un fichier ou un dossier vide

```
rm fichier.tmp  
rm -i fichier.tmp      # demander confirmation  
rmdir dossier-vide
```

Supprimer plusieurs fichiers précis :

```
rm -- fichier1.tmp fichier2.tmp
```

Supprimer un dossier et son contenu

```
rm -r ancien-dossier/  
rm -ri ancien-dossier/ # confirmation interactive
```

L'option `-f` force la suppression et réduit les protections. La combinaison `rm -rf` doit rester exceptionnelle et ne doit être utilisée qu'après vérification exacte de la cible.

Méthode sûre avec un motif

Avant de supprimer tous les fichiers `.tmp`, afficher ce que le shell sélectionne :

```
printf '%s\n' -- *.tmp
```

Puis supprimer seulement si la liste est correcte :

```
rm -i -- *.tmp
```

Avec `find`, commencer par `-print` :

```
find /var/tmp/mon-application -type f -name '*.tmp' -print
```

Après contrôle :

```
find /var/tmp/mon-application -type f -name '*.tmp' -delete
```

L'ordre des critères de `find` compte : `-delete` ne doit être ajouté qu'à la fin, après un test réussi avec `-print`.

Préférer une opération récupérable

Si l'environnement graphique fournit `gio` :

```
gio trash fichier.txt  
gio trash --list
```

Autre méthode : déplacer d'abord l'élément vers un dossier de quarantaine.

```
mkdir -p ~/A_supprimer  
mv -i ancien-dossier ~/A_supprimer/
```

Erreurs à éviter

- Ne jamais exécuter une suppression récursive sur `/`, `/home`, `~` ou une variable non vérifiée.
- Toujours mettre les variables entre guillemets : `rm -- "$fichier"`.

- Ne pas supposer qu'une variable est définie.
- Vérifier `pwd`, puis la cible avec `ls -ld -- chemin`.
- Ne pas précéder `rm` de `sudo` si les droits normaux suffisent.

Exemple de contrôle d'une variable :

```
cible="/var/tmp/mon-application"  
test -n "$cible" && test "$cible" != "/" && ls -ld -- "$cible"
```

Cette vérification n'autorise pas automatiquement la suppression ; elle sert seulement à examiner la cible.

2.4 - Types de fichiers et liens

Linux représente de nombreuses ressources sous forme de fichiers : documents, dossiers, périphériques, sockets et liens.

Identifier un type de fichier

```
file document.pdf
file image-sans-extension
stat document.pdf
ls -l document.pdf
```

Le début de `ls -l` indique le type :

Caractère	Type
-	fichier ordinaire
d	dossier
l	lien symbolique
b	périphérique bloc, par exemple un disque
c	périphérique caractère
s	socket
p	tube nommé

Lien symbolique

Un lien symbolique contient un chemin vers une autre ressource. Il ressemble à un raccourci.

```
ln -s /srv/app/config.yml ~/config-app.yml
ls -l ~/config-app.yml
readlink ~/config-app.yml
readlink -f ~/config-app.yml
```

Si la cible est déplacée ou supprimée, le lien devient cassé. Rechercher les liens cassés sous le dossier actuel :

```
find . -xtype l -print
```

Lien physique

Un lien physique est un second nom vers les mêmes données sur le même système de fichiers.

```
ln rapport.txt rapport-alias.txt
ls -li rapport.txt rapport-alias.txt
```

Les deux noms affichent le même numéro d'inode. Modifier l'un modifie le contenu visible par l'autre. Supprimer un nom ne détruit les données que lorsqu'aucun lien physique ne les référence plus et qu'aucun processus ne les maintient ouvertes.

Les liens physiques ne traversent généralement pas les systèmes de fichiers et ne s'appliquent normalement pas aux dossiers.

Extension et contenu réel

Linux ne dépend pas uniquement de l'extension :

```
file sauvegarde
file --mime-type sauvegarde
```

Un fichier nommé `.jpg` peut ne pas être une image. `file` examine sa signature, sans garantir pour autant qu'il est sûr.

Informations utiles avec stat

```
stat -c 'Nom=%n Taille=%s octets Droits=%A Propriétaire=%U Groupe=%G' fichier
```

Sur BSD ou macOS, les options de `stat` diffèrent. Les exemples ci-dessus visent les outils GNU courants sous Linux.

3 - Lire, rechercher et traiter du texte

Afficher, filtrer, rechercher et transformer efficacement des fichiers texte.

3.1 - Lire un fichier avec cat, less, head et tail

Afficher un petit fichier

```
cat /etc/os-release  
cat -n notes.txt          # numéroté les lignes
```

`cat` affiche tout le contenu d'un coup. Il convient aux petits fichiers, mais devient peu pratique pour un journal de plusieurs gigaoctets.

Assembler plusieurs fichiers :

```
cat partiel1.txt partie2.txt > document-complet.txt
```

Parcourir avec less

```
less /var/log/syslog  
less +F /var/log/syslog    # suivre les ajouts ; Ctrl+C pour revenir à less
```

Touches essentielles :

Touche	Action
Espace / b	page suivante / précédente
g / G	début / fin
/erreur	rechercher vers le bas
n / N	occurrence suivante / précédente
F	suivre les nouvelles lignes
q	quitter

`less` ne charge pas nécessairement tout le fichier en mémoire et convient donc mieux aux gros fichiers.

Début et fin d'un fichier

```
head fichier.log
head -n 20 fichier.log
tail fichier.log
tail -n 50 fichier.log
```

Suivre un journal en direct :

```
tail -f application.log
tail -F application.log
```

`-F` continue de suivre le nom même si le fichier est recréé lors d'une rotation de journaux. Quitter avec `Ctrl` + `C`.

Extraire une plage de lignes

```
sed -n '20,40p' fichier.txt
```

Cette commande affiche les lignes 20 à 40 sans modifier le fichier.

Afficher les caractères invisibles

```
cat -A fichier.txt
sed -n 'l' fichier.txt
```

Utile pour repérer tabulations, espaces finaux et fins de ligne Windows.

Encodage et fins de ligne

```
file -i fichier.txt
```

Convertir un fichier Windows CRLF vers Linux LF si `dos2unix` est installé :

```
dos2unix script.sh
```

Avant de lancer un script téléchargé, l'afficher et l'examiner :

```
less script.sh  
file script.sh
```

Afficher un fichier est normalement non destructif, mais son contenu peut comporter des séquences de terminal indésirables. Pour une source non fiable, `less` est souvent préférable à l'exécution ou au copier-coller aveugle.

3.2 - Rechercher avec grep, rg, find et locate

Rechercher du texte avec grep

```
grep 'erreur' application.log
grep -i 'erreur' application.log      # ignorer la casse
grep -n 'erreur' application.log      # afficher les numéros de ligne
grep -v '^#' fichier.conf            # exclure les commentaires
grep -RIn '192.168.25.1' /etc        # recherche récursive
```

Afficher du contexte :

```
grep -n -C 3 'failed' service.log     # 3 lignes avant et après
grep -n -A 5 'failed' service.log     # 5 lignes après
grep -n -B 2 'failed' service.log     # 2 lignes avant
```

Expressions régulières étendues :

```
grep -En 'error|failed|denied' application.log
grep -E '^[0-9]{4}-[0-9]{2}-[0-9]{2}' application.log
```

Rechercher rapidement avec rg

`rg` désigne **ripgrep**. Il est très rapide et ignore par défaut certains fichiers cachés ou exclus par `.gitignore`.

```
rg 'motif' .
rg -n -i 'motif' /etc
rg --hidden 'motif' .
rg -g '*.yaml' 'password' .
```

Il n'est pas installé par défaut partout. Utiliser `grep -R` comme solution universelle.

Rechercher des fichiers avec find

```
find . -type f -name '*.log'
find /var/log -type f -iname '*error*'
find . -type d -name 'cache'
find . -type f -size +100M -print
find . -type f -mtime -7 -print      # modifiés depuis moins de 7 jours
find . -maxdepth 2 -type f -print
```

Exécuter une commande sur chaque résultat :

```
find . -type f -name '*.conf' -exec file {} \;
```

Pour les noms contenant des espaces ou retours à la ligne, préférer des séparateurs nuls :

```
find . -type f -print0 | xargs -0 file
```

Recherche indexée avec locate

```
locate sshd_config
sudo updatedb
```

`locate` interroge une base mise à jour périodiquement : il est rapide, mais peut afficher un fichier qui vient d'être supprimé ou manquer un fichier récent.

Rechercher une commande disponible

```
command -v ssh
type -a python
```

Toujours délimiter précisément une recherche sous  : elle peut être lente, rencontrer des permissions refusées et parcourir des systèmes de fichiers distants.

3.3 - Redirections, tubes et tee

Les commandes utilisent trois flux standards :

Numéro	Flux	Rôle
0	entrée standard, <code>stdin</code>	données reçues par la commande
1	sortie standard, <code>stdout</code>	résultat normal
2	erreur standard, <code>stderr</code>	avertissements et erreurs

Rediriger la sortie

```
commande > resultat.txt      # remplacer le fichier
commande >> resultat.txt     # ajouter à la fin
commande 2> erreurs.txt      # erreurs seulement
commande > tout.txt 2>&1      # sortie et erreurs dans le même fichier
commande &> tout.txt         # raccourci Bash
```

`>` vide le fichier avant d'exécuter la commande. Vérifier le chemin et préférer `>>` lorsqu'un historique doit être conservé.

Ignorer une sortie :

```
commande > /dev/null
commande > /dev/null 2>&1
```

Fournir une entrée

```
sort < noms.txt
```

Document intégré :

```
cat <<'EOF'  
Le contenu $VARIABLE reste littéral  
Deuxième ligne  
EOF
```

Relier des commandes avec un tube

Le tube `|` transmet la sortie standard d'une commande à l'entrée de la suivante.

```
ps aux | grep '[s]shd'  
journalctl -b | grep -i 'error'  
find . -type f | wc -l
```

Une chaîne se lit de gauche à droite : produire, filtrer, transformer puis résumer.

```
cut -d: -f1 /etc/passwd | sort | uniq
```

Voir et enregistrer avec tee

```
ip addr | tee diagnostic-reseau.txt  
ip addr | tee -a diagnostic-complet.txt
```

Écrire dans un fichier protégé :

```
printf '%s\n' 'option=valeur' | sudo tee /etc/application.conf
```

`tee` affiche aussi ce qu'il écrit. Ajouter `> /dev/null` pour masquer cette copie d'écran.

Codes d'erreur dans un pipeline

Par défaut, le shell retourne souvent le code de la dernière commande du pipeline. Dans un script Bash sensible :

```
set -o pipefail
commande1 | commande2
echo "$?"
```

`pipefail` signale l'échec d'une commande intermédiaire. Dans un terminal, voir les codes de tout le dernier pipeline :

```
printf '%s\n' "${PIPESTATUS[@]}"
```

Ne mélangez pas au hasard `sudo`, les redirections et les tubes : chaque partie peut s'exécuter avec des privilèges différents.

3.4 - Trier et transformer avec sort, uniq, cut, sed et awk

Les petits outils Unix sont conçus pour être combinés. Ils permettent d'extraire une information sans ouvrir un tableur ou écrire un programme complet.

Compter avec wc

```
wc fichier.txt
wc -l fichier.txt      # lignes
wc -w fichier.txt      # mots
wc -c fichier.txt      # octets
```

Trier et supprimer les doublons

```
sort noms.txt
sort -n nombres.txt
sort -h tailles.txt
sort -r noms.txt
sort noms.txt | uniq
sort noms.txt | uniq -c | sort -nr
```

`uniq` ne regroupe que les lignes identiques adjacentes ; il est donc souvent précédé de `sort`.

Extraire des colonnes

```
cut -d: -f1 /etc/passwd  
cut -d';' -f1,3 inventaire.csv
```

`cut` convient aux séparateurs simples. Un CSV avec des guillemets, des retours à la ligne ou des séparateurs échappés nécessite un véritable analyseur CSV.

Remplacer des caractères avec tr

```
printf '%s\n' 'bonjour' | tr '[:lower:]' '[:upper:]'  
tr -d '\r' < windows.txt > linux.txt  
tr ':' '\n' < /etc/passwd
```

Transformer avec sed

Afficher sans modifier :

```
sed -n '1,10p' fichier.txt  
sed 's/ancien/nouveau/g' fichier.txt  
sed '/^#/d; /^$/d' fichier.conf
```

Créer un nouveau fichier :

```
sed 's/ancien/nouveau/g' fichier.txt > fichier-modifie.txt
```

La modification directe `sed -i` varie entre GNU/Linux et BSD/macOS. Créer d'abord une sauvegarde :

```
sed -i.bak 's/ancien/nouveau/g' fichier.txt
```

Extraire avec awk

```
awk -F: '{print $1}' /etc/passwd  
awk '{print NR, $0}' fichier.txt  
awk '$3 > 80 {print $1, $3}' mesures.txt  
df -P | awk 'NR>1 {print $6, $5}'
```

Dans `awk`, `$1` désigne le premier champ, `$0` toute la ligne et `NR` le numéro de ligne.

Exemple combiné

Compter les shells déclarés dans `/etc/passwd` :

```
cut -d: -f7 /etc/passwd | sort | uniq -c | sort -nr
```

Tester d'abord les transformations sans `-i`, comparer le résultat avec `diff`, puis remplacer le fichier uniquement après validation.

4 - Permissions, propriétaires et comptes

Lire et modifier les permissions, les propriétaires, les utilisateurs et les groupes.

4.1 - Comprendre les permissions Linux

Chaque fichier possède un propriétaire, un groupe et des permissions pour trois catégories : **utilisateur** (u), **groupe** (g) et **autres** (o).

Lire les droits

```
ls -l fichier.txt
ls -ld dossier
stat fichier.txt
```

Exemple :

```
-rw-r----- 1 alex informatique 1250 août 7 10:00 rapport.txt
```

Décomposition :

Bloc	Signification
-	fichier ordinaire
rw-	propriétaire : lecture et écriture
r--	groupe : lecture
---	autres : aucun droit

Signification sur un fichier

Droit	Lettre	Effet
lecture	r	lire le contenu
écriture	w	modifier le contenu

Droit	Lettre	Effet
exécution	x	lancer le fichier comme programme

Signification sur un dossier

Droit	Effet
r	lister les noms présents
w	créer, renommer ou supprimer des entrées
x	traverser le dossier et accéder aux éléments connus

Les permissions d'un dossier expliquent pourquoi un utilisateur peut parfois supprimer un fichier qu'il ne peut pas modifier : la suppression agit sur l'entrée du dossier parent.

Tester un accès

```
test -r fichier && echo "lisible"
test -w fichier && echo "modifiable"
test -x script.sh && echo "exécutable"
```

Permissions par défaut et umask

```
umask
umask -S
```

La valeur `umask` retire des droits lors de la création. Une valeur courante `022` produit généralement des fichiers `644` et des dossiers `755`. Une valeur `027` limite davantage l'accès des autres utilisateurs.

Modification temporaire dans le shell courant :

```
umask 027
```

Au-delà des droits classiques

Un `+` après les permissions dans `ls -l` peut signaler des ACL :

```
getfacl fichier.txt
```

Les systèmes avec SELinux ou AppArmor peuvent aussi refuser une opération malgré des droits Unix apparemment corrects. Avant d'appliquer `chmod 777`, identifier le propriétaire, les droits du dossier parent, les ACL et le contexte de sécurité.

`777` donne lecture, écriture et exécution à tout le monde. Ce n'est presque jamais une bonne correction : cela masque la cause et augmente les risques.

4.2 - Modifier les droits avec chmod

`chmod` modifie les permissions d'un fichier ou d'un dossier. Il accepte une notation symbolique, plus lisible, ou une notation numérique.

Notation symbolique

Catégories : `u` propriétaire, `g` groupe, `o` autres, `a` tous.

Opérations : `+` ajouter, `-` retirer, `=` définir exactement.

```
chmod u+x script.sh      # ajouter l'exécution au propriétaire
chmod g+w partage/       # autoriser l'écriture au groupe
chmod o-r fichier.txt    # retirer la lecture aux autres
chmod u=rw,g=r,o= fichier # définir précisément les droits
chmod a+r documentation.pdf # lecture pour tous
```

Notation numérique

Chaque droit possède une valeur : lecture `4`, écriture `2`, exécution `1`.

Valeur	Droits	Signification
7	<code>rwX</code>	lecture, écriture, exécution
6	<code>rw-</code>	lecture et écriture
5	<code>r-X</code>	lecture et exécution
4	<code>r--</code>	lecture seule
0	<code>---</code>	aucun droit

```
chmod 644 fichier.txt      # rw-r--r--
chmod 600 secret.txt      # rw-----
chmod 755 script.sh       # rwxr-xr-x
chmod 750 dossier-prive   # rwxr-x---
```

Appliquer récursivement

```
chmod -R u+rwX,g+rX,o-rwx projet/
```

Le `X` majuscule ajoute l'exécution uniquement aux dossiers et aux fichiers déjà exécutables. Il évite de rendre exécutables tous les documents ordinaires.

Pour régler séparément fichiers et dossiers :

```
find projet -type d -exec chmod 750 {} +
find projet -type f -exec chmod 640 {} +
```

Commencer par afficher les cibles :

```
find projet -type f -print
```

Cas courant : clé SSH

```
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
chmod 600 ~/.ssh/id_ed25519
chmod 644 ~/.ssh/id_ed25519.pub
```

Vérifier

```
stat -c '%A %a %U:%G %n' fichier.txt
namei -l /chemin/vers/fichier.txt
```

`namei -l` affiche les droits de chaque composant du chemin, ce qui aide à repérer un dossier parent non traversable.

Ne pas utiliser `chmod -R 777` comme dépannage universel. Accorder seulement les droits nécessaires au bon utilisateur ou au bon groupe.

4.3 - Propriétaires, groupes, chown et chgrp

`chown` change le propriétaire et éventuellement le groupe. `chgrp` change seulement le groupe.

Examiner l'appartenance

```
ls -l fichier.txt
stat -c '%U:%G %n' fichier.txt
id alex
getent group informatique
```

Changer propriétaire ou groupe

```
sudo chown alex fichier.txt
sudo chown alex:informatique fichier.txt
sudo chown :informatique fichier.txt
sudo chgrp informatique fichier.txt
```

Appliquer récursivement :

```
sudo chown -R alex:informatique /srv/projet
```

Une opération récursive doit cibler un chemin explicite et vérifié :

```
sudo ls -ld /srv/projet
sudo find /srv/projet -maxdepth 2 -printf '%u:%g %p\n'
```

Dossier partagé par un groupe

Créer un dossier collaboratif :

```
sudo mkdir -p /srv/partage
sudo chown root:informatique /srv/partage
sudo chmod 2770 /srv/partage
```

Le chiffre spécial `2` active le bit **setgid** sur le dossier. Les nouveaux éléments héritent alors du groupe `informatique` au lieu d'utiliser systématiquement le groupe principal du créateur.

Vérifier :

```
ls -ld /srv/partage
```

La permission de groupe apparaît avec `s`, par exemple `drwxrws---`.

Bits spéciaux

Bit	Valeur	Usage courant
setuid	4	exécuter un programme avec l'identité de son propriétaire
setgid	2	héritage du groupe sur un dossier
sticky bit	1	chacun ne supprime que ses propres fichiers dans un dossier partagé

Exemple du sticky bit :

```
sudo chmod 1777 /srv/depot-temporaire
```

Le répertoire `/tmp` utilise habituellement ce principe.

Retrouver les identifiants numériques

```
id -u alex
id -g alex
getent passwd alex
getent group informatique
```

Sur un partage NFS ou dans un conteneur, les numéros UID/GID sont souvent plus importants que les noms affichés. Une différence d'identifiants peut provoquer un accès refusé malgré des noms semblables.

4.4 - Gérer les utilisateurs et les groupes

La gestion des comptes varie légèrement selon la distribution. Les commandes ci-dessous couvrent les systèmes Linux courants.

Consulter les comptes

```
id alex
getent passwd alex
getent group informatique
cut -d: -f1 /etc/passwd
```

`getent` est préférable à la lecture directe des fichiers lorsque les comptes peuvent provenir de LDAP, Active Directory ou d'un autre annuaire.

Créer un utilisateur

Sur Debian et Ubuntu, `adduser` guide la création :

```
sudo adduser alex
```

Commande plus générique :

```
sudo useradd -m -s /bin/bash alex
sudo passwd alex
```

Gérer les groupes

```
sudo groupadd informatique
sudo usermod -aG informatique alex
id alex
```

L'option `-aG` signifie **ajouter aux groupes supplémentaires**. Oublier `-a` avec `usermod -G` peut retirer l'utilisateur de ses autres groupes secondaires.

La nouvelle appartenance s'applique à la prochaine ouverture de session. Pour un test ponctuel :

```
newgrp informatique
```

Verrouiller ou déverrouiller un compte

```
sudo passwd -l alex
sudo passwd -u alex
sudo chage -l alex
```

Verrouiller le mot de passe ne termine pas forcément les sessions existantes et n'interdit pas toujours les autres méthodes d'authentification, comme une clé SSH. La procédure de départ d'un utilisateur doit donc inclure sessions, clés, jetons et tâches planifiées.

Modifier un shell

```
chsh -s /bin/bash
getent passwd "$USER"
cat /etc/shells
```

Supprimer un compte

Examiner d'abord les fichiers appartenant au compte :

```
sudo find /home /srv -xdev -user alex -print
```

Puis, selon la distribution :

```
sudo userdel alex          # conserver le dossier personnel
sudo userdel -r alex       # supprimer aussi le dossier personnel
```

`userdel -r` est destructif. Sauvegarder ou transférer les données avant de l'utiliser.

Fichiers importants

Fichier	Rôle
<code>/etc/passwd</code>	comptes et shells, sans mots de passe en clair
<code>/etc/shadow</code>	empreintes des mots de passe et expiration
<code>/etc/group</code>	groupes locaux
<code>/etc/sudoers</code>	règles sudo, à modifier avec <code>visudo</code>

Ne jamais éditer `/etc/shadow` manuellement. Utiliser les commandes prévues afin de préserver la cohérence et les permissions.

5 - Logiciels, processus, services et journaux

Installer des paquets, surveiller les processus, piloter systemd et consulter les journaux.

5.1 - Installer des logiciels avec apt, pacman et dnf

Les gestionnaires de paquets installent les logiciels, leurs dépendances et les mises à jour depuis des dépôts configurés. Utiliser celui de sa distribution évite les installations dispersées.

Identifier la distribution

```
cat /etc/os-release
```

Debian et Ubuntu : apt

```
sudo apt update           # actualiser l'index
apt list --upgradable     # voir les mises à jour
sudo apt upgrade          # appliquer les mises à jour
apt search nom
apt show paquet
sudo apt install paquet
sudo apt remove paquet
sudo apt purge paquet     # retirer aussi sa configuration système
sudo apt autoremove       # dépendances devenues inutiles
```

Retrouver le paquet d'un fichier installé :

```
dpkg -S /usr/bin/curl
dpkg -L curl
```

Arch Linux : pacman

```
sudo pacman -Syu                # synchroniser et mettre tout à jour
pacman -Ss nom                  # rechercher dans les dépôts
pacman -Si paquet               # informations dépôt
pacman -Qi paquet               # informations paquet installé
sudo pacman -S paquet           # installer
sudo pacman -Rns paquet         # retirer avec dépendances inutiles et configuration
pacman -Ql paquet               # lister les fichiers
pacman -Qo /usr/bin/curl        # propriétaire d'un fichier
```

Sur Arch, éviter les mises à jour partielles comme `pacman -Sy paquet` sans mise à niveau générale : elles peuvent rendre les bibliothèques incohérentes.

Pour les paquets AUR avec `yay`, examiner le `PKGBUILD` avant la compilation :

```
yay -S nom-du-paquet
```

Fedora : dnf

```
sudo dnf check-upgrade
sudo dnf upgrade
dnf search nom
dnf info paquet
sudo dnf install paquet
sudo dnf remove paquet
dnf provides /usr/bin/curl
```

Vérifier une installation

```
command -v curl
curl --version
```

Bonnes pratiques

- Lire la liste des paquets supprimés ou remplacés avant de confirmer.
- Ne pas mélanger sans nécessité dépôts stables, testing et dépôts tiers.

- Préférer les dépôts officiels ou une source vérifiée.
- Sauvegarder les configurations importantes avant une mise à niveau majeure.
- Ne pas interrompre une opération de paquet en cours.
- Sur un serveur, prévoir une fenêtre de maintenance et vérifier les services après la mise à jour.

Flatpak, Snap, AppImage et les conteneurs complètent le gestionnaire natif, mais possèdent leurs propres règles de mise à jour, d'isolation et de stockage.

5.2 - Surveiller et arrêter les processus

Un **processus** est une instance d'un programme en cours d'exécution. Il possède un identifiant unique, le PID, ainsi qu'un propriétaire et un état.

Lister les processus

```
ps
ps aux
ps -ef
ps -u "$USER"
ps -p 1234 -o pid,ppid,user,%cpu,%mem,etime,cmd
```

Rechercher par nom :

```
pgrep -a sshd
pidof sshd
```

Éviter `ps aux | grep nom` lorsque `pgrep -a nom` suffit.

Surveiller en direct

```
top
htop
```

Dans `top`, `P` trie par processeur, `M` par mémoire et `q` quitte. `htop` est plus convivial mais n'est pas toujours installé.

Résumé de la mémoire :

```
free -h
```

Sous Linux, une partie de la mémoire « utilisée » sert de cache récupérable. La colonne `available` est généralement plus pertinente que `free` seule.

Envoyer un signal

```
kill 1234          # SIGTERM : demande un arrêt propre
kill -15 1234      # identique
kill -9 1234       # SIGKILL : arrêt forcé par le noyau
pkill -TERM nom
```

Méthode recommandée :

1. vérifier le PID et le propriétaire ;
2. envoyer `SIGTERM` ;
3. attendre et contrôler ;
4. utiliser `SIGKILL` seulement si le processus ne répond pas.

`SIGKILL` empêche le programme de nettoyer ses fichiers temporaires ou de terminer une écriture.

Tâches au premier et à l'arrière-plan

```
commande-longue &
jobs
fg %1
bg %1
```

Suspendre le programme au premier plan avec `Ctrl + Z`, puis utiliser `bg` ou `fg`. `Ctrl + C` demande son interruption.

Garder une commande après la fermeture du terminal :

```
nohup commande-longue > commande.log 2>&1 &
```

Pour les tâches importantes, préférer un service `systemd`, `tmux` ou `screen`, qui offrent une gestion plus claire.

Priorité

```
nice -n 10 commande  
renice 10 -p 1234
```

Une valeur `nice` plus élevée réduit la priorité CPU. Des valeurs négatives demandent généralement des privilèges.

Pour un service, agir avec `systemctl` plutôt que tuer directement son processus : le gestionnaire peut sinon le relancer immédiatement.

5.3 - Gérer les services avec systemctl

Sur de nombreuses distributions, **systemd** démarre le système et gère les services. Une unité de service porte généralement un nom comme `ssh.service` ou `nginx.service`.

Examiner un service

```
systemctl status ssh
systemctl is-active ssh
systemctl is-enabled ssh
systemctl show ssh
```

Le nom peut être `sshd` au lieu de `ssh` selon la distribution :

```
systemctl list-unit-files --type=service | grep -i ssh
```

Démarrer, arrêter et recharger

```
sudo systemctl start nginx
sudo systemctl stop nginx
sudo systemctl restart nginx
sudo systemctl reload nginx
sudo systemctl reload-or-restart nginx
```

`reload` demande au service de relire sa configuration sans arrêt complet, s'il prend cette opération en charge. Après une modification, valider la syntaxe avant le rechargement :

```
sudo nginx -t && sudo systemctl reload nginx
```

Démarrage automatique

```
sudo systemctl enable nginx
sudo systemctl disable nginx
sudo systemctl enable --now nginx
sudo systemctl disable --now nginx
```

`enable` prépare le prochain démarrage mais ne lance pas forcément le service immédiatement. `--now` combine les deux actions.

Masquer un service

```
sudo systemctl mask nom.service
sudo systemctl unmask nom.service
```

`mask` empêche tout démarrage, même manuel ou demandé par une dépendance. À utiliser seulement lorsque cet effet est voulu.

État général

```
systemctl --failed
systemctl list-units --type=service --state=running
systemctl list-timers --all
```

Après modification d'une unité

```
sudo systemctl daemon-reload
sudo systemctl restart mon-service
```

`daemon-reload` est requis après modification d'un fichier `.service`, `.timer` ou d'une surcharge `systemd` ; il n'est pas nécessaire pour la configuration interne de l'application.

Créer une surcharge sans modifier le fichier fourni par le paquet :

```
sudo systemctl edit mon-service
```

Puis vérifier la configuration assemblée :

```
systemctl cat mon-service  
systemd-analyze verify /etc/systemd/system/mon-service.service
```

Redémarrer ou éteindre

```
systemctl reboot  
systemctl poweroff
```

Sur un serveur distant, confirmer qu'aucune opération de stockage ou mise à jour critique n'est en cours avant l'arrêt.

5.4 - Consulter les journaux avec journalctl et dmesg

Les journaux permettent de comprendre ce qui s'est produit avant de modifier la configuration. Avec systemd, `journalctl` interroge le journal central.

Journaux d'un service

```
journalctl -u ssh
journalctl -u nginx --since today
journalctl -u nginx -n 100
journalctl -u nginx -f
```

`-f` suit les nouvelles entrées ; quitter avec `Ctrl` + `C`.

Journaux du démarrage

```
journalctl -b          # démarrage actuel
journalctl -b -1       # démarrage précédent
journalctl --list-boots
```

Priorités importantes :

```
journalctl -p warning..alert -b
journalctl -p err -b
```

Filtrer dans le temps :

```
journalctl --since "2026-08-07 08:00:00" --until "2026-08-07 09:00:00"
journalctl --since "30 minutes ago"
```

Afficher sans pagination :

```
journalctl -u nginx --no-pager -n 50
```

Messages du noyau

```
dmesg -T  
sudo dmesg -T | tail -n 100  
sudo dmesg -T | grep -Ei 'error|fail|warn'  
journalctl -k -b
```

Les dates produites par `dmesg -T` peuvent être approximatives après une veille ou une modification de l'horloge ; `journalctl -k` fournit souvent un contexte temporel plus cohérent.

Journaux classiques

Selon la distribution et le service :

```
ls -lah /var/log  
sudo less /var/log/auth.log  
sudo less /var/log/syslog  
sudo less /var/log/messages
```

Espace occupé

```
journalctl --disk-usage  
sudo journalctl --vacuum-time=14d
```

La dernière commande supprime les journaux archivés plus anciens que 14 jours. Elle doit respecter la politique de conservation et les besoins d'audit.

Collecte de diagnostic

```
journalctl -u mon-service --since "1 hour ago" --no-pager > journal-service.txt  
systemctl status mon-service --no-pager >> journal-service.txt
```

Avant de partager un journal, rechercher et masquer mots de passe, jetons, adresses privées sensibles, noms d'utilisateurs et autres secrets.

6 - Réseau, Web et accès distant

Contrôler la configuration réseau, tester DNS et ports, télécharger et administrer à distance.

6.1 - Adresses IP et tests de connectivité

Le diagnostic réseau doit progresser du local vers le distant : interface, adresse, route, passerelle, DNS, port puis application.

Afficher les interfaces et les adresses

```
ip -brief link
ip -brief address
ip address show
ip link show enp1s0
```

Activer ou désactiver une interface temporairement :

```
sudo ip link set enp1s0 up
sudo ip link set enp1s0 down
```

Sur une machine distante, désactiver l'interface utilisée par SSH coupe immédiatement la connexion.

Afficher les routes

```
ip route
ip -6 route
ip route get 1.1.1.1
```

`ip route get` indique la route, l'interface et l'adresse source choisies par le noyau.

Tester la pile locale et une destination

```
ping -c 4 127.0.0.1
ping -c 4 192.168.1.1
ping -c 4 1.1.1.1
ping -c 4 example.com
```

Interprétation progressive :

1. `127.0.0.1` teste la pile IP locale.
2. La passerelle teste le lien au réseau local.
3. Une IP Internet teste le routage sans dépendre du DNS.
4. Un nom de domaine teste aussi la résolution DNS.

Certains équipements bloquent ICMP. Un échec de `ping` ne prouve donc pas à lui seul que le service est indisponible.

Voir le chemin suivi

```
tracpath example.com
traceroute example.com
```

`tracpath` est souvent utilisable sans privilèges. Les étoiles peuvent correspondre à un routeur qui ne répond pas aux sondes, sans que le trafic final soit nécessairement bloqué.

NetworkManager

Si la machine utilise NetworkManager :

```
nmcli device status
nmcli connection show
nmcli device show enp1s0
```

Statistiques et erreurs d'interface

```
ip -s link show enp1s0  
sudo ethtool enp1s0
```

Rechercher pertes, erreurs, négociation de vitesse et état du lien. `ethtool` peut nécessiter une installation séparée.

Les commandes `ip` modifient souvent l'état en mémoire seulement. Pour une configuration persistante, utiliser l'outil de la distribution : NetworkManager, systemd-networkd, Netplan ou les fichiers réseau prévus.

6.2 - DNS, routes et ports réseau

Vérifier la résolution DNS

```
getent hosts example.com
resolvectl query example.com
dig example.com
dig +short example.com
dig @1.1.1.1 example.com
```

`getent` suit la configuration complète du système, y compris `/etc/hosts`, DNS et éventuellement un annuaire. `dig` interroge directement le DNS et fournit davantage de détails.

Afficher la configuration DNS gérée par systemd-resolved :

```
resolvectl status
```

Examiner aussi :

```
cat /etc/resolv.conf
grep '^hosts:' /etc/nsswitch.conf
```

Ne pas modifier `/etc/resolv.conf` avant de vérifier s'il est généré automatiquement.

Afficher la table de routage et les voisins

```
ip route
ip neigh
```

`ip neigh` remplace les usages courants de l'ancienne commande `arp`.

Voir les ports en écoute

```
ss -tuln
sudo ss -tulpn
ss -tan state established
```

Options : `t` TCP, `u` UDP, `l` écoute, `n` valeurs numériques, `p` processus.

Exemples ciblés :

```
sudo ss -ltnp 'sport = :22'
sudo lsof -iTCP:8080 -sTCP:LISTEN
```

Tester un port distant

```
nc -vz serveur.example 22
nc -vz 192.168.1.10 443
```

Pour HTTPS, tester aussi le protocole réel :

```
curl -I --connect-timeout 5 https://serveur.example
openssl s_client -connect serveur.example:443 -servername serveur.example </dev/null
```

Un port ouvert ne garantit pas que l'application fonctionne correctement. À l'inverse, un délai peut venir du pare-feu, du routage ou d'un filtrage intermédiaire.

Ordre de diagnostic conseillé

Interface UP → adresse IP → route par défaut → passerelle → DNS → port → application

Commandes correspondantes :

```
ip -brief address
ip route
```

```
ping -c 2 adresse-passerelle  
getent hosts serveur.example  
nc -vz serveur.example 443  
curl -I https://serveur.example
```

Cette progression permet de séparer un problème réseau d'un problème applicatif au lieu de modifier plusieurs couches en même temps.

6.3 - Requêtes et téléchargements avec curl et wget

`curl` envoie des requêtes et affiche ou enregistre la réponse. `wget` est particulièrement pratique pour télécharger des fichiers de manière non interactive.

Utiliser curl

Afficher une page :

```
curl https://example.com
```

Afficher uniquement les en-têtes :

```
curl -I https://example.com
```

Suivre les redirections et enregistrer :

```
curl -L -o fichier.zip https://example.com/fichier.zip  
curl -LO https://example.com/fichier.zip
```

Échouer si le serveur renvoie une erreur HTTP :

```
curl --fail --show-error --location -o fichier.zip URL
```

Afficher les détails de connexion :

```
curl -v https://example.com  
curl -w '\nHTTP=%{http_code} Temps=%{time_total}s\n' -o /dev/null -sS https://example.com
```

Tester une API JSON :

```
curl --fail --silent --show-error \  
-H 'Accept: application/json' \  
https://api.example.com/status
```

Ne placez pas un jeton secret directement dans une capture d'écran, l'historique du shell ou une documentation partagée.

Utiliser wget

```
wget https://example.com/fichier.iso  
wget -O image.iso https://example.com/fichier.iso  
wget -c https://example.com/gros-fichier.iso
```

`-c` tente de reprendre un téléchargement interrompu si le serveur le permet.

Contrôler un téléchargement

```
file fichier.zip  
sha256sum fichier.zip
```

Comparer la somme SHA-256 avec une valeur obtenue depuis une source officielle indépendante :

```
echo 'SOMME_ATTENDUE fichier.zip' | sha256sum -c -
```

Sécurité TLS

Éviter les options `curl -k` ou `wget --no-check-certificate`, qui désactivent la validation du certificat. Sur un réseau interne, installer l'autorité de certification correcte constitue la solution durable.

Ne pas exécuter directement un téléchargement

Éviter :

```
curl URL | sudo sh
```

Procédure plus sûre : télécharger, vérifier la source et la somme, lire le script, puis l'exécuter avec les droits strictement nécessaires.

6.4 - SSH, SCP et rsync

SSH permet d'administrer une machine distante de manière chiffrée. SCP copie des fichiers via SSH et rsync synchronise efficacement des arborescences.

Se connecter en SSH

```
ssh utilisateur@serveur.example  
ssh -p 2222 utilisateur@serveur.example  
ssh utilisateur@192.168.1.20
```

Exécuter une commande distante :

```
ssh utilisateur@serveur.example 'hostname && uptime'
```

Créer une clé moderne

```
ssh-keygen -t ed25519 -a 64 -C "alex-poste"  
ssh-copy-id utilisateur@serveur.example
```

Tester ensuite la connexion avant de désactiver une authentification par mot de passe.

Permissions recommandées :

```
chmod 700 ~/.ssh  
chmod 600 ~/.ssh/id_ed25519  
chmod 644 ~/.ssh/id_ed25519.pub  
chmod 600 ~/.ssh/authorized_keys
```

Ne jamais partager la clé privée `id_ed25519`.

Copier avec scp

```
scp rapport.pdf utilisateur@serveur:/srv/documents/  
scp utilisateur@serveur:/var/log/app.log .  
scp -r dossier/ utilisateur@serveur:/srv/sauvegarde/
```

Synchroniser avec rsync

Simulation locale :

```
rsync -avhn source/ destination/
```

Exécution :

```
rsync -avh --progress source/ destination/  
rsync -avh --progress source/ utilisateur@serveur:/srv/destination/
```

La barre finale change le sens :

```
rsync -avh source/ destination/  # contenu de source  
rsync -avh source destination/  # dossier source lui-même
```

Synchronisation miroir avec suppression :

```
rsync -avhn --delete source/ destination/  # toujours simuler  
rsync -avh --delete source/ destination/
```

`--delete` supprime dans la destination ce qui n'existe plus dans la source. Vérifier les chemins et la simulation avant l'exécution.

Diagnostic SSH

```
ssh -v utilisateur@serveur.example  
ssh -G serveur.example | less
```

Vérifier côté serveur :

```
sudo systemctl status ssh  
sudo ss -ltnp 'sport = :22'  
sudo journalctl -u ssh -n 100
```

Sur Fedora ou Arch, le service peut s'appeler `sshd`. Une empreinte d'hôte modifiée doit être vérifiée auprès de l'administrateur avant de supprimer l'ancienne entrée de `known_hosts`.

7 - Disques, système et dépannage

Contrôler le stockage et le matériel, monter un disque, archiver et appliquer une méthode de diagnostic.

7.1 - Espace disque avec lsblk, df et du

Voir les disques et partitions

```
lsblk
lsblk -f
lsblk -o NAME,SIZE,FSTYPE,FSVER,LABEL,UUID,MOUNTPOINTS
sudo blkid
```

`lsblk` affiche les périphériques bloc et leur hiérarchie. `lsblk -f` ajoute le système de fichiers, l'étiquette, l'UUID et le point de montage.

Espace des systèmes de fichiers

```
df -h
df -hT
df -i
```

- `df -hT` affiche la taille, l'espace utilisé et le type de système de fichiers.
- `df -i` contrôle les inodes. Un système peut refuser de nouveaux fichiers même s'il reste des gigaoctets, lorsque tous les inodes sont utilisés.

Limiter la sortie aux systèmes locaux courants :

```
df -h -x tmpfs -x devtmpfs
```

Taille des dossiers

```
du -sh dossier
du -h --max-depth=1 /var | sort -h
sudo du -xhd1 /var | sort -h
```

`-x` évite de traverser d'autres systèmes de fichiers montés. Sur un serveur, cette option limite les surprises.

Trouver de gros fichiers dans un emplacement précis :

```
sudo find /var -xdev -type f -size +500M -printf '%s %p\n' | sort -n
```

Afficher les plus gros éléments de façon lisible avec GNU `du` :

```
sudo du -ahx /var | sort -h | tail -n 30
```

Cette commande peut être longue et solliciter le stockage.

Fichier supprimé mais encore ouvert

Si `df` indique beaucoup plus d'espace utilisé que `du`, un processus peut conserver un fichier supprimé ouvert :

```
sudo lsof +L1
```

Redémarrer proprement le service concerné libère généralement l'espace. Ne videz pas arbitrairement un descripteur de fichier dans `/proc`.

Utilisation interactive

Si `ncdu` est installé :

```
sudo ncdu -x /var
```

Réflexes avant de nettoyer

1. Identifier le système plein avec `df -hT` et `df -i`.
2. Localiser la consommation avec `du -x`.
3. Distinguer données, cache, journaux et fichiers ouverts supprimés.
4. Utiliser la procédure de nettoyage propre à l'application.
5. Vérifier le résultat et la santé du service.

Ne supprimez pas manuellement des fichiers inconnus sous `/var/lib`, qui contient souvent des bases de données et l'état des applications.

7.2 - Monter, démonter et archiver

Voir les montages actuels

```
findmnt  
findmnt /mnt/donnees  
mount | column -t
```

Monter temporairement un système de fichiers

Créer un point de montage et identifier d'abord le bon périphérique :

```
lsblk -f  
sudo mkdir -p /mnt/donnees  
sudo mount /dev/sdb1 /mnt/donnees  
findmnt /mnt/donnees
```

Démonter lorsque plus aucun processus ne l'utilise :

```
sudo umount /mnt/donnees
```

Si la cible est occupée :

```
sudo fuser -vm /mnt/donnees  
sudo lsof +D /mnt/donnees
```

Fermer proprement les fichiers et sessions concernés. Le démontage forcé ou paresseux peut masquer un problème et risquer une perte de données.

Montage persistant avec fstab

Obtenir l'UUID :

```
lsblk -f  
sudo blkid /dev/sdb1
```

Sauvegarder puis éditer :

```
sudo cp -a /etc/fstab /etc/fstab.bak  
sudoedit /etc/fstab
```

Exemple indicatif :

```
UUID=xxxx-xxxx /mnt/donnees ext4 defaults,nofail 0 2
```

Tester sans redémarrer :

```
sudo findmnt --verify --verbose  
sudo mount -a
```

Une erreur dans `/etc/fstab` peut perturber le démarrage. Utiliser les options adaptées au système de fichiers et au besoin réel.

Créer une archive tar

```
tar -cvf sauvegarde.tar dossier/  
tar -czvf sauvegarde.tar.gz dossier/  
tar -cJvf sauvegarde.tar.xz dossier/
```

Lister et extraire :

```
tar -tvf sauvegarde.tar.gz  
mkdir extraction  
tar -xzvf sauvegarde.tar.gz -C extraction/
```

Une archive provenant d'une source non fiable peut contenir des chemins dangereux. Lister d'abord son contenu et extraire dans un dossier vide sans privilèges.

ZIP

```
zip -r archive.zip dossier/  
unzip -l archive.zip  
unzip archive.zip -d extraction/
```

Contrôler l'intégrité

```
sha256sum sauvegarde.tar.gz > sauvegarde.tar.gz.sha256  
sha256sum -c sauvegarde.tar.gz.sha256
```

Une somme de contrôle détecte une modification accidentelle, mais ne prouve l'authenticité que si la somme attendue vient d'une source fiable.

7.3 - Informations système et matériel

Système d'exploitation et noyau

```
cat /etc/os-release  
uname -a  
uname -r  
hostnamectl  
uptime
```

`/etc/os-release` décrit la distribution ; `uname -r` indique le noyau actuellement démarré, qui peut différer du dernier noyau installé.

Processeur et mémoire

```
lscpu  
free -h  
grep -E 'MemTotal|MemAvailable|SwapTotal|SwapFree' /proc/meminfo
```

Matériel PCI, USB et disques

```
lspci  
lspci -nnk  
lsusb  
lsblk -o NAME,MODEL,SERIAL,SIZE,TYPE,FSTYPE,MOUNTPOINTS
```

`lspci -nnk` affiche les identifiants matériels, le pilote utilisé et les modules disponibles, très utile pour un GPU ou une carte réseau.

Modules du noyau

```
lsmod  
modinfo nom_du_module  
sudo modprobe nom_du_module
```

Retirer un module peut couper un périphérique actif. Ne pas utiliser `modprobe -r` à distance sur le pilote de la carte réseau employée.

Périphériques et événements récents

```
sudo dmesg -T | tail -n 100  
journalctl -k -b  
udevadm info --query=all --name=/dev/sdb
```

Temps et fuseau horaire

```
date  
timedatectl
```

Sur un serveur, une horloge incorrecte peut casser TLS, Kerberos, l'authentification et la corrélation des journaux.

Charge et sessions

```
uptime  
w  
last -n 20
```

Les trois valeurs de charge d'`uptime` couvrent environ 1, 5 et 15 minutes. Elles doivent être interprétées avec le nombre de processeurs logiques et l'attente d'entrées-sorties.

Construire un petit rapport non sensible

```
{  
  date  
  hostnamectl  
  uptime  
  free -h  
  df -hT  
  ip -brief address  
} > diagnostic-systeme.txt
```

Examiner le fichier avant de le transmettre : il peut révéler nom de machine, adresses IP, utilisateurs, numéros de série ou structure du stockage.

Commandes optionnelles utiles

```
inxi -Fxxxz      # synthèse détaillée si inxi est installé  
sudo dmidecode   # informations DMI/BIOS, nécessite root  
sudo smartctl -a /dev/sda  # santé SMART si smartmontools est installé
```

Ne lancez pas de test SMART long ou destructif sur un disque de production sans connaître son impact et sans sauvegarde valide.

7.4 - Méthode de dépannage et mémo des commandes

Méthode de dépannage

Une bonne résolution d'incident sépare les faits, les hypothèses et les modifications.

1. **Décrire le symptôme** : message exact, heure, utilisateur et action.
2. **Définir le périmètre** : une application, une machine, un VLAN ou tous les utilisateurs.
3. **Reproduire prudemment** : noter les étapes et éviter d'aggraver l'incident.
4. **Observer avant de modifier** : état du service, journaux, ressources, réseau.
5. **Formuler une hypothèse testable** : une cause à la fois.
6. **Changer une seule chose** : noter la commande et prévoir le retour arrière.
7. **Valider de bout en bout** : service, client, persistance après redémarrage si nécessaire.
8. **Documenter** : cause, correction, vérification et prévention.

Collecte initiale

```
date
hostnamectl
uptime
systemctl --failed
free -h
df -hT
df -i
ip -brief address
ip route
```

Pour un service :

```
systemctl status mon-service --no-pager  
journalctl -u mon-service --since "30 minutes ago" --no-pager  
sudo ss -tulpn
```

Mémo des commandes essentielles

Besoin	Commande
savoir où je suis	<code>pwd</code>
lister en détail	<code>ls -lah</code>
changer de dossier	<code>cd chemin</code>
copier un dossier	<code>cp -a source destination</code>
déplacer/renommer	<code>mv -i source destination</code>
lire un gros fichier	<code>less fichier</code>
suivre un journal	<code>tail -F fichier.log</code>
rechercher du texte	<code>grep -RIn 'motif' chemin</code>
trouver un fichier	<code>find chemin -type f -name 'motif'</code>
espace disque	<code>df -hT</code> et <code>du -xhd1 chemin</code>
mémoire	<code>free -h</code>
processus	<code>ps aux</code> , <code>top</code> , <code>pgrep -a nom</code>
service	<code>systemctl status nom</code>
journaux	<code>journalctl -u nom -n 100</code>
réseau	<code>ip -brief address</code> , <code>ip route</code>
DNS	<code>getent hosts nom</code>
ports	<code>sudo ss -tulpn</code>
accès distant	<code>ssh utilisateur@serveur</code>
aide	<code>commande --help</code> , <code>man commande</code>

Commandes qui exigent une attention particulière

- `rm -rf` : suppression récursive sans confirmation.
- `chmod -R` et `chown -R` : changement récursif de droits ou propriétaire.
- `dd` : copie brute capable d'écraser un disque entier.
- `mkfs` : création d'un système de fichiers, donc destruction du contenu précédent.
- `fdisk`, `parted`, `zpool destroy` : modification ou destruction du stockage.
- `rsync --delete` : suppression dans la destination.
- modification de `/etc/fstab`, du réseau ou de SSH : risque de perte d'accès après redémarrage.

Avant une commande sensible :

```
whoami  
pwd  
ls -ld -- /chemin/cible
```

Utiliser ensuite un mode simulation lorsqu'il existe, par exemple `rsync -n`, puis relire la commande complète. Une sauvegarde n'est considérée valide qu'après un test de restauration.