Guide Très Détaillé d'Arch Linux : Des Bases aux Commandes Avancées

Bienvenue dans ce guide très détaillé d'Arch Linux ! Conçu pour vous accompagner de la découverte à une maîtrise plus approfondie, ce document explore l'histoire, la philosophie et surtout les commandes essentielles et avancées de cette distribution unique.

Un Peu d'Histoire

Arch Linux, né en mars 2002 sous l'impulsion de Judd Vinet, visait la simplicité et le contrôle utilisateur, inspiré par des distributions comme CRUX. Sa philosophie **KISS** ("Keep It Simple, Stupid") se traduit par une base technique épurée. Le modèle **"rolling release"** assure des mises à jour continues, offrant les logiciels les plus récents sans attendre des versions majeures. Arch est une toile vierge : vous construisez votre système.

La Philosophie d'Arch : "The Arch Way"

Les principes fondamentaux d'Arch :

1. **Simplicité :** Privilégie une conception technique directe, sans complexité cachée.
2. **Modernité :** Fournit rapidement les dernières versions stables des logiciels.
3. **Pragmatisme :** Décisions basées sur la technique, pas l'idéologie.
4. **Centré sur l'Utilisateur :** L'utilisateur contrôle tout, de l'installation aux logiciels choisis.
5. **Polyvalence :** Adaptable à n'importe quel besoin.

Les Commandes Essentielles (Expliquées en Détail)

La ligne de commande est le cœur de l'interaction avec Arch.

1. Naviguer et Explorer le Système de Fichiers

- `pwd` **(Print Working Directory)**
    - **Utilité :** Affiche le chemin complet du répertoire dans lequel vous vous trouvez actuellement. Indispensable pour savoir où vous êtes dans l'arborescence, surtout lorsque le prompt n'affiche pas le chemin complet.
    - **Exemple :** Si vous êtes dans le dossier `Documents` de votre répertoire personnel, `pwd` affichera quelque chose comme `/home/votre_utilisateur/Documents`.
- `ls` **(List)**
    - **Utilité :** Liste les fichiers et les sous-dossiers présents dans le répertoire courant (ou un autre répertoire si spécifié, ex: `ls /etc`). C'est l'une des commandes les plus utilisées.
    - **Options courantes :**
        - `ls -l` : Affiche une "liste longue". Pour chaque fichier/dossier, elle montre : type et permissions (`drwxr-xr-x`), nombre de liens, propriétaire, groupe, taille, date/heure de dernière modification, et nom. Très utile pour obtenir des détails.

        - `ls -a` : Affiche "tous" les fichiers, y compris ceux dont le nom commence par un point (`.`), qui sont cachés par défaut (souvent des fichiers de configuration). Essentiel pour voir la configuration dans votre dossier personnel.
        - `ls -h` : Utilisée avec `-l` (`ls -lh`), cette option rend les tailles de fichiers "lisibles par un humain" (Human-readable), en affichant `K` (Kio), `M` (Mio), `G` (Gio) au lieu d'un grand nombre d'octets. Facilite grandement la lecture des tailles.
        - `ls -lah` : Combine les trois options pour un aperçu complet, détaillé et lisible de tout le contenu d'un répertoire.
    - **Exemple :** `ls -lh /home/votre_utilisateur` listera le contenu de votre dossier personnel avec détails et tailles lisibles.

- **`cd <répertoire>` (Change Directory)**
    - **Utilité :** Permet de se déplacer dans l'arborescence des fichiers. C'est la commande fondamentale pour naviguer.
    - **Arguments courants :**
        - `cd NomDuDossier` : Entre dans le sous-dossier `NomDuDossier`.
        - `cd ..` : Remonte au répertoire parent (le dossier juste au-dessus).
        - `cd ~` ou simplement `cd` : Retourne instantanément à votre répertoire personnel (`/home/votre_utilisateur`), peu importe où vous étiez.
        - `cd -` : Retourne au répertoire où vous étiez juste avant le dernier `cd`. Très pratique pour basculer entre deux dossiers.
        - `cd /` : Va directement au répertoire racine, le sommet de l'arborescence.
    - **Exemple :** `cd /var/log` vous déplace dans le répertoire contenant les fichiers journaux du système. `cd ..` vous ramènerait ensuite à `/var`.

- **`tree`**
    - **Utilité :** Affiche le contenu d'un répertoire et de ses sous-répertoires sous forme d'arbre graphique. Donne une vue d'ensemble de la structure.
    - **Installation :** N'est pas toujours installé par défaut. `sudo pacman -S tree` si nécessaire.
    - **Exemple :** `tree Documents` affichera l'arborescence de votre dossier Documents.

2. Gérer les Fichiers et Dossiers

- **`mkdir <nom_dossier>` (Make Directory)**
    - **Utilité :** Crée un nouveau répertoire (dossier) vide.
    - **Option utile :**
        - `mkdir -p <chemin/vers/dossier>` : L'option `-p` (parents) permet de créer toute l'arborescence de répertoires nécessaire si elle n'existe pas. Par exemple, `mkdir -p Projets/Web/Assets` créera `Projets`, puis `Web` à l'intérieur, puis `Assets` à l'intérieur de `Web`, si l'un d'eux manque.
    - **Exemple :** `mkdir Sauvegardes` crée un dossier nommé Sauvegardes.

- **`touch <nom_fichier>`**
    - **Utilité :** Principalement utilisé pour créer un nouveau fichier vide rapidement. Si le fichier existe déjà, `touch` met simplement à jour sa date et heure de dernière modification/accès à l'heure

actuelle, sans en changer le contenu.
    - **Exemple :** `touch mon_nouveau_script.sh` crée un fichier vide prêt à être édité.

- **`cp <source> <destination>` (Copy)**
    - **Utilité :** Copie des fichiers ou des répertoires.
    - **Syntaxe :** `cp <ce_qui_doit_être_copié> <où_le_copier>`
    - **Option essentielle :**
        - `cp -r <dossier_source> <dossier_destination>` : L'option `-r` (récursif) est **obligatoire** pour copier un répertoire et tout ce qu'il contient (fichiers et sous-dossiers). Sans `-r`, `cp` refusera de copier un répertoire.
    - **Exemples :**
        - `cp mon_fichier.txt mon_fichier_copie.txt` : Copie le fichier dans le même répertoire avec un nouveau nom.
        - `cp rapport.pdf Documents/` : Copie `rapport.pdf` dans le sous-dossier `Documents`.
        - `cp -r ~/Images /media/usb_backup/` : Copie récursivement tout le dossier `Images` vers une clé USB montée.

- **`mv <source> <destination>` (Move)**
    - **Utilité :** A deux fonctions principales : déplacer des fichiers/dossiers d'un endroit à un autre, ou renommer un fichier/dossier. Le shell détermine l'action en fonction de la `destination`.
    - **Pour renommer :** Si la `destination` est un nom qui n'existe pas dans le répertoire courant, `mv` renomme la `source`.
        - **Exemple :** `mv fichier_temporaire.txt rapport_final.txt` renomme le fichier.
    - **Pour déplacer :** Si la `destination` est un répertoire existant, `mv` déplace la `source` à l'intérieur de ce répertoire.
        - **Exemple :** `mv image.jpg ~/Images/` déplace `image.jpg` dans le dossier `Images`.
    - **Déplacer et renommer en même temps :** `mv rapport.txt Documents/rapport_avril.txt` déplace `rapport.txt` dans `Documents` et le renomme `rapport_avril.txt`.

- **`rm <fichier>` (Remove)**
    - **Utilité :** Supprime des fichiers. **Attention : Par défaut, `rm` est définitif ! Les fichiers ne vont pas dans une corbeille. Une fois supprimés, ils sont très difficiles (voire impossibles) à récupérer.**
    - **Option de sécurité :**
        - `rm -i <fichier>` : L'option `-i` (interactive) demande une confirmation (`y` ou `n`) avant de supprimer chaque fichier. Recommandé si vous n'êtes pas sûr.
    - **Exemple :** `rm fichier_inutile.log` supprime le fichier.

- **`rmdir <dossier_vide>` (Remove Directory)**
    - **Utilité :** Supprime un répertoire, mais **uniquement** s'il est complètement vide. C'est une sécurité pour éviter de supprimer accidentellement un dossier contenant des fichiers.
    - **Exemple :** `rmdir vieux_projet` ne fonctionnera que si `vieux_projet` ne contient absolument rien.

- `rm -r <dossier>` **(Remove Recursive)**
  - **Utilité :** Supprime un répertoire et **absolument tout** ce qu'il contient (fichiers, sous-dossiers, et leur contenu). C'est la commande pour supprimer des dossiers non vides.
  - **Option `-f` (Force) :**
    - `rm -rf <dossier>` : Combine la récursivité (`-r`) avec la force (`-f`). `-f` supprime toute demande de confirmation, même pour les fichiers protégés en écriture appartenant à l'utilisateur. **C'EST UNE COMMANDE EXTRÊMEMENT DANGEREUSE.** Une faute de frappe (ex: `rm -rf /` au lieu de `rm -rf ./` dans un mauvais dossier) peut effacer tout votre système. **À utiliser avec la plus grande prudence et seulement si vous êtes absolument certain de ce que vous faites.**
  - **Exemple prudent :** `rm -rI vieux_projet/` (L'option `-I` demande une confirmation avant de supprimer plus de trois fichiers ou lors d'une suppression récursive. Moins intrusive que `-i`).

3. Visualiser et Éditer des Fichiers

- `cat <fichier>` **(Concatenate)**
  - **Utilité :** Affiche le contenu *intégral* d'un ou plusieurs fichiers directement dans le terminal. Simple et direct pour les petits fichiers. Pour les fichiers longs, le contenu défilera très vite.
  - **Exemple :** `cat /etc/hosts` affiche le fichier hosts. `cat fichier1.txt fichier2.txt > fichier_combine.txt` concatène les deux fichiers dans un troisième.

- `less <fichier>`
  - **Utilité :** Affiche le contenu d'un fichier de manière interactive, page par page. C'est le visualiseur standard pour les fichiers longs (logs, fichiers de configuration, man pages).
  - **Navigation :** Utilisez les flèches haut/bas, PageUp/PageDown pour vous déplacer. `/` suivi d'un terme permet de rechercher. `q` pour quitter.
  - **Exemple :** `less /var/log/pacman.log` permet de consulter l'historique de pacman confortablement.

- `head <fichier>`
  - **Utilité :** Affiche le début (la "tête") d'un fichier. Par défaut, les 10 premières lignes.
  - **Option utile :**
    - `head -n <nombre> <fichier>` : Affiche le `<nombre>` spécifié de lignes.
  - **Exemple :** `head -n 20 mon_script.sh` affiche les 20 premières lignes du script.

- `tail <fichier>`
  - **Utilité :** Affiche la fin (la "queue") d'un fichier. Par défaut, les 10 dernières lignes. Très utile pour voir les dernières entrées d'un fichier journal.
  - **Options utiles :**
    - `tail -n <nombre> <fichier>` : Affiche le `<nombre>` spécifié de dernières lignes.
    - `tail -f <fichier>` : L'option `-f` (follow) est extrêmement utile. Elle affiche les dernières lignes, puis reste active et affiche en temps réel toute nouvelle ligne ajoutée au fichier. Idéal pour surveiller un log pendant qu'un processus écrit dedans. Appuyez sur Ctrl+C pour arrêter le suivi.

    - **Exemple :** `tail -f /var/log/syslog` (si ce fichier existe) surveille les messages système en temps réel.

- **`nano <fichier>`**
    - **Utilité :** Un éditeur de texte en ligne de commande très simple et convivial pour les débutants. Les commandes principales (Sauver, Quitter, Chercher...) sont affichées en bas avec les raccourcis clavier (ex: `^X` signifie Ctrl+X).
    - **Exemple :** `nano ~/.bashrc` ouvre le fichier de configuration de Bash pour modification.

- **`vim <fichier>` ou `vi <fichier>`**
    - **Utilité :** Un éditeur de texte en ligne de commande extrêmement puissant et omniprésent sur les systèmes Unix/Linux. Il est "modal", ce qui signifie qu'il a différents modes (Normal pour les commandes, Insertion pour taper du texte, Visuel pour sélectionner). Sa courbe d'apprentissage est raide, mais il offre une efficacité redoutable une fois maîtrisé. `vi` est la version historique, `vim` (Vi IMproved) est la version moderne avec plus de fonctionnalités (coloration syntaxique, etc.).
    - **Commandes de base (Mode Normal) :** `i` pour passer en mode Insertion, `Esc` pour revenir en mode Normal, `:w` pour sauvegarder (Write), `:q` pour quitter (Quit), `:wq` pour sauvegarder et quitter, `:q!` pour quitter sans sauvegarder.
    - **Exemple :** `vim /etc/fstab` ouvre le fichier de configuration des montages système.

4. Gérer les Logiciels (`pacman`)

`pacman` est le gestionnaire de paquets puissant et rapide d'Arch Linux. La plupart des commandes `pacman` qui modifient le système (installer, supprimer, mettre à jour) nécessitent les privilèges root, obtenus via `sudo`.

- **`sudo pacman -S <paquet>` (Sync)**
    - **Utilité :** Installe un ou plusieurs paquets depuis les dépôts configurés. `pacman` résout automatiquement les dépendances nécessaires et les installe aussi.
    - **Exemple :** `sudo pacman -S firefox` installe le navigateur Firefox. `sudo pacman -S gimp inkscape` installe GIMP et Inkscape en une seule commande.

- **`sudo pacman -Syu` (Sync, Refresh, Upgrade)**
    - **Utilité :** C'est la commande **essentielle** pour maintenir votre système Arch à jour. Elle effectue trois actions :
        1. `-y` (Refresh) : Télécharge une copie fraîche des listes de paquets depuis les serveurs miroirs.
        2. `-u` (Upgrade) : Compare les versions des paquets installés avec celles des listes téléchargées et met à jour tous les paquets pour lesquels une nouvelle version est disponible.
        3. `-S` (Sync) : Contexte général de synchronisation avec les dépôts.
    - **Fréquence :** Il est recommandé d'exécuter `sudo pacman -Syu` régulièrement (de quotidiennement à hebdomadairement) pour bénéficier des dernières mises à jour de sécurité et de fonctionnalités. Ne pas mettre à jour pendant très longtemps peut rendre les mises à jour ultérieures plus complexes. **Toujours lire les nouvelles sur archlinux.org avant une mise à jour importante.**

- **`sudo pacman -Syyu` (Sync, Force Refresh, Upgrade)**
  - **Utilité :** Identique à `-Syu`, mais le double `-yy` force le re-téléchargement des listes de paquets même si `pacman` pense qu'elles sont à jour.
  - **Quand l'utiliser ?** Principalement si vous avez changé de miroirs de dépôts ou si `-Syu` échoue avec des erreurs indiquant des listes de paquets potentiellement corrompues ou désynchronisées. Ne pas utiliser systématiquement car cela augmente inutilement la charge sur les serveurs miroirs.
- **`sudo pacman -R <paquet>` (Remove)**
  - **Utilité :** Désinstalle un paquet spécifié.
  - **Comportement :** Ne supprime **pas** les dépendances qui ont été installées avec ce paquet, même si elles ne sont plus nécessaires à aucun autre paquet. Elles deviennent alors des paquets "orphelins".
- **`sudo pacman -Rs <paquet>` (Remove, Recursive)**
  - **Utilité :** Désinstalle un paquet et **supprime également ses dépendances** à condition qu'elles ne soient requises par aucun autre paquet explicitement installé sur le système.
  - **Recommandation :** C'est généralement l'option **préférée** pour désinstaller proprement un logiciel et éviter d'accumuler des paquets orphelins inutiles.
  - **Exemple :** `sudo pacman -Rs gimp` désinstalle GIMP et toutes les bibliothèques installées uniquement pour GIMP.
- **`sudo pacman -Rns <paquet>` (Remove, NoSave, Recursive)**
  - **Utilité :** Combine `-Rs` (suppression du paquet et de ses dépendances orphelines) avec l'option `-n` (NoSave).
  - **Option `-n` :** Indique à `pacman` de supprimer également les fichiers de configuration importants associés au paquet (ceux listés dans la directive `backup=(...)` du fichier PKGBUILD du paquet, souvent situés dans `/etc/`). Par défaut (`-R` ou `-Rs`), `pacman` conserve ces fichiers de configuration (en les renommant parfois avec `.pacsave`).
  - **Prudence :** Utilisez `-n` si vous êtes sûr de ne plus jamais vouloir réutiliser la configuration de ce paquet. Sinon, préférez `-Rs`.
- **`pacman -Q` (Query local database)**
  - **Utilité :** Interroge la base de données locale des paquets installés sur votre système. Utilisé sans argument, `pacman -Q` liste tous les paquets installés et leur version.
  - **Option utile :**
    - `pacman -Qe` : Liste uniquement les paquets que vous avez "explicitement" installés (avec `pacman -S` ou équivalent). Exclut les paquets installés automatiquement comme dépendances. Utile pour voir les logiciels principaux que vous avez choisi d'installer.
- **`pacman -Q <paquet>`**
  - **Utilité :** Vérifie si un paquet spécifique est installé localement et, si oui, affiche son nom et sa version.
  - **Exemple :** `pacman -Q bash`

- `pacman -Qs <terme>` **(Query, Search local)**
  - **Utilité :** Recherche un `<terme>` (expression régulière) dans les noms et descriptions des paquets installés localement.
  - **Exemple :** `pacman -Qs networkmanager` cherche les paquets installés liés à NetworkManager.
- `pacman -Ss <terme>` **(Sync, Search remote)**
  - **Utilité :** Recherche un `<terme>` (expression régulière) dans les noms et descriptions de tous les paquets disponibles dans les dépôts distants (synchronisés avec `-Sy`). C'est la commande pour trouver de nouveaux logiciels à installer.
  - **Exemple :** `pacman -Ss text editor` cherche des éditeurs de texte dans les dépôts.
- `pacman -Si <paquet>` **(Sync, Info remote)**
  - **Utilité :** Affiche des informations détaillées (description, version, dépendances, taille, etc.) sur un paquet disponible dans les dépôts distants, avant de l'installer.
  - **Exemple :** `pacman -Si neofetch`
- `pacman -Qi <paquet>` **(Query, Info local)**
  - **Utilité :** Affiche des informations détaillées sur un paquet qui est déjà installé localement (date d'installation, raison de l'installation, dépendances requises, paquets optionnels...).
  - **Exemple :** `pacman -Qi linux`
- `pacman -Ql <paquet>` **(Query, List files)**
  - **Utilité :** Liste tous les fichiers qui ont été installés sur le système par un paquet spécifique. Très utile pour savoir où se trouvent les fichiers d'un programme ou pour vérifier si un fichier appartient bien à un paquet.
  - **Exemple :** `pacman -Ql bash` liste tous les fichiers installés par le paquet `bash`.
- `pacman -Qo <chemin/fichier>` **(Query, Owns file)**
  - **Utilité :** Fait l'inverse de `-Ql`. Indique quel paquet installé sur le système possède (a installé) le fichier spécifié. Utile pour identifier l'origine d'un fichier système.
  - **Exemple :** `pacman -Qo /usr/bin/ls` indiquera que ce fichier appartient au paquet `coreutils`.
- `sudo pacman -Sc` **(Sync, Clean cache)**
  - **Utilité :** Nettoie le cache des paquets de `pacman` (situé dans `/var/cache/pacman/pkg/`). Cette commande supprime toutes les versions de paquets qui ne sont pas actuellement installées sur le système, mais conserve les versions correspondant à ce qui est installé. Utile pour libérer de l'espace disque sans risque.
  - **Contexte :** `pacman` conserve les fichiers `.pkg.tar.zst` téléchargés dans le cache, ce qui permet de réinstaller rapidement un paquet ou de revenir à une version précédente (downgrade) si nécessaire.
- `sudo pacman -Scc` **(Sync, Clean cache aggressively)**

    - **Utilité :** Nettoie le cache de manière beaucoup plus agressive. Cette commande supprime **tous** les fichiers du cache, y compris ceux correspondant aux paquets actuellement installés.
    - **Conséquences :** Libère un maximum d'espace disque, mais vous ne pourrez plus revenir à une version précédente ou réinstaller un paquet sans le re-télécharger complètement. À utiliser avec prudence.

- `pactree <paquet>`

    - **Utilité :** Affiche l'arbre des dépendances d'un paquet. Montre de quels autres paquets il dépend, et récursivement, de quoi ces dépendances dépendent. Utile pour comprendre les relations entre les paquets. (Peut nécessiter `sudo pacman -S pacman-contrib`).
    - **Exemple :** `pactree firefox`

Au-delà des Bases : Commandes et Concepts Avancés (Expliqués en Détail)

5. L'Arch User Repository (AUR)

- **Concept :** L'AUR n'est pas un dépôt de paquets binaires comme les dépôts officiels. C'est un dépôt géré par la communauté contenant des "recettes de construction" appelées `PKGBUILD`. Un `PKGBUILD` est un script shell qui contient les instructions pour télécharger le code source d'un logiciel (souvent depuis son site officiel ou un dépôt Git), le compiler si nécessaire, et l'empaqueter dans un format que `pacman` peut installer. L'AUR donne accès à des milliers de logiciels non disponibles dans les dépôts officiels.
- **Avertissement Crucial :** Les `PKGBUILD` sont créés par des utilisateurs. **Il n'y a aucune garantie officielle de qualité, de sécurité ou de non-malveillance.** Avant de construire et d'installer un paquet depuis l'AUR, il est **fortement recommandé d'inspecter le contenu du fichier `PKGBUILD`** (et des éventuels fichiers `.install` ou patchs associés) pour comprendre exactement quelles commandes seront exécutées sur votre système.
- **AUR Helpers (`yay`, `paru`, etc.) :** Ce sont des outils en ligne de commande qui automatisent le processus de recherche, téléchargement du `PKGBUILD`, construction et installation depuis l'AUR. Ils rendent l'utilisation de l'AUR beaucoup plus pratique. Ils ne sont pas officiellement supportés par Arch Linux mais sont très populaires.
    - **Installation d'un Helper (exemple `yay`) :**
        1. `sudo pacman -S --needed git base-devel` : Installe les prérequis : `git` pour cloner, `base-devel` est un groupe de paquets essentiels pour la compilation (make, gcc, etc.). `--needed` évite de réinstaller ce qui l'est déjà.
        2. `git clone https://aur.archlinux.org/yay.git` : Télécharge le `PKGBUILD` de `yay` lui-même depuis l'AUR.
        3. `cd yay` : Entre dans le dossier téléchargé.
        4. `makepkg -si` : C'est la commande standard pour construire un paquet à partir d'un `PKGBUILD`.
            - `makepkg` : Lit le `PKGBUILD`, télécharge les sources, compile, et crée le paquet `.pkg.tar.zst`. **Ne jamais exécuter `makepkg` en tant que root (`sudo`) !**
            - `-s` (syncdeps) : Demande à `makepkg` d'utiliser `pacman` pour installer automatiquement les dépendances listées dans le `PKGBUILD` avant de compiler.
            - `-i` (install) : Demande à `makepkg` d'utiliser `pacman -U` pour installer le paquet une fois qu'il a été construit avec succès.

- **Utilisation de `yay` :**
  - `yay -S <paquet>` : Recherche d'abord dans les dépôts officiels. Si non trouvé, cherche dans l'AUR. Si trouvé dans l'AUR, propose d'afficher le `PKGBUILD`, puis le construit et l'installe.
  - `yay -Syu` : Équivalent à `sudo pacman -Syu` suivi d'une vérification et mise à jour des paquets installés depuis l'AUR.
  - `yay -Ss <terme>` : Recherche dans les dépôts officiels ET dans l'AUR.
  - `yay -Rns <paquet>` : Désinstalle un paquet (qu'il vienne des dépôts ou de l'AUR) et ses dépendances orphelines.

6. Gestion des Permissions et Propriétaires

Le système de permissions Unix/Linux est fondamental pour la sécurité et le fonctionnement multi-utilisateurs. Chaque fichier et dossier a :

- Un propriétaire (utilisateur).
- Un groupe propriétaire.
- Des permissions définies pour trois catégories : le propriétaire (`u`ser), le groupe (`g`roup), et les autres (`o`thers).
- Pour chaque catégorie, trois permissions de base : lecture (`r`ead), écriture (`w`rite), exécution (`x`ecute).
- **`chmod <permissions> <fichier/dossier>` (Change Mode)**
  - **Utilité :** Modifie les permissions d'accès.
  - **Mode Octal :** Méthode numérique concise. Chaque permission a une valeur : `r=4`, `w=2`, `x=1`. On additionne les valeurs pour chaque catégorie (user, group, other).
    - `chmod 755 mon_script.sh` : `7` (4+2+1 = rwx) pour le propriétaire, `5` (4+0+1 = r-x) pour le groupe, `5` (4+0+1 = r-x) pour les autres. Permissions typiques pour un script exécutable ou un dossier accessible.
    - `chmod 644 fichier.txt` : `6` (4+2+0 = rw-) pour le propriétaire, `4` (4+0+0 = r--) pour le groupe, `4` (4+0+0 = r--) pour les autres. Permissions typiques pour un fichier de données lisible par tous mais modifiable seulement par le propriétaire.
    - `chmod 600 cle_privee.key` : `6` (rw-) pour le propriétaire, `0` (---) pour le groupe, `0` (---) pour les autres. Permissions très restrictives, typiques pour des fichiers sensibles.
  - **Mode Symbolique :** Plus verbeux mais parfois plus clair pour des modifications ciblées.
    - `u`, `g`, `o`, `a` (all) : Spécifie à qui s'applique le changement.
    - `+`, `-`, `=` : Ajoute, retire, ou définit exactement les permissions.
    - `r`, `w`, `x` : Les permissions.
    - **Exemples :** `chmod u+x script.sh` (ajoute l'exécution pour le propriétaire), `chmod g-w rapport.doc` (retire l'écriture pour le groupe), `chmod go=r notes.txt` (définit les permissions à lecture seule pour le groupe et les autres).
  - **Option `-R` (Récursif) :** `chmod -R 644 Documents/` applique les permissions à tous les fichiers et sous-dossiers dans `Documents`.
- **`chown <utilisateur>:<groupe> <fichier/dossier>` (Change Owner)**

- **Utilité :** Change le propriétaire et/ou le groupe propriétaire d'un fichier ou dossier. Nécessite généralement `sudo` car on ne peut donner un fichier qu'à un utilisateur dont on est membre du groupe (sauf pour root).
- **Syntaxe :**
  - `sudo chown nouvel_utilisateur fichier.txt` : Change uniquement le propriétaire.
  - `sudo chown :nouveau_groupe fichier.txt` : Change uniquement le groupe.
  - `sudo chown nouvel_utilisateur:nouveau_groupe fichier.txt` : Change les deux.
- **Option `-R` (Récursif) :** `sudo chown -R www-data:www-data /var/www/html` change récursivement le propriétaire et le groupe de tous les fichiers d'un répertoire web (exemple courant).

- **`ls -l`**
  - **Utilité :** Bien que déjà mentionnée, il est crucial de rappeler que `ls -l` est la commande pour *vérifier* les permissions et propriétaires actuels après les avoir modifiés avec `chmod` ou `chown`. La première colonne (`-rw-r--r--`) montre le type (`-` pour fichier, `d` pour dossier, `l` pour lien symbolique) et les permissions (rwx pour user, group, other). Les 3ème et 4ème colonnes montrent le propriétaire et le groupe.

7. Surveillance et Analyse Système

Ces commandes permettent de comprendre ce qui se passe sur votre système en temps réel ou d'analyser des problèmes.

- **`top` / `htop`**
  - **Utilité :** Fournissent un tableau de bord dynamique des processus en cours, triés par défaut par utilisation CPU. Affiche l'utilisation globale CPU/Mémoire/Swap, la charge système (load average), et des détails pour chaque processus (PID, utilisateur, %CPU, %MEM, commande).
  - **`htop` vs `top` :** `htop` (nécessite `sudo pacman -S htop`) est généralement préféré car il est plus interactif (défilement, sélection avec les flèches, envoi de signaux avec les touches de fonction comme F9 pour 'kill') et plus lisible (couleurs, barres de progression).
  - **Interaction :** Dans `top` ou `htop`, appuyez sur `h` pour l'aide, `k` pour tuer un processus (demande le PID), `q` pour quitter. `htop` offre plus d'options via les touches F1-F10.

- **`ps aux` (Process Status)**
  - **Utilité :** Donne une "photographie" instantanée de tous les processus actifs au moment où la commande est lancée. Moins dynamique que `top`/`htop`, mais utile pour capturer l'état à un instant T ou pour utiliser avec d'autres commandes (comme `grep`).
  - **Options expliquées :**
    - `a` : Affiche les processus de tous les utilisateurs (pas seulement les vôtres).
    - `u` : Affiche des informations détaillées dans un format orienté utilisateur (USER, PID, %CPU, %MEM, VSZ, RSS, TTY, STAT, START, TIME, COMMAND).
    - `x` : Inclut les processus qui ne sont pas attachés à un terminal (les démons et services système).

    - **Usage courant :** `ps aux | grep <nom_processus>` pour rechercher si un processus spécifique est en cours et obtenir son PID.

- `vmstat <intervalle> <compte>` **(Virtual Memory Statistics)**
    - **Utilité :** Fournit des rapports périodiques sur l'activité système : processus (en attente, bloqués), mémoire (swap in/out), IO bloc (lus/écrits), interruptions, et utilisation CPU (user, system, idle, wait). Utile pour détecter des goulots d'étranglement.
    - **Exemple :** `vmstat 2 5` affiche 5 rapports, un toutes les 2 secondes. La première ligne est une moyenne depuis le démarrage, les suivantes sont les mesures sur l'intervalle.

- `iostat <intervalle> <compte>` **(Input/Output Statistics)**
    - **Utilité :** Rapporte des statistiques sur l'utilisation du CPU et, surtout, sur les activités d'entrée/sortie des périphériques de stockage (disques). Affiche le nombre de transferts par seconde (tps), les Ko lus/écrits par seconde, etc., par disque ou partition. Essentiel pour diagnostiquer des problèmes de performance disque.
    - **Installation :** Fait partie du paquet `sysstat` (`sudo pacman -S sysstat`).
    - **Exemple :** `iostat -dx 2` affiche les statistiques disque étendues (`-x`) toutes les 2 secondes.

- `lsof` **(List Open Files)**
    - **Utilité :** Sous Unix/Linux, "tout est fichier", y compris les connexions réseau, les pipes, etc. `lsof` liste tous les fichiers ouverts par tous les processus actifs. C'est un outil de diagnostic extrêmement puissant.
    - **Installation :** Peut nécessiter `sudo pacman -S lsof`.
    - **Exemples d'utilisation :**
        - `sudo lsof -i` : Liste toutes les connexions réseau ouvertes.
        - `sudo lsof -i :<port>` : Montre quel(s) processus utilise(nt) un port réseau spécifique (ex: `sudo lsof -i :80` pour voir quel serveur web tourne).
        - `lsof /chemin/vers/fichier` : Montre quel processus a ouvert ce fichier spécifique (utile si vous ne pouvez pas démonter un disque car "ressource occupée").
        - `lsof -u <utilisateur>` : Liste les fichiers ouverts par un utilisateur spécifique.

- `ss` **(Socket Statistics)**
    - **Utilité :** Outil moderne pour examiner les sockets réseau (connexions TCP, UDP, sockets Unix). Il est plus rapide et fournit plus d'informations que l'ancien `netstat`.
    - **Options courantes (`ss -tulnp`) :**
        - `-t` : Affiche les sockets TCP.
        - `-u` : Affiche les sockets UDP.
        - `-l` : Affiche uniquement les sockets en état d'écoute (LISTEN).
        - `-n` : N'essaie pas de résoudre les noms de services ou d'hôtes (affiche les numéros de port et adresses IP). Plus rapide.
        - `-p` : Affiche le processus (PID et nom) qui utilise le socket (nécessite `sudo`).
    - **Exemple :** `sudo ss -tulnp | grep :22` vérifie si un processus écoute sur le port 22 (SSH) et lequel.

- `dmesg` **(Display Message or Driver Message)**

    - **Utilité :** Affiche le contenu du tampon des messages du noyau (kernel ring buffer). C'est là que le noyau enregistre les informations sur le matériel détecté au démarrage, les erreurs matérielles, les messages des pilotes de périphériques, etc. C'est la première commande à utiliser si un matériel ne fonctionne pas correctement.
    - **Options utiles :**
        - `dmesg | tail` : Affiche les messages les plus récents.
        - `dmesg -w` (wait) : Affiche les messages existants puis reste actif, affichant les nouveaux messages du noyau au fur et à mesure qu'ils arrivent (utile pour voir les messages lors du branchement d'un périphérique USB, par exemple). Ctrl+C pour arrêter.
        - `dmesg -T` : Affiche les timestamps lisibles.

- **`strace <commande>` (System Trace)**
    - **Utilité :** Intercepte et enregistre les appels système effectués par un processus et les signaux qu'il reçoit. C'est un outil de débogage très puissant mais avancé, utilisé pour comprendre exactement ce que fait un programme au niveau du système d'exploitation (quels fichiers il ouvre, quelles connexions réseau il tente, etc.).
    - **Exemple :** `strace ls` montrera tous les appels système effectués par la simple commande `ls`.

- **`perf`**
    - **Utilité :** Un outil d'analyse de performance extrêmement puissant intégré au noyau Linux. Il permet de profiler l'utilisation CPU, d'analyser les 'cache misses', de tracer des événements noyau et utilisateur, etc. Son utilisation est complexe et sort du cadre de ce guide de base/intermédiaire.
    - **Installation :** `sudo pacman -S perf`.

8. Gestion Avancée du Réseau

Ces commandes permettent de configurer et diagnostiquer le réseau plus finement.

- **`ip a` / `ip addr` (IP Address)**
    - **Utilité :** La commande moderne (du paquet `iproute2`) pour afficher les informations sur toutes les interfaces réseau (Ethernet, Wi-Fi, loopback...) et les adresses IP (IPv4 et IPv6) qui leur sont assignées, ainsi que leur état (UP/DOWN). Remplace l'ancienne commande `ifconfig`.
    - **Exemple :** `ip a` liste toutes les interfaces. `ip a show eth0` montre uniquement les détails pour l'interface `eth0`.

- **`ip r` / `ip route` (IP Route)**
    - **Utilité :** Affiche la table de routage du noyau. Indique comment le système décide d'envoyer les paquets réseau vers différentes destinations (via quelle interface, quel routeur/gateway). Essentiel pour diagnostiquer les problèmes de connectivité au-delà du réseau local. Remplace l'ancienne commande `route`.
    - **Exemple :** `ip r` affiche la table. La ligne commençant par `default via ...` indique la passerelle par défaut.

- **`ip link set <interface> up/down`**

    - **Utilité :** Permet d'activer (`up`) ou de désactiver (`down`) une interface réseau spécifique. Nécessite les privilèges root (`sudo`). Utile pour réinitialiser une interface ou la désactiver temporairement.
    - **Exemple :** `sudo ip link set wlan0 down` désactive l'interface Wi-Fi `wlan0`. `sudo ip link set wlan0 up` la réactive.

- **`ping <hôte_ou_ip>`**
    - **Utilité :** Envoie des paquets ICMP "echo request" à une destination et attend des réponses "echo reply". C'est l'outil le plus basique pour vérifier si une machine est joignable sur le réseau et mesurer le temps aller-retour (latence).
    - **Fonctionnement :** Fonctionne en continu jusqu'à ce que vous l'arrêtiez avec Ctrl+C.
    - **Option utile :** `ping -c <nombre> <hôte>` envoie seulement `<nombre>` paquets puis s'arrête.
    - **Exemple :** `ping -c 4 archlinux.org` envoie 4 pings au site d'Arch Linux.

- **`traceroute <hôte_ou_ip>`**
    - **Utilité :** Affiche le chemin (la séquence de routeurs) que les paquets empruntent pour atteindre une destination sur Internet. Utile pour identifier où se situe un problème de connexion (quel routeur sur le chemin ne répond pas ou introduit une latence élevée).
    - **Installation :** Peut nécessiter `sudo pacman -S traceroute`.
    - **Exemple :** `traceroute archlinux.org`

- **`nmap <options> <cible>` (Network Mapper)**
    - **Utilité :** Un scanner de réseau extrêmement puissant utilisé pour découvrir des hôtes sur un réseau, les services (ports ouverts) qu'ils proposent, les systèmes d'exploitation qu'ils exécutent, etc. Outil essentiel pour l'audit de sécurité et l'administration réseau. **Avertissement : N'utilisez `nmap` que sur des réseaux ou des machines que vous avez l'autorisation explicite de scanner. Son utilisation non autorisée est illégale et contraire à l'éthique.**
    - **Installation :** `sudo pacman -S nmap`.
    - **Exemples courants :**
        - `nmap -sn 192.168.1.0/24` (ou `nmap -sP` sur d'anciennes versions) : Effectue un "Ping Scan" pour découvrir rapidement quels hôtes sont actifs sur le sous-réseau local `192.168.1.0/24` sans scanner leurs ports.
        - `nmap localhost` : Scanne les ports TCP les plus courants sur votre propre machine pour voir quels services sont en écoute.
        - `sudo nmap -sS -O <hôte_distant>` : Effectue un scan TCP SYN (furtif, `-sS`, nécessite `sudo`) et tente de détecter le système d'exploitation (`-O`) de l'hôte distant.

- **`tcpdump -i <interface> <filtre>`**
    - **Utilité :** Un outil de capture et d'analyse de paquets réseau en ligne de commande. Il permet d'écouter le trafic réseau passant par une interface spécifique et d'afficher les en-têtes (ou le contenu complet) des paquets correspondant à un filtre. Indispensable pour le diagnostic réseau avancé. Nécessite `sudo`.
    - **Installation :** `sudo pacman -S tcpdump`.
    - **Options/Filtres courants :**

        - `-i <interface>` : Spécifie l'interface réseau à écouter (ex: `eth0`, `wlan0`). `any` écoute sur toutes les interfaces.
        - `-nn` : Ne résout pas les adresses IP en noms d'hôtes ni les numéros de port en noms de services. Rend la sortie plus rapide et moins ambiguë.
        - `host <ip_ou_nom>` : Capture uniquement le trafic vers ou depuis cet hôte.
        - `port <numero>` : Capture uniquement le trafic utilisant ce port (source ou destination).
        - `tcp`, `udp`, `icmp` : Filtre par protocole.
    - **Exemple :** `sudo tcpdump -i any -nn host 8.8.8.8 and port 53` capture tout le trafic DNS (port 53) échangé avec le serveur 8.8.8.8 sur n'importe quelle interface, sans résolution de noms. Ctrl+C pour arrêter.

- **Pare-feu (`iptables`/`nftables`/`ufw`)**
    - **Concept :** Un pare-feu réseau filtre le trafic entrant et/ou sortant en fonction de règles prédéfinies (basées sur les adresses IP, les ports, les protocoles...). C'est un composant de sécurité essentiel. Arch Linux n'en active pas par défaut.
    - **Outils :**
        - `iptables` : L'outil traditionnel et très puissant de configuration du pare-feu Netfilter du noyau Linux. Sa syntaxe est complexe (chaînes, tables, règles).
        - `nftables` : Le successeur moderne d'`iptables`, offrant une syntaxe plus cohérente et de meilleures performances. Vise à remplacer `iptables`, `ip6tables`, `arptables`, `ebtables`.
        - `ufw` (Uncomplicated Firewall) : Un frontend conçu pour simplifier la gestion du pare-feu (il utilise `iptables` ou `nftables` en arrière-plan). Idéal pour les postes de travail ou les serveurs simples.
    - **Utilisation de `ufw` (exemple) :**
        1. Installation : `sudo pacman -S ufw`
        2. Activation du service : `sudo systemctl enable ufw.service --now` (active au démarrage et démarre immédiatement)
        3. Activation du pare-feu : `sudo ufw enable` (commence à bloquer par défaut le trafic entrant)
        4. Ajouter une règle (ex: autoriser SSH entrant) : `sudo ufw allow ssh` ou `sudo ufw allow 22/tcp`
        5. Vérifier le statut : `sudo ufw status verbose`

9. Gestion des Disques et Systèmes de Fichiers

Commandes pour gérer les périphériques de stockage, les partitions et les systèmes de fichiers.

- **`lsblk` (List Block Devices)**
    - **Utilité :** Affiche les informations sur tous les périphériques de stockage bloc disponibles (disques durs, SSD, clés USB, cartes SD...) et les partitions qu'ils contiennent, sous forme d'arbre. Indique aussi la taille et, si monté, le point de montage de chaque système de fichiers. Très utile pour avoir une vue d'ensemble rapide des disques.
    - **Exemple :** `lsblk`

- **`df -h` (Disk Free)**

    - **Utilité :** Rapporte l'utilisation de l'espace disque pour chaque système de fichiers *monté*. Affiche la taille totale, l'espace utilisé, l'espace disponible, le pourcentage d'utilisation et le point de montage.
    - **Option `-h` (Human-readable) :** Affiche les tailles en Ko, Mo, Go, etc., ce qui est beaucoup plus pratique que les blocs par défaut.
    - **Exemple :** `df -h`

- **`du -sh <dossier_ou_fichier>` (Disk Usage)**

    - **Utilité :** Estime et affiche l'espace disque occupé par un fichier ou un dossier (et son contenu).
    - **Options courantes :**
        - `-s` (Summary) : Affiche uniquement le total pour l'argument donné (le dossier), pas pour chaque sous-élément.
        - `-h` (Human-readable) : Affiche les tailles en Ko, Mo, Go...
    - **Exemple :** `du -sh /home/votre_utilisateur` affiche la taille totale de votre répertoire personnel.
    - **Usage avancé :** `du -h --max-depth=1 /var/log | sort -hr` affiche la taille de chaque sous-dossier direct dans `/var/log` et les trie par taille décroissante (`sort -hr`). Utile pour trouver où l'espace disque est utilisé.

- **`fdisk -l`**

    - **Utilité :** Outil historique pour lister les tables de partition des disques. Fonctionne bien pour le schéma de partitionnement MBR (Master Boot Record). Peut aussi lire les tables GPT mais `gdisk` ou `parted` sont souvent préférés pour GPT. Nécessite généralement `sudo`.
    - **Exemple :** `sudo fdisk -l` liste les partitions de tous les disques détectés.

- **`parted -l`**

    - **Utilité :** Un autre outil pour afficher les informations sur les disques et les tables de partition. Il gère bien MBR et GPT (GUID Partition Table), le standard moderne. Nécessite souvent `sudo`.
    - **Exemple :** `sudo parted -l`

- **`sudo fdisk /dev/sdX` / `sudo gdisk /dev/sdX` / `sudo parted /dev/sdX`**

    - **Utilité :** Ce sont les outils interactifs en ligne de commande pour **modifier** les tables de partition : créer, supprimer, redimensionner (parted), changer le type de partitions.
        - `fdisk` : Principalement pour MBR.
        - `gdisk` : Spécialisé et recommandé pour GPT.
        - `parted` : Très puissant, gère MBR/GPT et peut effectuer certaines opérations non interactivement.
    - **AVERTISSEMENT EXTRÊME : La modification des partitions est une opération très risquée qui peut entraîner une perte totale et irréversible des données sur le disque si elle est mal effectuée. Sauvegardez toujours vos données importantes avant d'utiliser ces outils. Assurez-vous de cibler le bon disque (`/dev/sda`, `/dev/sdb`, `/dev/nvme0n1`...).**

- **`sudo mkfs.<type> /dev/sdXN` (Make Filesystem)**

    - **Utilité :** Crée un système de fichiers (formate) sur une partition existante (`/dev/sdXN`, ex: `/dev/sda1`). C'est l'étape nécessaire après avoir créé une partition pour pouvoir y stocker des

fichiers.
    - `<type>` **:** Spécifie le type de système de fichiers à créer. Exemples courants :
        - `mkfs.ext4` : Le système de fichiers standard pour Linux. Journalisé, robuste.
        - `mkfs.vfat` : FAT32, pour une compatibilité maximale (clés USB, cartes SD, partage avec Windows). Limité en taille de fichier (4Go).
        - `mkfs.xfs` : Performant pour les gros fichiers et systèmes de fichiers.
        - `mkfs.btrfs` : Système de fichiers moderne avec des fonctionnalités avancées (snapshots, compression...).
    - **AVERTISSEMENT : Formater une partition efface définitivement toutes les données qui s'y trouvaient.**
    - **Exemple :** `sudo mkfs.ext4 /dev/sdb1` formate la première partition du disque `sdb` en ext4.

- **`sudo mount /dev/sdXN /mnt/point_de_montage`**

    - **Utilité :** Attache le système de fichiers d'une partition (ou d'un autre périphérique bloc) à un répertoire existant (le point de montage) dans l'arborescence principale. C'est ce qui rend le contenu de la partition accessible.
    - **Point de montage :** Doit être un répertoire vide existant. Les conventions courantes sont d'utiliser des sous-dossiers dans `/mnt` pour les montages temporaires ou `/media` pour les médias amovibles.
    - **Exemple :** `sudo mount /dev/sdb1 /mnt/usb_drive` monte la partition `sdb1` sur le dossier `/mnt/usb_drive`.

- **`sudo umount /dev/sdXN`** **ou** **`sudo umount /mnt/point_de_montage`**

    - **Utilité :** Détache ("démonte") un système de fichiers de son point de montage. C'est une étape **essentielle** avant de débrancher physiquement un périphérique amovible (clé USB, disque externe) pour s'assurer que toutes les données en attente ont été écrites et que le système de fichiers est dans un état cohérent.
    - **Exemple :** `sudo umount /mnt/usb_drive` démonte le périphérique monté sur `/mnt/usb_drive`.

- **`/etc/fstab`** **(FileSystem TABle)**

    - **Utilité :** Fichier de configuration critique qui liste les systèmes de fichiers que le système doit monter automatiquement au démarrage. Chaque ligne décrit un système de fichiers : le périphérique (par UUID, LABEL ou chemin /dev/), le point de montage, le type de système de fichiers, les options de montage (ex: `defaults`, `rw`, `ro`, `noatime`), et deux chiffres pour `dump` et `fsck` (souvent `0 0`).
    - **Modification :** À éditer avec précaution (utiliser `sudo nano /etc/fstab` ou `sudo vim /etc/fstab`). Une erreur dans `fstab` peut empêcher le système de démarrer correctement.

- **`smartctl -a /dev/sdX`** **(S.M.A.R.T. Control)**

    - **Utilité :** Interroge le système S.M.A.R.T. (Self-Monitoring, Analysis, and Reporting Technology) d'un disque dur ou SSD pour obtenir des informations sur son état de santé, ses attributs (température, nombre d'heures de fonctionnement, erreurs détectées...) et lancer des auto-tests. Utile pour anticiper une panne matérielle.
    - **Installation :** `sudo pacman -S smartmontools`.

- **Option `-a` (all) :** Affiche toutes les informations S.M.A.R.T. disponibles pour le disque.
- **Exemple :** `sudo smartctl -a /dev/sda`

10. Scripting et Automatisation

Écrire des scripts pour automatiser des tâches répétitives ou complexes.

- **Shell Scripting (Bash)**
  - **Concept :** Le shell (Bash étant le plus courant sous Linux) est lui-même un langage de programmation. Un script shell est simplement un fichier texte contenant une séquence de commandes shell, de structures de contrôle (boucles, conditions), de variables, etc., qui peuvent être exécutées ensemble.
  - **Éléments clés :**
    - `#!/bin/bash` (ou autre shell: `#!/bin/sh`) : Le "Shebang", doit être la toute première ligne. Indique au système quel interpréteur utiliser pour exécuter le script.
    - `chmod +x mon_script.sh` : Rend le fichier script exécutable.
    - `./mon_script.sh` : Exécute le script depuis le répertoire courant.
    - Variables : `MA_VAR="Bonjour"`, `echo "Message : $MA_VAR"`
    - Conditions : `if [ "$USER" == "root" ]; then echo "Admin !"; else echo "Utilisateur standard"; fi`
    - Boucles : `for f in *.txt; do echo "Traitement de $f"; done`
    - Fonctions : `sauvegarde() { rsync -avz source/ destination/; }; sauvegarde`
  - **Utilité :** Sauvegardes automatisées, traitement de fichiers en lot, configuration système, surveillance simple...
- **`cron` / `crontab -e`**
  - **Concept :** Le démon `cron` est le service système traditionnel pour exécuter des tâches planifiées (appelées "cron jobs") à des moments précis ou à intervalles réguliers. Chaque utilisateur peut avoir sa propre table de cron.
  - **`crontab -e` :** La commande pour éditer la table de cron de l'utilisateur courant. Ouvre un éditeur de texte où chaque ligne représente une tâche planifiée.
  - **Format d'une ligne crontab :** `MIN HOUR JOUR MOIS JOUR_SEMAINE COMMANDE`
    - `*` signifie "chaque" (ex: chaque minute, chaque heure).
    - Exemples :
      - `0 2 * * * /chemin/vers/script_sauvegarde.sh` : Exécute le script tous les jours à 2h00 du matin.
      - `*/15 * * * * /chemin/vers/script_surveillance.sh` : Exécute le script toutes les 15 minutes.
  - **Activation :** Le service `cronie` (ou équivalent) doit être installé et activé (`sudo systemctl enable cronie.service --now`).
- **`systemd Timers`**
  - **Concept :** L'approche moderne de la planification de tâches, intégrée à `systemd`. Offre plus de flexibilité que `cron` (ex: déclenchement basé sur le temps écoulé depuis le dernier démarrage,

exécution après un événement spécifique, meilleure intégration avec la journalisation `journalctl`).
    - **Fonctionnement :** Nécessite généralement deux fichiers "unit" dans `/etc/systemd/system/` (ou `~/.config/systemd/user/` pour les tâches utilisateur) :
        1. Un fichier `.service` : Décrit la commande ou le script à exécuter (similaire à un service systemd normal).
        2. Un fichier `.timer` (portant le même nom que le `.service`) : Décrit quand le service doit être démarré (ex: `OnCalendar=daily`, `OnBootSec=5min`, `OnUnitActiveSec=1h`).
    - **Activation :** Une fois les fichiers créés, le timer doit être activé et démarré : `sudo systemctl enable mon_timer.timer --now`.

11. Noyau (Kernel) et Modules

Commandes pour interagir avec le noyau Linux et ses modules chargeables.

- **`uname -r` (Unix Name)**
    - **Utilité :** Affiche la version exacte du noyau Linux (`r` pour release) qui est actuellement chargée et en cours d'exécution.
    - **Option `-a` (all) :** `uname -a` affiche plus d'informations : nom du noyau, nom d'hôte, version du noyau, informations de compilation, architecture machine, nom du système d'exploitation.
    - **Exemple :** `uname -r` pourrait afficher `6.8.7-arch1-1`.
- **`lsmod` (List Modules)**
    - **Utilité :** Liste tous les modules du noyau qui sont actuellement chargés en mémoire. Affiche le nom du module, sa taille, le nombre de fois où il est utilisé, et les autres modules qui en dépendent. Utile pour vérifier si un pilote matériel spécifique est chargé.
    - **Exemple :** `lsmod | grep nvidia` (si vous avez une carte Nvidia) pourrait montrer les modules du pilote Nvidia chargés.
- **`sudo modprobe <module>` (Module Probe)**
    - **Utilité :** Charge un module du noyau en mémoire (ainsi que ses dépendances). C'est la commande standard pour charger manuellement un pilote ou une fonctionnalité du noyau qui n'est pas chargée automatiquement.
    - **Exemple :** `sudo modprobe vboxdrv` (si VirtualBox est installé) charge le module principal de VirtualBox.
- **`sudo modprobe -r <module>` (Module Probe, Remove)**
    - **Utilité :** Décharge (supprime) un module du noyau de la mémoire, s'il n'est pas actuellement utilisé par un processus ou un autre module.
    - **Exemple :** `sudo modprobe -r vboxdrv` tente de décharger le module VirtualBox.
- **`modinfo <module>` (Module Information)**
    - **Utilité :** Affiche des informations détaillées sur un module spécifique, qu'il soit chargé ou non (il cherche dans les répertoires de modules). Indique le chemin du fichier du module, sa licence, sa description, ses dépendances, et les paramètres qu'il accepte.

    - **Exemple :** `modinfo iwlwifi` affiche des informations sur le module du pilote Wi-Fi Intel.

- **`/etc/mkinitcpio.conf` et `sudo mkinitcpio -P`**
    - **Concept `mkinitcpio` :** `mkinitcpio` (Make Initial CPIO) est l'outil utilisé par Arch pour créer l'**initramfs** (Initial RAM File System). L'initramfs est une petite archive CPIO compressée contenant un noyau minimaliste et les modules essentiels (ex: pour le disque dur, le système de fichiers racine) nécessaires pour démarrer le système *avant* que le vrai système de fichiers racine ne soit accessible.
    - **`/etc/mkinitcpio.conf` :** Le fichier de configuration principal de `mkinitcpio`. On y définit quels modules, binaires, fichiers et "hooks" (scripts d'automatisation) doivent être inclus dans l'initramfs. On le modifie par exemple pour ajouter des modules pour le chiffrement de disque (LUKS), LVM, ou des pilotes spécifiques nécessaires très tôt au démarrage.
    - **`sudo mkinitcpio -P` (Presets) :** La commande pour régénérer *toutes* les images initramfs définies dans les fichiers "preset" situés dans `/etc/mkinitcpio.d/` (typiquement un preset pour le noyau standard `linux`, un pour le noyau `linux-lts`, etc.). **Il est crucial d'exécuter cette commande après chaque mise à jour du noyau ou après avoir modifié `/etc/mkinitcpio.conf` pour que les changements soient pris en compte au prochain démarrage.**

12. Gestion des Utilisateurs et Groupes (Avancé)

Commandes pour gérer les comptes utilisateurs et les groupes. La plupart nécessitent `sudo`.

- **`whoami` (Who Am I)**
    - **Utilité :** Affiche simplement le nom d'utilisateur de l'utilisateur actuellement connecté dans ce terminal.
    - **Exemple :** `whoami`

- **`id`**
    - **Utilité :** Affiche des informations d'identification plus complètes pour l'utilisateur courant (ou un autre utilisateur si spécifié, ex: `id root`) :
        - `uid` (User ID) : Numéro unique identifiant l'utilisateur.
        - `gid` (Group ID) : Numéro du groupe principal de l'utilisateur.
        - `groups` : Liste de tous les groupes (numéros et noms) auxquels l'utilisateur appartient.
    - **Exemple :** `id`

- **`sudo useradd <options> <user>` (User Add)**
    - **Utilité :** Crée un nouveau compte utilisateur.
    - **Options importantes :**
        - `-m` : Crée automatiquement le répertoire personnel (`/home/<user>`) de l'utilisateur avec les permissions par défaut. **Presque toujours souhaitable.**
        - `-G <groupes>` : Ajoute l'utilisateur à une liste de groupes supplémentaires (séparés par des virgules, sans espace). Ex: `-G wheel,audio,video`. Le groupe `wheel` est souvent utilisé sous Arch pour accorder les droits `sudo`.
        - `-s <shell>` : Spécifie le shell de connexion par défaut pour l'utilisateur (ex: `-s /bin/bash`). Si omis, utilise souvent `/bin/sh` par défaut.

    - **Exemple :** `sudo useradd -m -G wheel -s /bin/bash nouvelutilisateur`

- **`sudo passwd <user>` (Password)**

    - **Utilité :** Définit (s'il n'existe pas) ou change le mot de passe d'un utilisateur spécifié. Si utilisé sans nom d'utilisateur (`sudo passwd`), change le mot de passe de root. Si utilisé par un utilisateur normal sans `sudo` (`passwd`), change son propre mot de passe.
    - **Exemple :** `sudo passwd nouvelutilisateur`

- **`sudo usermod <options> <user>` (User Modify)**

    - **Utilité :** Modifie les paramètres d'un compte utilisateur existant.
    - **Option cruciale :**
        - `-aG <groupe>` : L'option `-a` (append) doit être utilisée avec `-G` pour *ajouter* l'utilisateur à un groupe *sans le retirer* des autres groupes auxquels il appartient déjà. **Oublier `-a` lors de l'utilisation de `-G` retirera l'utilisateur de tous les groupes sauf celui spécifié !**
    - **Exemple :** `sudo usermod -aG audio nom_utilisateur` ajoute `nom_utilisateur` au groupe `audio` (souvent nécessaire pour accéder au matériel audio).

- **`sudo groupadd <group>` (Group Add)**

    - **Utilité :** Crée un nouveau groupe sur le système.
    - **Exemple :** `sudo groupadd developpeurs`

- **`sudo userdel <user>` (User Delete)**

    - **Utilité :** Supprime un compte utilisateur.
    - **Option utile :**
        - `-r` : Supprime également le répertoire personnel (`/home/<user>`) et la boîte mail de l'utilisateur. Sans `-r`, le compte est supprimé mais les fichiers personnels restent.
    - **Exemple :** `sudo userdel -r utilisateur_a_supprimer`

- **`su <user>` / `sudo -i` / `sudo -s` (Switch User / Superuser Do)**

    - **Utilité :** Permettent de changer d'identité ou d'obtenir les privilèges root.
    - **`su <user>` :** Change l'utilisateur courant pour `<user>`. Demande le mot de passe de `<user>`. `su` sans argument tente de passer root (demande le mot de passe root). `su -` (ou `su -l <user>`) simule une connexion complète : charge le profil du nouvel utilisateur, change le répertoire courant vers son `/home`.
    - **`sudo <commande>` :** Exécute une seule `<commande>` avec les privilèges root (après avoir entré *votre* propre mot de passe, si vous êtes dans le groupe `wheel` et que `/etc/sudoers` est configuré correctement). C'est la méthode préférée pour les tâches administratives ponctuelles.
    - **`sudo -i` (interactive login) :** Ouvre un shell root interactif qui simule une connexion root complète (charge l'environnement de root, va dans `/root`).
    - **`sudo -s` (shell) :** Ouvre un shell root interactif mais conserve une partie de l'environnement de votre utilisateur d'origine.

- **`sudo visudo`**

    - **Utilité : La seule et unique commande recommandée pour éditer le fichier `/etc/sudoers`**, qui définit qui a le droit d'utiliser `sudo` et comment.

- **Pourquoi `visudo` ?** Il utilise `vi` (ou l'éditeur défini par la variable `EDITOR`) mais surtout, il verrouille le fichier `/etc/sudoers` pour éviter les éditions concurrentes et **effectue une vérification de la syntaxe avant de sauvegarder**. Cela empêche de sauvegarder un fichier `sudoers` corrompu qui vous bloquerait complètement hors de `sudo`. **Ne jamais éditer `/etc/sudoers` directement avec `nano` ou `vim` !**
- **Configuration courante :** Pour autoriser tous les membres du groupe `wheel` à utiliser `sudo`, décommentez (supprimez le `#` au début) la ligne `%wheel ALL=(ALL:ALL) ALL` dans le fichier ouvert par `visudo`.

13. Sécurité Élémentaire

Quelques pratiques et outils de base pour sécuriser votre système Arch.

- **Mises à jour régulières :** Répétons-le : `sudo pacman -Syu` fréquemment est la mesure de sécurité la plus importante pour corriger les vulnérabilités connues.
- **Pare-feu :** Activez et configurez un pare-feu (ex: `ufw`) pour limiter les connexions réseau entrantes aux seuls services nécessaires. Par défaut, bloquez tout le trafic entrant et n'autorisez explicitement que ce dont vous avez besoin (ex: port 22 pour SSH si nécessaire).
- **Privilèges minimum :** N'utilisez pas le compte `root` pour vos activités quotidiennes. Utilisez un compte utilisateur standard et n'utilisez `sudo` que lorsque c'est indispensable. Moins vous opérez avec des privilèges élevés, moins les dégâts potentiels d'une erreur ou d'une attaque sont importants.
- **Mots de passe forts :** Utilisez des mots de passe longs (15+ caractères), complexes (mélange majuscules, minuscules, chiffres, symboles) et uniques pour votre compte utilisateur et surtout pour le compte `root`. Envisagez un gestionnaire de mots de passe.
- **SSH Hardening (si le serveur SSH `sshd.service` est actif) :**
  - **Désactiver la connexion root :** Dans `/etc/ssh/sshd_config` (éditez avec `sudo nano` ou `sudo vim`), mettez `PermitRootLogin no`. Force l'utilisation d'un compte utilisateur normal puis `sudo`.
  - **Privilégier l'authentification par clé SSH :** Beaucoup plus sécurisé que les mots de passe. Générez une paire de clés (`ssh-keygen`), copiez la clé publique sur le serveur (`ssh-copy-id user@host`), puis désactivez l'authentification par mot de passe dans `sshd_config` : `PasswordAuthentication no`.
  - **Changer le port par défaut (22) :** Optionnel, peut réduire le bruit des scans automatiques mais n'arrête pas une attaque ciblée (`Port 2222`).
  - **Redémarrer le service SSH :** Après toute modification de `sshd_config`, appliquez les changements avec `sudo systemctl restart sshd`.
- **`fail2ban`**
  - **Utilité :** Un service qui surveille les fichiers journaux (logs) à la recherche de tentatives d'authentification échouées répétées (ex: pour SSH, serveurs web, serveurs mail) et bannit automatiquement les adresses IP sources pour une durée déterminée en ajoutant une règle au pare-feu. Très efficace contre les attaques par force brute automatisées.
  - **Installation :** `sudo pacman -S fail2ban`
  - **Activation :** `sudo systemctl enable fail2ban --now`
  - **Configuration :** Ne modifiez pas `jail.conf`. Créez un fichier `jail.local` (ex: `sudo nano /etc/fail2ban/jail.local`) pour surcharger les paramètres par défaut et activer les "jails" spécifiques dont vous avez besoin (ex: `[sshd] enabled = true`).

14. Configuration et Maintenance

Maintenir un système Arch propre et fonctionnel.

- **Fichiers de Configuration :** Rappel : la configuration sous Arch (et Linux en général) se fait principalement en éditant des fichiers texte.
    - `/etc/` : Contient les fichiers de configuration globaux du système et des services.
    - `~/.config/` : Emplacement standard pour les fichiers de configuration spécifiques à l'utilisateur pour de nombreuses applications modernes.
    - `~/.*` : De nombreux autres fichiers de configuration utilisateur cachés directement dans le répertoire personnel (ex: `~/.bashrc`, `~/.profile`, `~/.vimrc`).
    - Apprendre à trouver et éditer ces fichiers est essentiel pour personnaliser votre environnement.
- **Maintenance Régulière :**
    - **Lire Arch News AVANT `sudo pacman -Syu` : Non négociable.** Le site officiel publie des annonces lorsqu'une mise à jour nécessite une intervention manuelle de l'utilisateur (ex: fusionner un fichier de configuration, exécuter une commande spécifique). Ignorer ces annonces est la cause la plus fréquente de systèmes cassés après une mise à jour.
    - **Gérer les fichiers `.pacnew`/`.pacsave` :** Après une mise à jour (`pacman -Syu`), `pacman` peut créer des fichiers avec ces extensions s'il met à jour un paquet dont vous aviez modifié le fichier de configuration.
        - `.pacnew` : La nouvelle version du fichier de configuration fournie par le paquet. Votre version modifiée est conservée.
        - `.pacsave` : Votre ancienne version modifiée sauvegardée (cela arrive moins souvent, typiquement si le paquet est supprimé).
        - **Action requise :** Vous devez **impérativement** comparer votre fichier de configuration avec le fichier `.pacnew` et fusionner manuellement les changements pertinents de la nouvelle version dans votre fichier. Ignorer les `.pacnew` peut entraîner des dysfonctionnements, des problèmes de sécurité, ou empêcher des services de démarrer.
        - **Outils :** Utilisez `pacdiff` (du paquet `pacman-contrib`) pour lister les fichiers `.pacnew`/`.pacsave` et lancer un outil de comparaison (diff). `pacdiff` utilise souvent `vimdiff` par défaut, mais peut être configuré pour utiliser des outils graphiques comme `meld` (`sudo pacman -S meld`, puis `DIFFPROG=meld pacdiff`).
    - **Nettoyer le cache `pacman` :** Périodiquement, utilisez `sudo pacman -Sc` pour supprimer les anciennes versions de paquets du cache et libérer de l'espace disque.
    - **Vérifier les services échoués :** `systemctl --failed` après un démarrage ou une mise à jour pour voir si des services système ont rencontré des problèmes.
    - **Consulter les logs système :** Utilisez `journalctl` pour investiguer les erreurs ou les avertissements.
        - `journalctl -b` : Voir les logs depuis le dernier démarrage.
        - `journalctl -p err -b` : Filtrer pour ne voir que les erreurs (priority `err` et supérieures) depuis le dernier démarrage.
        - `journalctl -f` : Suivre les logs en temps réel.
        - `journalctl -u <nom_service.service>` : Voir les logs spécifiques à un service.

Conclusion

Arch Linux offre une expérience Linux unique, axée sur la simplicité, le contrôle et la personnalisation. Maîtriser ces commandes, des plus basiques aux plus avancées, est la clé pour exploiter pleinement son potentiel. N'oubliez jamais la ressource inestimable qu'est l'Arch Wiki : consultez-la souvent. Explorez, apprenez, et surtout, construisez le système Linux qui vous ressemble !