Systèmes LINUX

Alves Lobo Michael

- https://www.linkedin.com/in/michael-alves-lobo
- https://github.com/kairel/learning
-
- Devops, dev, réseau et administration système
- En poste chez Vade/Hornet
- Automatisation
  - ansible
  - puppet
  - terraform
  - docker

SOMMAIRE

JOUR 1 - Fondamentaux et premiers pas

Introduction générale (45min)

- Historique de Linux et des systèmes Unix
- Les différentes distributions (Ubuntu, CentOS, Debian, etc.)
- Avantages et cas d'usage de Linux
- Architecture générale d'un système Linux

Installation et découverte de l'interface (1h15)

- Installation sur machine virtuelle ou dual boot
- Premier démarrage et configuration de base
- Interface graphique vs ligne de commande
- Le bureau Linux (GNOME, KDE, etc.)

SOMMAIRE

Premiers pas en ligne de commande (1h30)

- Le terminal et le shell
- Structure des répertoires Linux (/home, /etc, /var, etc.)
- Commandes de base : pwd, ls, cd, mkdir, rmdir
- Aide et documentation : man, help, --help

Manipulation de fichiers (1h45)

- Créer, copier, déplacer, supprimer : touch, cp, mv, rm
- Affichage du contenu : cat, less, more, head, tail
- Édition basique avec nano
- Liens symboliques et liens durs

SOMMAIRE

Permissions et propriétés (1h30)

- Système de permissions Unix (rwx)
- chmod, chown, chgrp
- Utilisateurs et groupes
- sudo et élévation de privilèges

Autres

- mount
- chroot

SOMMAIRE

JOUR 2 - Gestion du système et outils avancés

Gestion des processus (1h15)

- Qu'est-ce qu'un processus
- ps, top, htop
- Gestion des processus : kill, killall
- Processus en arrière-plan : &, nohup, jobs

Gestion des paquets (1h)

- Gestionnaires de paquets selon les distributions
- Installation/suppression de logiciels (apt, yum, dnf)
- Mise à jour du système
- Dépôts et sources de paquets

SOMMAIRE

SSH

- Introduction
- Installation
- Configuration
- Exemples pratiques et debug

SOMMAIRE

Redirection et pipes (1h)

- Redirection d'entrée/sortie : >, >>, <
- Pipes : |
- Filtres utiles : grep, sort, uniq, wc
- Exemples pratiques de traitement de données

Variables d'environnement et scripts basiques (1h15)

- Variables d'environnement : PATH, HOME, etc.
- Création de scripts bash simples
- Rendre un script exécutable
- Crontab et tâches planifiées

SOMMAIRE

Réseau et services (1h)

- Configuration réseau basique
- ping, wget, curl
- Services et démons
- systemctl (systemd)

Surveillance système et dépannage (1h)

- Espace disque : df, du
- Logs système : /var/log/
- Surveillance des ressources
- Commandes de dépannage courantes

SOMMAIRE

- **Plusieurs TP à la fin de chaque concept**
- **Un TP de fin de projet (½ journée)**

Introduction générale

- Historique de Linux et des systèmes Unix

Introduction générale

- Historique de Linux et des systèmes Unix

Unix, officiellement **UNIX**, est une famille de systèmes d'exploitation ouverts, multitâche et multi-utilisateur. Unix est dérivé de l'Unix d'origine créé par AT&T, le développement de ce dernier ayant commencé dans les années 1970 au centre de recherche de Bell Labs mené par Kenneth Thompson. Il est composé d'un noyau (qui assure la gestion de la mémoire, des entrées et sorties de bas niveau, l'enchaînement des tâches), d'un interpréteur ou superviseur (le *shell*), et de nombreux petits utilitaires, accomplissant chacun une action spécifique, commutables entre eux (mécanisme de « redirection ») et appelés depuis la ligne de commande.

Introduction générale

- Historique de Linux et des systèmes Unix

Philosophie d'Unix :

- **(presque) tout s'utilise comme un fichier**
- **I "Do one thing, do it well" (Doug McIlroy, l'inventeur des pipes Unix)**
- **I Write programs that do one thing and do it well.**
- **I Write programs to work together.**

Introduction générale

- Historique de Linux et des systèmes Unix

SYSTEME MULTI-TACHES

Les systèmes d'exploitation multi-tâches permettent de partager le temps du processeur entre plusieurs programmes, ainsi ceux-ci sembleront s'exécuter simultanément. Pour ce faire, les applications sont découpées en séquence d'instructions que l'on appelle tâches ou processus. Ces tâches seront tour à tour actives, en attente, suspendues ou détruites, suivant la priorité qui leur est associée et l'état d'avancement du programme. Un système est dit préemptif lorsqu'il possède un ordonnanceur (aussi appelé planificateur ou scheduler), qui répartit, selon différents critères de priorité, le temps machine entre les différentes tâches qui en font la demande. C'est le planificateur qui active et qui décide d'arrêter les tâches. Ce système est relativement fiable (ex: Windows XP, Windows 7)

Introduction générale

- Historique de Linux et des systèmes Unix

SYSTEME MULTI-UTILISATEURS

Plusieurs utilisateurs à travers des terminaux ou à travers le réseau peuvent accéder aux ressources de l'ordinateur. Ceci rejoint un peu le multi-tâches, dans le sens où le microprocesseur partage son "temps" entre plusieurs utilisateurs donc plusieurs programmes.

Introduction générale

- Historique de Linux et des systèmes Unix

1991 : Linus Torvalds, étudiant finlandais de 21 ans

Tout a démarré par un message posté par Linus Benedict Torvalds sur le réseau Usenet le 25 août 1991. Dans ce document, le jeune homme de 22 ans écrivait « je suis en train de faire un système d'exploitation (libre) (il s'agit d'un hobby, il ne sera pas grand, ni professionnel comme GNU) pour les clones 386 (486 AT) ». Il précise ensuite que ce « projet est en gestation depuis avril et commence à être prêt ». Dans son message, il demande l'avis des utilisateurs sur ce qu'ils aiment ou pas dans minix, car « mon OS lui ressemble un peu ». A la fin de son post, Linus Torvalds glisse que toutes les suggestions sont les bienvenues, mais ne promet pas de les mettre en œuvre.

La suite de l'aventure débute quelques mois après, le 5 octobre 1991, date à laquelle le développeur finlandais publie les sources d'un noyau libre de type minix pour 386-AT. Il indique dans son texte que les sources de son projet « peuvent être trouvées à nic.funet.fi (12.214.6.100) dans le répertoire /pub/OS/Linux ». L'histoire de Linux pouvait alors s'écrire et surtout se développer.

Introduction générale

- Historique de Linux et des systèmes Unix

EN BREF

- **1991** : Linus Torvalds commence le développement du noyau **Linux**.
- **17 juillet 1993** : Sortie de **Slackware Linux**, la plus ancienne distribution encore active.
- **Mars 2002** : Lancement de **Gentoo** et **Arch Linux**.
- **Septembre 2003** : Naissance du projet **Fedora**.
- **Mai 2004** : Sortie de **CentOS**.
- **Octobre 2004** : Publication de la première version de **Ubuntu**, marquant une avancée majeure dans la démocratisation de **Linux**.
- **Août 2006** : Première version de **Linux Mint** sous le nom de code **Ada**.

Introduction générale

- Historique de Linux et des systèmes Unix

GNU

Le projet GNU a développé un ensemble complet d'outils libres destinés à Unix™ et aux systèmes d'exploitation de type Unix, tel que GNU/Linux. Ces outils permettent aux utilisateurs d'accomplir aussi bien les tâches les plus simples (copier ou effacer un fichier) que les plus complexes (écrire et compiler des programmes, éditer de façon sophistiquée dans un grand nombre de formats).

Le nom « GNU » **est un acronyme récursif pour « GNU's Not Unix »**

Introduction générale

- Les distribution Linux

Une **distribution Linux**® est un système d'exploitation prêt à être installé, conçu à partir d'un noyau **Linux** et qui prend en charge des référentiels, des bibliothèques et des programmes utilisateur. Chaque version d'un fournisseur ou d'une communauté est appelée **distribution**

Noyau Linux + logiciels + gestionnaire de paquets + support

Plus de 600 distributions actives

Adaptation à différents besoins et publics

Introduction générale

- Les distribution Linux

Debian et dérivées :

- Debian : stabilité, serveurs, communautaire
- Ubuntu : desktop, facilité d'usage, support Canonical
- Linux Mint : interface familière pour les débutants
- Gestionnaire : APT (.deb)

Introduction générale

- Les distribution Linux

Red Hat et dérivées :

- RHEL : entreprise, support commercial
- CentOS/AlmaLinux : gratuit, base RHEL
- Fedora : technologies récentes, communautaire
- Gestionnaire : YUM/DNF (.rpm)

Introduction générale

- Les distribution Linux

Autres familles importantes :

- SUSE : entreprise européenne
- Arch Linux : utilisateurs avancés, rolling release
- Gentoo : compilation des sources

Introduction générale

- Les distribution Linux

Comment choisir ?

- Débutant : Ubuntu, Linux Mint
- Serveur : Debian, CentOS/AlmaLinux, Ubuntu Server
- Entreprise : RHEL, SLES, Ubuntu Pro
- Développeur : Fedora, Ubuntu
- Expert : Arch, Gentoo

Introduction générale

- Avantages et cas d'usage

Avantages techniques

- **Sécurité** : moins de malwares, permissions strictes, mises à jour fréquentes
- **Performance** : utilisation optimale des ressources, pas de "bloatware"
- **Stabilité** : serveurs avec uptime de plusieurs années
- **Personnalisation** : interface, noyau, services modulaires

Introduction générale

- Avantages et cas d'usage

Avantages économiques

- Gratuit (pas de licence)
- Matériel moins cher (pas de "taxe Windows")
- Cycle de vie du matériel prolongé
- Support communautaire gratuit

Introduction générale

- Avantages et cas d'usage

Cas d'usage principaux (4 min)

- **Serveurs** : 96% du top 1M des serveurs web
- **Cloud** : AWS, Google Cloud, Azure utilisent Linux
- **Développement** : outils natifs, conteneurs Docker
- **Scientifique** : calcul haute performance, recherche
- **Embarqué** : IoT, routeurs, smartphones (Android)
- **Desktop** : alternative à Windows/macOS

Introduction générale

- Architecture générale de Linux

Applications utilisateur (Firefox, LibreOffice...)

↕

Interface utilisateur (Shell, GUI)

↕

Services système (réseau, impression...)

↕

Noyau Linux (kernel)

↕

Matériel (CPU, RAM, disques...)

Introduction générale

- Architecture générale de Linux

Le noyau Linux

- Cœur du système d'exploitation
- Gestion de la mémoire, processus, fichiers
- Pilotes matériels (drivers)
- Interface entre logiciel et matériel
- Monolithique mais modulaire

Introduction générale

- Architecture générale de Linux

L'espace utilisateur

- Programmes et processus utilisateurs
- Bibliothèques système (libc)
- Services et démons (systemd)
- Interface graphique (X11, Wayland)

Installation et découverte de l'interface

- Préparation a l'installation

Logiquement vous avez tous un dual boot , ce qui veut dire que vous pouvez démarrer votre pc soit avec Windows soit avec Linux

Linux est donc déja installé sur votre pc, néanmoins si c'est possible on va faire une installation via virtualbox ou vmware juste pour voir au moins une fois une installation d'un Linux.

On utilisera le dual boot pour tous les exercices suivant

Installation et découverte de l'interface

- TP: Préparation a l'installation

- Installer utiliser un outil de virtualisation (virtualbox, vmware)
- Téléchargement de l'ISO Ubuntu LTS (version stable) avec interface
- créer une nouvelle machine a partir cet iso
- choisir le mode réseau NAT pour garder la connectivité internet
- disque de 20G suffit largement

A vous de jouer

Installation et découverte de l'interface

- TP: Préparation a l'installation

Maintenant on va se connecter avec le dual-boot

On vérifie que tout fonctionne:

- le user pour se connecter
- le réseau (on a bien accès a internet)
- les droits du user
    - on va upgrader le système
        - on ouvre un terminal (apt install, apt upgrade)

Installation et découverte de l'interface

- Découverte de l'interface

Vue d'ensemble du bureau

Barre supérieure :

- Menu Activités (coin supérieur gauche)
- Horloge et notifications (centre)
- Indicateurs système (droite) : réseau, son, batterie, utilisateur

Installation et découverte de l'interface

- Découverte de l'interface

Vue d'ensemble du bureau

Dock Ubuntu :

- Applications favorites sur le côté gauche
- Applications ouvertes
- Corbeille en bas
- Personnalisation possible

Installation et découverte de l'interface

- Découverte de l'interface

Vue d'ensemble du bureau

Gestion des fenêtres :

- Maximiser, minimiser, fermer
- Redimensionnement et déplacement
- Espaces de travail multiples

Installation et découverte de l'interface

- Découverte de l'interface

Navigation et raccourcis

Raccourcis essentiels :

- `Super` (touche Windows) : Vue des activités
- `Super + A` : Applications
- `Alt + Tab` : Basculer entre applications
- `Super + L` : Verrouiller la session
- `Ctrl + Alt + T` : Ouvrir un terminal

Installation et découverte de l'interface

- Découverte de l'interface

Navigation et raccourcis

Menu Activités :

- Vue d'ensemble des fenêtres ouvertes
- Recherche universelle (applications, fichiers, paramètres)
- Espaces de travail
- Grille d'applications

Installation et découverte de l'interface

- Découverte de l'interface

Navigation et raccourcis

Gestionnaire de fichiers Nautilus (5 min)

- Navigation dans l'arborescence
- Dossiers principaux : Documents, Téléchargements, Images
- Raccourcis et signets
- Vue en liste vs icônes
- Propriétés des fichiers

Installation et découverte de l'interface

- Interface graphique vs ligne de comande

Interface graphique :

- Intuitive pour débuter
- Idéale pour multimedia, bureautique
- Consomme plus de ressources
- Limitation sur serveurs distants

Installation et découverte de l'interface

- Interface graphique vs ligne de comande

Ligne de commande :

- Puissante et flexible
- Automatisation possible (scripts)
- Fonctionne à distance (SSH)
- Plus rapide pour certaines tâches
- Courbe d'apprentissage plus raide

Installation et découverte de l'interface

- Interface graphique vs ligne de comande

Quand utiliser quoi ?

- **GUI** : Navigation fichiers, édition texte simple, web, multimedia
- **CLI** : Administration système, traitement de données, serveurs, automatisation
- **Hybrid** : Développement, configuration système

Installation et découverte de l'interface

- TP: quelques exercices

- **Navigation interface** : Ouvrir 3 applications, basculer entre elles, utiliser les espaces de travail
- **Personnalisation** : Changer le fond d'écran, passer en thème sombre
- **Gestionnaire de fichiers** : Créer un dossier, y placer un fichier téléchargé
- **Premier terminal** : Ouvrir le terminal, taper `whoami` et `pwd`

Premiers pas en ligne de commande

- Le terminal et le shell

Qu'est-ce que le terminal ?

- **Définitions** :
  - Terminal : interface texte pour interagir avec le système
  - Shell : interpréteur de commandes (bash, zsh, fish...)
  - Console : terminal physique (distinction historique)

Premiers pas en ligne de commande

- Le terminal et le shell

Qu'est-ce que le terminal ?

Accès au terminal :

- `Ctrl + Alt + T` (raccourci Ubuntu)
- Menu Applications → Terminal
- Clic droit sur bureau → "Ouvrir un terminal"
- Terminal intégré dans l'éditeur de code

Premiers pas en ligne de commande

- Le terminal et le shell

Qu'est-ce que le terminal ?

Anatomie du prompt :

utilisateur@machine:~$

`utilisateur` : nom de l'utilisateur connecté

`@` : séparateur

`machine` : nom de l'ordinateur (hostname)

`:` : séparateur

`~` : répertoire courant (~ = home)

`$` : utilisateur normal (`#` pour root)

Premiers pas en ligne de commande

- Le terminal et le shell

Types de shells

- **Bash** (Bourne Again Shell) : standard sur Ubuntu
- **Zsh** : moderne, autocomplete avancée
- **Fish** : convivial pour débutants
- Vérifier son shell : `echo $SHELL`
- Historique des commandes : flèches ↑↓, `history`

Premier pas en ligne de commande

- Système de fichiers hiérarchique

```
/                    (racine, root)
├── home/            (dossiers utilisateurs)
│   ├── utilisateur/ (votre dossier personnel)
│   └── autres_users/
├── etc/             (configuration système)
├── var/             (données variables)
│   ├── log/         (fichiers de logs)
│   └── www/         (sites web)
├── usr/             (programmes utilisateur)
│   ├── bin/         (exécutables)
│   └── lib/         (bibliothèques)
├── bin/             (commandes essentielles)
├── tmp/             (fichiers temporaires)
├── dev/             (périphériques)
├── proc/            (informations processus)
└── mnt/             (points de montage)
```

Premier pas en ligne de commande

- Systéme de fichiers hiérarchique

Répertoires importants détaillés

`/` **(root)** : Point de départ de tout le système `/home` : Dossiers personnels des utilisateurs

- Équivalent de "Users" sous Windows
- Chaque utilisateur a son sous-dossier

`/etc` : Fichiers de configuration

- `/etc/passwd` : liste des utilisateurs
- `/etc/hosts` : résolution de noms
- `/etc/apt/sources.list` : sources de paquets

Premier pas en ligne de commande

- Systéme de fichiers hiérarchique

Répertoires importants détaillés

`/var` : Données qui changent fréquemment

- `/var/log` : logs système et applications
- `/var/www` : sites web par défaut

`/usr` : Programmes et données utilisateur

- `/usr/bin` : exécutables installés
- `/usr/share` : données partagées

Premier pas en ligne de commande

- Systéme de fichiers hiérarchique

Chemin absolu : commence par `/`

- `/home/utilisateur/Documents`
- `/etc/passwd`

Chemin relatif : par rapport au répertoire courant

- `Documents/fichier.txt`
- `../autre_dossier`
- `.` = répertoire courant
- `..` = répertoire parent

Premier pas en ligne de commande

- Commandes de navigation

```
pwd                          # Print Working Directory
# Affiche : /home/utilisateur
```

Indique où vous êtes dans l'arborescence

Utile quand le prompt n'affiche pas le chemin complet

Premier pas en ligne de commande

- Commandes de navigation

```
ls                  # Liste le contenu du répertoire courant
ls -l               # Liste détaillée (permissions, taille, date)
ls -a               # Affiche les fichiers cachés (commençant par .)
ls -la              # Combinaison des options
ls -lh              # Tailles lisibles (K, M, G)
ls /etc             # Liste le contenu de /etc
ls *.txt            # Tous les fichiers .txt
```

Premier pas en ligne de commande

- Commandes de navigation

Interprétation de

`ls -l` :

drwxr-xr-x 2 user user 4096 déc 15 10:30 Documents

-rw-r--r-- 1 user user 1234 déc 15 09:15 fichier.txt

1er caractère : type (`d`=dossier, `-`=fichier, `l`=lien)

9 caractères suivants : permissions (rwx pour user/group/others)

Nombre de liens, propriétaire, groupe, taille, date, nom

Premier pas en ligne de commande

- Commande de navigation

```
cd                      # Retour au home (~)
cd /                    # Va à la racine
cd ..                   # Remonte d'un niveau
cd ../..                # Remonte de deux niveaux
cd Documents            # Va dans Documents (chemin relatif)
cd /home/user/Documents # Va dans Documents (chemin absolu)
cd -                    # Retourne au répertoire précédent
   ~                    # Va au home (équivalent à cd seul)
```

Premier pas en ligne de commande

- Gestion des répertoires

```
mkdir nouveau_dossier            # Crée un dossier
mkdir dossier1 dossier2          # Crée plusieurs dossiers
mkdir -p parent/enfant/petit     # Crée l'arborescence complète
mkdir "dossier avec espaces"     # Guillemets pour les espaces
mkdir -v dossier                 # Mode verbose (affiche ce qui est fait)
```

Premier pas en ligne de commande

- Gestion des répertoires

```
rmdir dossier_vide              # Supprime un dossier vide
rmdir -v dossier                # Mode verbose
rmdir dossier1 dossier2         # Supprime plusieurs dossiers
```

Premier pas en ligne de commande

- Aide et documentation

```
man ls                  # Manuel de la commande ls
man mkdir               # Manuel de mkdir
man man                 # Manuel du manuel !
```

Flèches ou Page Up/Down : naviguer

`/mot` : rechercher "mot"

`n` : occurrence suivante

`q` : quitter

Premier pas en ligne de commande

- Aide et documentation

```
ls --help                # Aide concise de ls
mkdir --help             # Aide de mkdir
cd --help                # Ne fonctionne pas (commande interne)
```

Premier pas en ligne de commande

- TP 1

 # 1. Affichez votre répertoire courant

 # 2. Listez le contenu de votre home

 # 3. Allez à la racine du système

 # 4. Listez le contenu de /etc

 # 5. Revenez à votre home

Premier pas en ligne de commande

- TP 2

 # 1. Créez l'arborescence suivante dans votre home :

 # projet/

 # ├── src/

 # ├── doc/

 # └── test

 # Bonus : en une seule commande :

Premier pas en ligne de commande

- TP 3

 # 1. Explorez /usr/bin et comptez approximativement les commandes

 # 2. Trouvez votre shell par défaut

 # 3. Affichez l'aide de la commande ls

Manipulation de fichiers

- Création de fichier

```
touch fichier.txt                    # Crée un fichier vide
touch fichier1.txt fichier2.txt      # Crée plusieurs fichiers
touch "fichier avec espaces.txt"     # Nom avec espaces
touch ~/.bashrc                      # Met à jour la date de modification
```

Manipulation de fichiers

- Création de fichier

Fonctionnalités de `touch` :

- Crée un fichier vide s'il n'existe pas
- Met à jour la date de modification s'il existe
- Préserve le contenu existant
- Très utile pour les tests et scripts

Manipulation de fichiers

- Création de fichier

- **Autres méthodes de création**

```
echo "Contenu" > nouveau.txt          # Crée avec contenu
echo "Ligne 2" >> nouveau.txt         # Ajoute une ligne
cat > fichier.txt                     # Saisie interactive (Ctrl+D pour finir)
printf "Texte formaté\n" > file.txt # Création avec printf
```

`>` : redirige et écrase le contenu

`>>` : redirige et ajoute au contenu

`cat >` : permet la saisie multilignes

Manipulation de fichiers

- Copie de fichiers et de dossiers

```
cp source.txt destination.txt          # Copie simple
cp fichier.txt /tmp/                   # Copie vers un dossier
cp fichier.txt /tmp/nouveau_nom.txt    # Copie avec nouveau nom
cp *.txt backup/                       # Copie tous les .txt
cp -v source.txt dest.txt              # Mode verbose
cp -i source.txt dest.txt              # Mode interactif (confirme
l'écrasement)
```

Manipulation de fichiers

- Copie de fichiers et de dossiers

Options importantes de `cp` :

- `-v` : verbose (affiche ce qui est copié)
- `-i` : interactif (demande confirmation avant écrasement)
- `-n` : no-clobber (n'écrase pas)
- `-u` : update (copie seulement si source plus récente)

Manipulation de fichiers

- Copie de dossiers

```
cp -r dossier_source/ dossier_dest/ # Copie récursive
cp -r src/ backup/ # Sauvegarde complète
cp -r projet/ /tmp/projet_backup/ # Copie vers autre emplacement
cp -rv dossier/ backup/ # Verbose + récursif
```

Manipulation de fichiers

- Cas particulier et piége

 # Différence importante :

cp -r dossier/* dest/ *# Copie le CONTENU de dossier dans dest/*

cp -r dossier dest/ *# Copie dossier LUI-MÊME dans dest/ # Vérification :*

ls -la dest/ *# Voir le résultat*

`mv` vs `cp` :

- `mv` : déplace/renomme (fichier original supprimé)
- `cp` : copie (fichier original conservé)
- `mv` fonctionne sur fichiers ET dossiers sans option récursive

Manipulation de fichiers

- Cas d'usages avancé

```
# Renommer en lot avec boucle (aperçu)
for file in *.jpg; do mv "$file" "photo_$file"; done


# Déplacement conditionnel
mv -n fichier.txt backup/    # N'écrase pas si existe
mv -u source.txt dest.txt    # Seulement si source plus récente
```

Manipulation de fichiers

- Suppression de dossiers et de fichiers

```
rm fichier.txt                    # Supprime un fichier
rm fichier1.txt fichier2.txt      # Supprime plusieurs fichiers
rm *.tmp                          # Supprime tous les .tmp
rm -v fichier.txt                 # Mode verbose
rm -i fichier.txt                 # Demande confirmation
```

⚠️ **DANGER : `rm` est définitif !**

- Pas de corbeille en ligne de commande
- Suppression immédiate et irréversible

  Toujours vérifier avec `ls` avant `rm`

Manipulation de fichiers

- Suppression de dossiers et de fichiers

```
rmdir dossier_vide/ # Supprime dossier vide uniquement

rm -r dossier/ # Supprime dossier et contenu

rm -rf dossier/ # Force + récursif (TRÈS DANGEREUX)

rm -ri dossier/ # Récursif + interactif (plus sûr)

Options de sécurité

rm -i fichier.txt # Demande confirmation pour chaque fichier

rm -I *.txt # Demande confirmation pour plus de 3 fichiers

                alias rm='rm -i' # Protection permanente (dans .bashrc)
```

Manipulation de fichiers

- Affichage du contenu des fichiers

```
cat fichier.txt                    # Affiche tout le contenu
cat fichier1.txt fichier2.txt      # Concatène et affiche plusieurs fichiers
cat -n fichier.txt                 # Avec numéros de ligne
cat -A fichier.txt                 # Montre tous les caractères spéciaux
```

Cas d'usage de `cat` :

- Fichiers courts (quelques lignes/pages)
- Concaténation de fichiers
- Création rapide de fichiers (`cat > fichier.txt`)

Manipulation de fichiers

- Affichage de fichiers less & cat

- less fichier.txt *# Affichage paginé (recommandé)*
- more fichier.txt *# Affichage paginé (ancien)*

Navigation dans `less` :

- `Espace` ou `Page Down` : page suivante
- `b` ou `Page Up` : page précédente
- `↑ ↓` : ligne par ligne
- `/mot` : rechercher "mot"
- `n` : occurrence suivante de la recherche
- `N` : occurrence précédente
- `g` : aller au début
- `G` : aller à la fin
- `q` : quitter

Manipulation de fichiers

- Affichage de fichiers head & tail

```
head fichier.txt # 10 premières lignes

head -n 20 fichier.txt # 20 premières lignes

head -5 fichier.txt # 5 premières lignes

tail fichier.txt # 10 dernières lignes

tail -n 15 fichier.txt # 15 dernières lignes

tail -f /var/log/syslog # Suit les ajouts en temps réel
```

Manipulation de fichiers

- Affichage de fichiers head & tail

```
head -1 fichier.csv                # Voir les en-têtes CSV

tail -100 /var/log/apache2/error.log  # Dernières erreurs

tail -f /var/log/auth.log          # Surveillance en temps réel
```

Manipulation de fichiers

- Via un éditeur Nano(ou vim si vous voulez)

nano fichier.txt *# Ouvre/crée le fichier*

nano *# Ouvre sans nom (à sauver plus tard)*

nano +25 fichier.txt *# Ouvre à la ligne 25*

Interface de nano :

- Zone d'édition principale
- Barre de statut (ligne, colonne, nom fichier)
- Menu des raccourcis en bas (^ = Ctrl)

Manipulation de fichiers

- Via un éditeur Nano(ou vim si vous voulez)

Ctrl+O : Sauvegarder (Write Out)

Ctrl+X : Quitter

Ctrl+W : Rechercher (Where is)

Ctrl+\ : Rechercher et remplacer

Ctrl+K : Couper la ligne

Ctrl+U : Coller

Ctrl+G : Aide complète

ıer position du curseur

Liens symboliques et liens durs

- Liens symboliques

```
ln -s fichier_original.txt lien_symbolique.txt    # Crée un lien symbolique

ln -s /chemin/absolu/fichier.txt lien.txt         # Avec chemin absolu

ln -s ../parent/fichier.txt lien_relatif.txt      # Avec chemin relatif
```

Caractéristiques des liens symboliques :

- Pointent vers un autre fichier/dossier
- Si l'original est supprimé, le lien devient cassé
- Peuvent traverser les systèmes de fichiers
- Visibles avec `ls -l` (flèche →)

Liens symboliques et liens durs

- Liens durs

```
ln fichier_original.txt lien_dur.txt # Crée un lien dur
```

Caractéristiques des liens durs :

- Même inode que le fichier original
- Si l'original est supprimé, le contenu reste accessible
- Ne fonctionnent que sur le même système de fichiers
- Ne peuvent pas pointer vers des dossiers

Manipulation de fichiers

- TP1

 # 1. Créez un dossier "exercice" et allez dedans

 # 2. Créez 3 fichiers vides

 # 3. Écrivez du contenu dans file1.txt

 # 4. Copiez file1.txt vers file1_backup.txt

 # 5. Vérifiez le contenu

Manipulation de fichiers

- TP2

 # 1. Créez une arborescence avec les sous-dossiers projets, backups et src

 # 2. Créez des fichiers main.c, utils.c,header.c dans src/

 # 3. Ajoutez du contenu à main.c

 # 4. Sauvegardez tout dans backup/

 # 5. Vérifiez la structure

Manipulation de fichiers

- TP3

 # 1. Créez un fichier de configuration avec nano (Ajoutez quelques lignes de configuration)

 # 2. Créez un lien symbolique sur ce fichier

 # 3. Vérifiez que les deux fichiers ont le même contenu

 # 4. Modifiez via le lien symbolique

 # 5. Vérifiez que l'original a été modifié

Utilisateurs et groupes

- Concept fondamentaux

Le système multi-utilisateurs Linux

- Linux est conçu pour plusieurs utilisateurs simultanés
- Chaque utilisateur a un identifiant unique (UID)
- Chaque groupe a un identifiant unique (GID)
- Séparation stricte entre les utilisateurs pour la sécurité
-

Utilisateurs et groupes

- Concept fondamentaux

Types d'utilisateurs

Utilisateur root (UID 0) :

- Super-utilisateur avec tous les droits
- Peut tout faire sur le système
- Connexion directe déconseillée (utiliser sudo)

Utilisateurs et groupes

- Concept fondamentaux

Types d'utilisateurs

Utilisateurs système (UID 1-999) :

- Créés pour les services système
- Exemples : `www-data`, `mysql`, `sshd`, `daemon`
- Généralement sans shell de connexion

Utilisateurs normaux (UID ≥ 1000) :

- Comptes créés pour les personnes
- Ont un répertoire home (`/home/username`)
- Shell de connexion (bash, zsh, etc.)

Utilisateurs et groupes

- Gestion des utilisateurs

Consulter les infos utilisateurs(Taper les commandes en même temps pour dire ce que ça fait)

whoami *# Votre identité actuelle*

id

id alice

who

last

cat /etc/passwd

Utilisateurs et groupes

- Gestion des utilisateurs

 # Exemple de ligne :

alice:x:1001:1001:Alice Dupont,,,:/home/alice:/bin/bash

 # Format : nom:mdp:UID:GID:info:home:shell

Utilisateurs et groupes

- Gestion des utilisateurs

Créer des utilisateurs (Faites le en même temps)

```
sudo adduser bob # Interactive (Ubuntu/Debian)

sudo useradd -m -s /bin/bash carol # Manuel (toutes distributions)

# Exemple complet

sudo useradd -m -s /bin/bash -c "David Martin" -G sudo,developers david

sudo passwd bob # Changer le mot de passe de bob

passwd # Changer votre mot de passe
```

Utilisateurs et groupes

- Gestion des utilisateurs

Créer des utilisateurs (Faites le en même temps)

```
sudo usermod -G developers bob # Modifier un utilisateur existant

sudo usermod -s /bin/zsh alice # Ajouter aux développeurs

sudo deluser bob # Supprimer un utilisateur

sudo deluser --remove-home bob # Supprimer avec le répertoire home
```

Utilisateurs et groupes

- Gestion des groupes

Comprendre les groupes (4 min)

- **Groupe principal** : défini dans `/etc/passwd`, un seul par utilisateur
- **Groupes secondaires** : définis dans `/etc/group`, plusieurs possibles
- Les permissions de fichiers s'appliquent au groupe principal
- Utile pour organiser les utilisateurs par fonction/projet

Utilisateurs et groupes

- Gestion des groupes

Consulter les groupes

```
# Vos groupes

groups

groups alice # Groupes d'alice

cat /etc/group

# Groupes importants sur Ubuntu :

    # - sudo : peut utiliser sudo

    # - adm : peut lire les logs système

    # - www-data : serveur web
```

Utilisateurs et groupes

- Gestion des groupes

Structure du fichier `/etc/group`

 # Exemple de ligne :

developers:x:1002:alice,bob,carol

 # Format : nom:mdp:GID:membres

Utilisateurs et groupes

- Gestion des groupes

Créer et gérer les groupes

```
sudo addgroup developers # Ubuntu/Debian

sudo groupadd developers # Toutes distribution

sudo adduser alice developers # Ubuntu/Debian

sudo usermod -aG developers alice # Toutes distributions (-a = append)

sudo delgroup developers
```

Utilisateurs et groupes

- Bonnes pratiques et sécurité

Gestion des mots de passe

 # Politique de mots de passe

sudo cat /etc/login.defs | grep PASS

sudo chage -l alice *# Voir politique pour alice*

sudo passwd -l alice *# Verrouiller le compte*

sudo passwd -u alice *# Déverrouiller le compte*

 # Mots de passe forts # -

Au moins 8 caractères

Mélange majuscules/minuscules/chiffres/symboles

- Pas d'informations personnelles

Utilisateurs et groupes

- Bonnes pratiques et sécurité

Principe du moindre privilège

- **N'ajoutez au groupe sudo que si nécessaire**
- **Créez des groupes spécifiques par fonction**
- **Révisez régulièrement les appartenances aux groupes**
- **Supprimez les comptes inutilisés**

Utilisateurs et groupes

- Bonnes pratiques et sécurité

Audit et surveillance

```
# Qui a les droits sudo ? grep sudo /etc/group

# Dernières connexions last | head -10

# Comptes sans mot de passe (DANGEREUX) sudo awk -F: '($2 == "") {print $1}' /etc/shadow
```

Utilisateurs et groupes

- TP1 (car gros sujets a maitriser)

 # 1. Affichez votre identité complète

 # 2. Trouvez combien d'utilisateurs sont définis sur le système

 # 3. Listez les 5 premiers utilisateurs système

 # 4. Trouvez votre ligne dans /etc/passwd

 # 5. Vérifiez qui est connecté actuellement

Utilisateurs et groupes

- TP2 (car gros sujets a maitriser)

 # 1. Créez un utilisateur test

 # 2. Vérifiez qu'il a été créé

 # 3. Changez son mot de passe par “votrenom-annédenaissancedevotrevoisin”

 # 4. Testez la connexion (nouvelle session ou terminal)

5. Supprimez l'utilisateur test

Utilisateurs et groupes

- **TP3 (car gros sujets a maitriser)**

 # 1. Créez un groupe "projet"

 # 2. Ajoutez-vous au groupe projet

 # 3. Vérifiez (nécessite reconnexion pour être effectif)

 # 4. Créez un dossier de travail partagé

 # 5. Testez l'accès (après reconnexion)

 # 6. Nettoyage du groupe et du projet

Utilisateurs et groupes

- TP4 (car gros sujets a maitriser)

 # 1. Créez un user michael

- *avec un home directory /opt/michael*
- *des droits sudo*

Permissions et propriétés

- Comprendre le système de permissions

Philosophie des permissions Unix

- **Multi-utilisateurs** : plusieurs personnes peuvent utiliser le même système
- **Sécurité par défaut** : accès restreint, permissions explicites
- **Granularité** : contrôle précis sur qui peut faire quoi
- **Héritage** : les fichiers héritent des permissions du répertoire parent

Permissions et propriétés

- Comprendre le système de permissions

Les trois types d'utilisateurs (8 min)

ls -l fichier.txt -rw-r--r-- 1 utilisateur groupe 1234 déc 15 10:30 fichier.txt

Propriétaire (User/Owner) : créateur du fichier

Groupe (Group) : groupe auquel appartient le fichier

Autres (Others/World) : tous les autres utilisateurs du système

Permissions et propriétés

- Comprendre le système de permissions

Les trois types de permissions pour un fichier

`r` **(Read - 4)** : lire le contenu du fichier

`w` **(Write - 2)** : modifier le contenu du fichier

`x` **(Execute - 1)** : exécuter le fichier comme programme

Permissions et propriétés

- Comprendre le système de permissions

Les trois types de permissions pour un répertoire

`r` **(Read - 4)** : lister le contenu du répertoire

`w` **(Write - 2)** : créer/supprimer des fichiers dans le répertoire

`x` **(Execute - 1)** : traverser le répertoire (cd dedans)

Permissions et propriétés

- Lecture des permissions

Anatomie complète d'une ligne `ls -l`

```
drwxr-xr-x 2 utilisateur groupe 4096 déc 15 10:30 Documents
-rw-r--r-- 1 utilisateur groupe 1234 déc 15 09:15 fichier.txt
lrwxrwxrwx 1 utilisateur groupe   12 déc 15 11:00 lien -> fichier.txt
```

Permissions et propriétés

d rwx r-x r-x

| | | |

| | | └── Permissions pour "others"

| | └──────── Permissions pour "group"

| └──────────── Permissions pour "user" (propriétaire)

└──────────────── Type de fichier

Types de fichiers (premier caractère) :

- `-` : fichier régulier
- `d` : répertoire (directory)
- `l` : lien symbolique (link)
- `c` : périphérique caractère
- `b` : périphérique bloc
- `p` : pipe nommé
- `s` : socket

Permissions et propriétés

- Lecture des permissions

Faites le en même temps

```
# Créons des exemples

touch exemple.txt

mkdir exemple_dir

chmod 644 exemple.txt

chmod 755 exemple_dir

ls -l
```

Permissions et propriétés

- Modifications des permissions

CHMOD

Notation octale (Je vous l'ai mis pour info mais je ne sais pas l'expliquer)

Conversion binaire → octale :

rwx = 111_2 = 7_8

r-x = 101_2 = 5_8

r-- = 100_2 = 4_8

-wx = 011_2 = 3_8

-w- = 010_2 = 2_8

Permissions et propriétés

- Modifications des permissions

`CHMOD`

Valeurs courantes :

- **644** : `-rw-r--r--` (fichier texte standard)
- **755** : `-rwxr-xr-x` (script exécutable)
- **600** : `-rw-------` (fichier privé)
- **700** : `-rwx------` (dossier privé)
- **666** : `-rw-rw-rw-` (fichier accessible à tous)
- **777** : `-rwxrwxrwx` (tout accessible - DANGEREUX)

Permissions et propriétés

- Gestion des propriétaires

Commande chown

Syntaxe : `chown [utilisateur]:[groupe] fichier`

```
ls -l fichier.txt # Voir la propriété actuelle

chown alice fichier.txt # Changer le propriétaire

chown alice:developers fichier.txt # Changer le propriétaire et le groupe

chown -R alice:developers projet/  # Récursif sur un dossier
```

Permissions et propriétés

- Gestion des propriétaires

Commande `chgrp`

Syntaxe : `chown [utilisateur]:[groupe] fichier`

bash

 # Changer seulement le groupe

chgrp developers fichier.txt

chgrp -R admin projet/

 # Équivalent avec chown

chown :developers fichier.txt

Permissions et propriétés

- Vérification des utilisateurs et groupes

```
id # Votre identité

id alice # Identité d'alice

groups # Vos groupes

groups alice # Groupes d'alice

cat /etc/passwd | head -5 # Liste des utilisateurs

cat /etc/group | head -5 # Liste des groupes
```

Permissions et propriétés

- Sudo : Élévation de privilèges

Concept de sudo

- **su** : Switch User (changer d'utilisateur)
- **sudo** : Super User Do (faire en tant que super-utilisateur)
- Permet d'exécuter des commandes avec les droits root
- Plus sûr que se connecter directement en root

Permissions et propriétés

- Sudo : Élévation de privilèges

Utilisation de base

```
# Commandes nécessitant sudo

sudo apt update # Gestion des paquets

sudo mkdir /etc/newdir # Écriture dans /etc

sudo chown root:root file # Changer propriétaire

sudo systemctl restart ssh # Gestion des services

sudo -i # Devenir temporairement root
```

Permissions et propriétés

- Sudo : Élévation de privilèges

Configuration sudo

```
# Fichier de configuration (À NE PAS MODIFIER DIRECTEMENT)

sudo cat /etc/sudoers

# Édition sécurisée

sudo visudo

# Voir qui peut utiliser sudo

sudo cat /etc/group | grep sudo
```

Permissions et propriétés

- TP1

 # 1. Créez un repertoire avec 2 fichiers et 2 sous-repertoires

 # 2. Analysez les permissions par défaut de ces fichiers et repertoires

 # 3. Identifiez pour chaque élément : # - Type (fichier/dossier) # - Permissions du propriétaire, groupe, autres # - Propriétaire et groupe #

Questions :

 # - Pouvez-vous lire les fichiers ?

 # - Que signifie la permission x sur un dossier ?

Permissions et propriétés

- TP2

```
# 1. Créez un script

#!/bin/bash

echo "Hello, je suis un script !"

date

# 2. Analysez les permissions par défaut de ces fichiers et repertoires

# 3. Identifiez pour chaque élément : # - Type (fichier/dossier) # - Permissions du propriétaire, groupe, autres # - Propriétaire et groupe
```

Permissions et propriétés

- TP2

```
# 1. Créez un script

#!/bin/bash

echo "Hello, je suis un script !"

date

# 2. Essayez de l'exécuter

# 3. Ajoutez la permission d'exécution

# 4. Exécutez maintenant
```

Permissions et propriétés

- TP3

 # 1. Créez un fichier et vérifiez sa propriété

 # 2. changer le propriétaire

 # 3. Créer un fichier avec sudo

 # 4. Modifier ce fichier sans utiliser sudo

 # 5. Trouver un moyen de pouvoir modifier ce fichier

Mount sous Linux

- Concept de base

Qu'est-ce que le montage

- **Définition :** Processus d'attachement d'un système de fichiers à l'arborescence Linux
- **Point de montage :** Répertoire où le système de fichiers devient accessible
- **Principe Unix :** "Tout est fichier" - les périphériques apparaissent comme des fichiers
- **Arborescence unique :** Contrairement à Windows, Linux n'a qu'une seule hiérarchie commençant par `/`

Mount sous Linux

- Concept de base

TERMINOLOGIE

Système de fichiers : Structure logique d'organisation des données (ext4, xfs, ntfs...)

Partition : Division logique d'un disque dur

Device : Représentation d'un périphérique dans `/dev/`

UUID : Identifiant unique universel d'une partition

Label : Nom personnalisé donné à un système de fichiers

Mount sous Linux

- Concept de base

```
# Lister tous les périphériques de stockage
lsblk

# Affichage en arbre avec informations détaillées
lsblk -f

# Informations sur les partitions
fdisk -l

# Périphériques dans /dev
ls -la /dev/sd*     # Disques SATA/SCSI
ls -la /dev/nvme*   # Disques NVMe
ls -la /dev/mmcblk* # Cartes SD/eMMC
```

Mount sous Linux

- Systéme de fichier

Systèmes de fichiers Linux natifs :

- **ext4** : Standard Linux, journalisé, performant
- **xfs** : Haute performance, gros volumes
- **btrfs** : Moderne, snapshots, compression
- **zfs** : Avancé, intégrité des données

Systèmes de fichiers externes :

- **ntfs** : Windows NT/2000/XP/Vista/7/8/10
- **exfat** : Compatible Windows/Mac, gros fichiers
- **fat32** : Compatible universel, limité à 4GB par fichier
- **hfs+** : Mac OS

Mount sous Linux

- Systéme de fichier

Vérification du système de fichier

 # Méthode 1

file -s /dev/sda1

 # Méthode 2

blkid /dev/sda1

 # Méthode 3

lsblk -f

Mount sous Linux

- La commande mount

```
mount [options] <périphérique> <point_de_montage>

# Montage simple d'une partition

sudo mount /dev/sda2 /mnt

# Montage avec spécification du type

sudo mount -t ext4 /dev/sda2 /mnt

# Montage en lecture seule

sudo mount -o ro /dev/sda2 /mnt

# Montage d'une clé USB (auto-détection du type)

sudo mount /dev/sdb1 /media/usb
```

Mount sous Linux

- La commande mount

```
# Afficher tous les montages

mount

# Montages actifs avec df

df -h

# Montages de répertoires

mount -o bind /data1 /data2

# Montages d'un périphérique spécifique

mount | grep sda2
```

Mount sous Linux

- La commande umount

```
# Démontage par point de montage

sudo umount /mnt

# Démontage par périphérique

sudo umount /dev/sda2

# Démontage forcé (attention!)

sudo umount -f /mnt

# Démontage paresseux (détache immédiatement)

sudo umount -l /mnt

# Vérifier qu'aucun processus n'utilise le montage

lsof /mnt

fuser -m /mnt
```

Mount sous Linux

- Montage permanent

bash

```
# <file system>                          <mount point> <type> <options>                <dump> <pass>
UUID=550e8400-e29b-41d4-a716-446655440000 /           ext4   defaults                 0      1
UUID=f8b8c1a4-8b4f-4c7d-9e2a-1f3b5c7d9e2a /home       ext4   defaults,noatime         0      2
UUID=1234-5678                           /boot/efi    vfat   defaults,umask=0077      0      0
/dev/cdrom                               /media/cdrom iso9660 noauto,ro,user           0      0
   //serveur/partage                     /mnt/samba   cifs   username=user,
```

Chroot sous Linux

- Introduction et concept

Qu'est-ce que chroot ?

- **Définition :** CHange ROOT - Change la racine du système de fichiers
- **Principe :** Créer un environnement isolé avec une nouvelle racine `/`
- **Historique :** Introduit en 1982 dans Unix Version 7
- **Objectif :** Isolation des processus dans un sous-ensemble du système de fichiers

Chroot sous Linux

- Introduction et concepts

Cas d'usage principaux

- **Récupération système :** Réparer un système endommagé
- **Environnement de développement :** Tester dans un environnement contrôlé
- **Sécurité :** Isoler des services (sandbox basique)
- **Migration :** Préparer un système pour un nouveau matériel
- **Serveurs FTP :** Limiter l'accès utilisateur à un répertoire
- **Compilation :** Builder des paquets dans un environnement propre

Chroot sous Linux

- Introduction et concepts

Limites et alternatives

Limites de chroot :

- Pas d'isolation des ressources (CPU, mémoire, réseau)
- Peut être contourné par un utilisateur root
- Partage du même noyau que l'hôte

Alternatives modernes :

- **Conteneurs :** Docker, Podman, LXC
- **Namespaces :** Isolation plus poussée
- **Machines virtuelles :** Isolation complète

Chroot sous Linux

- Introduction et concepts

```
# Créer le répertoire racine du chroot

sudo mkdir -p /opt/mychroot/{bin,lib,lib64,usr/bin,usr/lib,etc,dev,proc,sys,tmp}

# Définir les permissions appropriées

sudo chmod 755 /opt/mychroot

sudo chmod 777 /opt/mychroot/tmp

# Monter /proc (informations processus)

sudo mount -t proc proc /opt/mychroot/proc
```

Chroot sous Linux

- Introduction et concepts

```
# Monter /sys (interface noyau)

sudo mount -t sysfs sys /opt/mychroot/sys

# Monter /dev (périphériques)

sudo mount --bind /dev /opt/mychroot/dev

# Monter /dev/pts (terminaux pseudo)

sudo mount --bind /dev/pts /opt/mychroot/dev/pts

# Vérifier les montages

mount | grep mychroot
```

Chroot sous Linux

- Introduction et concepts

```
# Chroot basique

sudo chroot /opt/mychroot

# Vérifier l'environnement

pwd          # Devrait afficher /

ls           # Voir la structure limitée

echo $0      # Vérifier le shell

exit         # Sortir du chroot
```

Chroot sous Linux

- TP

 # 1 Créer un chroot dans le repertoire /mnt

- *tester quelques commandes comme ls , touch , id*
- *créer un utilisateur userchroot dans ce chroot , attention aidez vous de l'IA car il vous faudra faire des*

montages supplémenataires

- *Est-ce que cet utilisateur est disponible en dehors de votre chroot ? pourquoi ?*

Gestion du système et outils avancés

Gestion des processus
- Comprendre les processus

Qu'est-ce qu'un processus ?

- **Programme en cours d'exécution** en mémoire
- **Instance** d'un programme (un programme peut avoir plusieurs processus)
- **Identifiant unique** : PID (Process ID)
- **Hiérarchie** : processus parent et enfants
- **Ressources** : mémoire, CPU, fichiers ouverts

Gestion des processus
- Comprendre les processus

États d'un processus

- **R (Running)** : en cours d'exécution ou prêt
- **S (Sleeping)** : en attente (interruptible)
- **D (Uninterruptible sleep)** : en attente (non-interruptible, I/O)
- **Z (Zombie)** : terminé mais pas encore nettoyé par le parent
- **T (Stopped)** : arrêté (Ctrl+Z ou signal STOP)
- **< (High priority)** : priorité élevée
- **N (Low priority)** : priorité basse
-

Gestion des processus
- Comprendre les processus

Hiérachie d'un processus

pstree *# Arbre des processus*

pstree -p *# Avec les PID*

pstree $USER *# Seulement vos processus*

 # Exemple typique :

```
systemd(1)———bash(1234)———nano(1456)
                ├─firefox(2345)———{firefox}(2346)
                └─gnome-terminal(3456)———bash(3457)
```

Gestion des processus
- Comprendre les processus

La commande ps

```
ps # Processus du terminal courant

ps -e # Tous les processus (-e = every)

ps -f # Format complet (-f = full)

ps -ef # Tous les processus, format complet
```

Gestion des processus
- Comprendre les processus

La commande ps options importantes

```
ps aux # TRÈS UTILISÉ : all users, format user

ps axu # Équivalent

ps aux | head -10 # 10 premiers processus

ps aux --sort=-%cpu # Trié par CPU décroissant

ps aux --sort=-%mem # Trié par mémoire décroissant

ps aux | grep firefox # Processus contenant "firefox"
```

Gestion des processus
- Surveillance en temps réel

top *# Surveillance temps réel (tapez la commande)*

 # Interface top :

 # Ligne 1 : uptime, utilisateurs, load average

 # Ligne 2 : nombre de processus et leurs états

 # Ligne 3 : utilisation CPU (us=user, sy=system, id=idle)

 # Ligne 4-5 : utilisation mémoire (Mem) et swap # Table : processus triés par CPU

Gestion des processus
- Surveillance en temps réel

 # load average: 0.15, 0.25, 0.30

 # 3 valeurs : 1min, 5min, 15min

 # Sur un système mono-core :

 # 0.00 = pas de charge

 # 0.50 = 50% de charge

 # 1.00 = 100% de charge (saturé)

 # Sur un système quad-core :

 # 4.00 = 100% de charge (tous les cores utilisés)

Gestion des processus
- Surveillance en temps réel

 # Installation si nécessaire (utilisez cette commande en même temps)

```
sudo apt install htop
```

```
htop # Version améliorée de top
```

Gestion des processus
- Signaux et kill

 # Les signaux sont des messages envoyés aux processus

 # Numéros et noms standards :

 # Signaux courants : kill -l

 # 1 HUP : Hangup (rechargement config)

 # 2 INT : Interrupt (Ctrl+C)

 # 3 QUIT : Quit (Ctrl+\)

 # 9 KILL : Kill (terminaison forcée)

 # 15 TERM : Terminate (terminaison propre, par défaut)

 # 18 CONT : Continue (reprendre processus)

Gestion des processus - Signaux et kill

Commande `kill`

```
# Syntaxe : kill [signal] PID

# Terminaison normale (signal 15 TERM)

kill 1234 # Signal TERM par défaut

# Terminaison forcée (signal 9 KILL)

kill -9 1234 # Force la terminaison

pgrep firefox # Trouve PID de firefox

pkill firefox # Tue tous les processus firefox

pkill -f "firefox" # Tue par ligne de commande complète

killall firefox # Tue tous les firefox (par nom exact)
```

Gestion des processus
- Processus en arrière plan

```
# Lancer directement en arrière-plan avec &

sleep 60 & # Process en arrière-plan

echo $! # PID du dernier processus background

# Voir les jobs en cours

jobs # Liste des jobs du shell courant

jobs -l # Avec les PID
```

Gestion des processus
- Processus en arrière plan

Suspendre et reprendre des processus

```
# Suspendre un processus en cours

# Ctrl+Z # Suspend le processus courant

# Exemple pratique :

nano fichier.txt # Ouvre nano

# Ctrl+Z # Suspend nano

# [1]+ Stopped nano fichier.txt

jobs # Voir les jobs suspendus

fg # Reprendre au premier plan (foreground)

bg # Continuer en arrière-plan (background)
```

Gestion des processus
- Processus en arrière plan

Commande nohup

```
# nohup = no hangup

# Processus qui continue même après fermeture du terminal

nohup ls & # Processus persistant

Redirection personnalisée

nohup command > output.log 2>&1 &

# Exemple pratique : sauvegarde longue

nohup tar -czf backup.tar.gz ~/Documents/ &
```

Gestion des processus
- Processus système et services

Processus système importants

```
ps aux | grep -E "(systemd|init)" # Processus init
ps aux | grep -E "(kernel|kthread)" # Processus noyau [entre crochets]
ps aux | grep -E "(sshd|NetworkManager)" # Processus réseau
```

Gestion des processus
- Processus système et services

Introduction à systemctl

systemctl list-units --type=service --state=active

systemctl status ssh

systemctl status NetworkManager

 # Gestion des services (nécessite sudo)

sudo systemctl stop ssh *# Arrêter*

sudo systemctl start ssh *# Démarrer*

sudo systemctl restart ssh *# Redémarrer*

sudo systemctl reload ssh *# Recharger config*

Gestion des processus
- Processus système et services

Introduction à systemctl

```
# Services au démarrage

systemctl is-enabled ssh # Vérifie si activé au boot

sudo systemctl enable ssh # Active au boot

sudo systemctl disable ssh # Désactive au boot
```

Gestion des processus - TP1

 # 1. Affichez tous vos processus

 # 2. Trouvez le processus le plus gourmand en CPU

 # 3. Trouvez le processus le plus gourmand en mémoire

 # 4. Comptez vos processus actifs

 # 5. Affichez l'arbre de vos processus

Gestion des processus - TP2

 # 1. Créez un processus long en arrière-plan

 # 2. Vérifiez qu'il fonctionne

 # 3. Suspendez-le

 # 4. Reprenez-le

 # 5. Terminez-le proprement

 # 6. Vérifiez qu'il est terminé

Gestion des paquets
- Comprendre les gestionnaires de paquets

Qu'est-ce qu'un gestionnaire de paquets ?

- **Outil central** pour installer, mettre à jour et supprimer des logiciels
- **Gestion des dépendances** : résolution automatique des bibliothèques requises
- **Intégrité** : vérification des signatures et checksums
- **Base de données** : suivi des paquets installés
- **Repositories** : sources centralisées de logiciels

Gestion des paquets
- Comprendre les gestionnaires de paquets

Gestionnaire de paquets (solution) :

 # - Installation en une commande

 # - Résolution automatique des dépendances

 # - Mises à jour centralisées

 # - Désinstallation propre

 # - Sécurité et intégrité vérifiées

Installation manuelle (problématique) :

 # - Téléchargement manuel des sources

 # - Compilation longue et complexe

 # - Gestion manuelle des dépendances

 # - Pas de désinstallation propre

 # - Conflits de versions

Gestion des paquets
- Comprendre les gestionnaires de paquets

Gestionnaires par distribution

Distribution	Famille	Gestionnaire	Format	Exemples
Ubuntu/Debian	Debian	APT	.deb	`apt install package`
CentOS/RHEL/Fedora	Red Hat	YUM/DNF	.rpm	`dnf install package`
openSUSE	SUSE	Zypper	.rpm	`zypper install package`
Arch Linux	Arch	Pacman	.pkg.tar.xz	`pacman -S package`
Alpine Linux	Alpine	APK	.apk	`apk add package`

Gestion des paquets

- Comprendre les gestionnaires de paquets

 # Composants APT :

 # - apt : interface moderne (Ubuntu 16.04+)

 # - apt-get : interface traditionnelle

 # - apt-cache : recherche et informations

 # - aptitude : interface avancée

 # - dpkg : gestionnaire de base (.deb)

Gestion des paquets
- Comprendre les gestionnaires de paquets

Mise à jour des sources et du système (8 min)

```
sudo apt update# a faire systématiquement avant d'installer un package

apt list --upgradable # List les paquets a upgrader

apt list --upgradable | wc -l

sudo apt upgrade # Mise à jour sans suppression

sudo apt full-upgrade # Mise à jour avec résolution avancée
```

Gestion des paquets
- Installation de paquets

```
sudo apt install package_name

sudo apt install vim # Éditeur vim

sudo apt install curl wget # Plusieurs paquets

sudo apt install -y htop # -y = yes à toutes les questions
```

Gestion des paquets
- Installation de paquets

```
# Installation d'une version spécifique

apt list -a package_name # Voir versions disponibles

sudo apt install package_name=version

# Installation depuis un fichier .deb local

sudo apt install ./package.deb # Résout les dépendances

sudo dpkg -i package.deb # Installation directe (sans dépendances)
```

Gestion des paquets
- Recherche de paquets

```
# Recherche par nom

apt search keyword

apt search "text editor"

apt search firefox | head -10

# Recherche exacte

apt list firefox

apt list firefox* # Avec wildcards

# Information sur un paquet

apt show package_name

apt show firefox
```

Gestion des paquets
- Suppression de paquets

```
# Suppression simple (garde les fichiers de config)

sudo apt remove package_name

# Suppression complète (avec fichiers de config)

sudo apt purge package_name

sudo apt remove --purge package_name # Équivalent

# Suppression avec dépendances orphelines

sudo apt autoremove package_name

# Suppression des dépendances non utilisées

sudo apt autoremove
```

Gestion des paquets
- Suppression de paquets

```
# Exemple pratique

sudo apt install gimp # Installation

sudo apt remove gimp # Suppression simple

dpkg -l | grep gimp # Vérifier (rc = removed, config kept)

sudo apt purge gimp # Purge complète

sudo apt autoremove # Nettoyer dépendances
```

Gestion des paquets
- Nettoyage du systéme

```
# Nettoyer le cache des paquets téléchargés

sudo apt clean # Supprime tout le cache (/var/cache/apt/archives/)

sudo apt autoclean # Supprime seulement versions obsolètes

# Voir l'espace utilisé par le cache

du -sh /var/cache/apt/archives/
```

Gestion des paquets
- Nettoyage du systéme

```
# Lister les paquets installés manuellement

apt-mark showmanual | head -10

# Lister les paquets installés automatiquement (dépendances)

apt-mark showauto | head -10

# Marquer un paquet comme installé manuellement

sudo apt-mark manual package_name

# Marquer comme automatique (sera supprimé par autoremove si orphelin)

sudo apt-mark auto package_name
```

Gestion des paquets
- Configuration des sources apt

```
# Fichier principal

cat /etc/apt/sources.list

# Fichiers additionnels

ls /etc/apt/sources.list.d/
```

```
# Format d'une ligne source :
# deb http://archive.ubuntu.com/ubuntu/ jammy main restricted
# |   |                                |     |
# |   |                                |     └── Composants
# |   |                                └──────── Version (codename)
# |   └─────────────────────────────────────────── URL du repository
# └─────────────────────────────────────────────── Type (deb ou deb-src)

# Composants Ubuntu :
# main       : logiciels supportés officiellement
# restricted : pilotes propriétaires supportés
# universe   : logiciels communautaires
# multiverse : logiciels avec restrictions légales
```

Gestion des paquets
- Clés GPG

```
bash

# Lister les clés installées

sudo apt-key list

# Ajouter une clé (méthode moderne)

curl -fsSL https://example.com/key.gpg | sudo gpg --dearmor -o /etc/apt/keyrings/example.gpg

# Ajouter un repository tiers (exemple : VS Code)

curl -fsSL https://packages.microsoft.com/keys/microsoft.asc | sudo gpg --dearmor -o /etc/apt/keyrings/packages.microsoft.gpg

echo "deb [arch=amd64,arm64,armhf signed-by=/etc/apt/keyrings/packages.microsoft.gpg] https://packages.microsoft.com/rep
stable main" | sudo tee /etc/apt/sources.list.d/vscode.list

sudo apt update

sudo apt install code
```

Gestion des paquets
- Problèmes courants

```
# 1. Paquets cassés
sudo apt --fix-broken install    # Réparer installations cassées
sudo dpkg --configure -a         # Configurer paquets non configurés
# 2. Cache corrompu
sudo apt clean
sudo apt update
# 3. Sources inaccessibles
sudo apt update 2>&1 | grep -i "failed\|error"
# 4. Espace disque insuffisant
df -h                            # Vérifier espace
sudo apt clean                   # Nettoyer cache
```

Gestion des paquets
- Problèmes courants

```
# 4. Espace disque insuffisant
df -h                    # Vérifier espace
sudo apt clean           # Nettoyer cache
sudo apt autoremove      # Supprimer orphelins

# 5. Verrous APT
sudo rm /var/lib/apt/lists/lock
sudo rm /var/cache/apt/archives/lock
sudo rm /var/lib/dpkg/lock*
sudo dpkg --configure -a
```

Gestion des paquets
- Logs et diagnostiques

```
# Logs APT
sudo tail -50 /var/log/apt/history.log    # Historique installations
sudo tail -50 /var/log/apt/term.log       # Sorties détaillées
sudo tail -50 /var/log/dpkg.log           # Logs dpkg
# Vérifier intégrité des paquets
sudo debsums -c                           # Vérifier checksums
sudo apt check                            # Vérifier cohérence
# Informations système
apt policy                                # Priorités des sources
apt-cache policy package_name             # Politique d'un paquet
```

Gestion des paquets - Bonnes pratiques

```
# 1. Toujours mettre à jour avant d'installer

sudo apt update && sudo apt install package

# 2. Mise à jour régulière du système

sudo apt update && sudo apt upgrade

# 3. Nettoyer régulièrement

sudo apt autoremove && sudo apt autoclean

# 4. Éviter les sources non fiables

# - Vérifier les clés GPG

# - Utiliser HTTPS quand possible

# - Préférer les repos officiels
```

Gestion des paquets - TP1

 # 1. Mettre à jour la liste des paquets

 # 2. Voir combien de mises à jour sont disponibles

 # 3. Installer l'outil de monitoring htop

 # 4. Vérifier l'installation

 # 5. Chercher des éditeurs de texte

 # 6. Obtenir des infos sur vim

Gestion des paquets - TP2

 # 1. Installer un paquet de test

 # 2. Tester le paquet

 # 3. Voir les dépendances installées

 # 4. Supprimer le paquet

 # 5. Vérifier qu'il est supprimé

 # 6. Nettoyer les dépendances orphelines

 # 7. Nettoyer le cache

Gestion des paquets - TP3

 # 1. Chercher tous les paquets liés à Python

 # 2. Lister seulement les paquets Python installés

 # 3. Trouver quel paquet fournit la commande 'tree'

 # 4. Installer tree et tester

 # 5. Voir l'espace utilisé par le cache APT

 # 6. Lister les 10 derniers paquets installés

 # 7. Installer un paquet dans une version spécifique

 # 8. Marquer cette version afin qu'elle ne soit plus mise a jour

SSH

- Introduction & concepts

Qu'est-ce que SSH ?

- **Définition :** Secure Shell, protocole de communication sécurisé
- **Port par défaut :** 22
- **But principal :** Accès distant sécurisé aux systèmes Unix/Linux
- **Année de création :** 1995 par Tatu Ylönen

SSH

- Introduction & concepts

Pourquoi utiliser SSH ?

- **Chiffrement :** Toutes les communications sont chiffrées
- **Authentification :** Plusieurs méthodes disponibles (mot de passe, clés)
- **Intégrité :** Vérification que les données n'ont pas été modifiées
- **Polyvalence :** Shell, transfert de fichiers, tunneling

SSH

- Introduction & concepts

Tunneling et encapsulation de protocole

Le protocole SSH peut permettre l'encapsulation de n'importe quel protocole. Ainsi, un protocole non chiffré comme l'HTTP peut être encapsulé (comprendre "emballé", comme un objet dans un colis) dans un le protocole SSH pour passer un pare-feu ou un proxy par exemple. Il sera ensuite désencapsulé à la destination ("déballé"). On peut également entendre la description de "*Proxy SSH*" lorsque l'on parle de ce procédé de *tunneling SSH*.

SSH

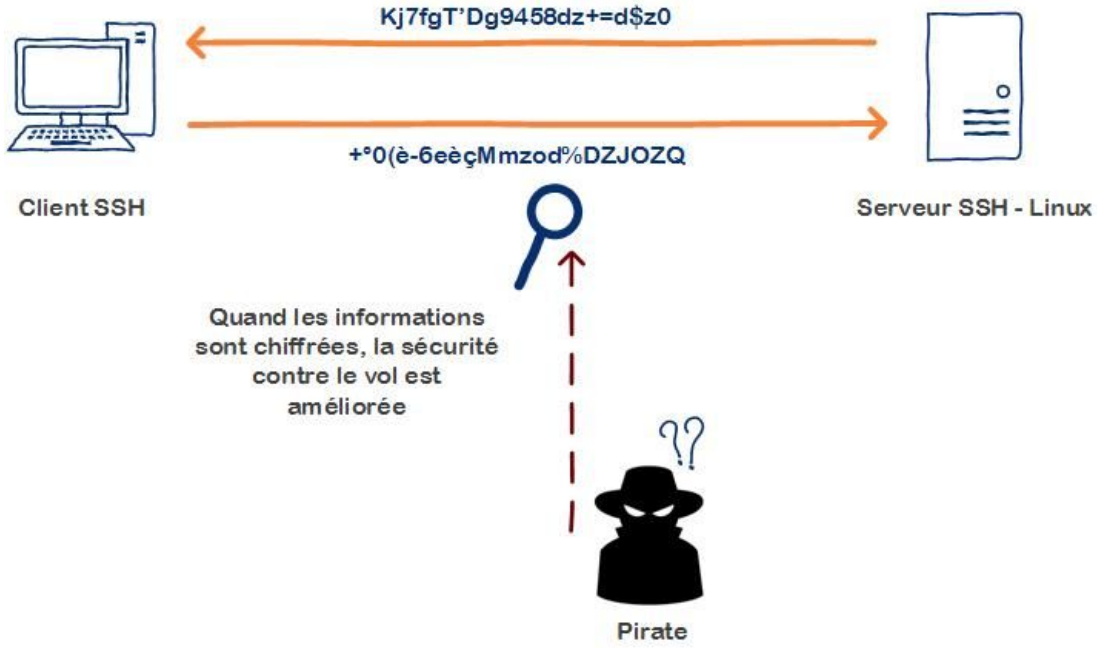
- Introduction & concepts

Architecture client-serveur

- **Client SSH :** Initiateur de la connexion (ssh, PuTTY, etc.)
- **Serveur SSH :** Daemon qui écoute les connexions (sshd)
- **Processus de connexion :**
    1. Négociation de la version du protocole
    2. Échange des clés de chiffrement
    3. Authentification
    4. Ouverture du canal sécurisé

SSH

- Introduction & concepts



SSH

- Installation

```
sudo apt update

sudo apt install openssh-server

sudo systemctl start ssh

sudo systemctl enable ssh

sudo systemctl status ssh
```

SSH

- Configuration

Fichier de configuration principal : `/etc/ssh/sshd_config`

Paramètres essentiels à connaître :

```
# Port d'écoute (défaut: 22)

Port 22

# Adresses d'écoute

ListenAddress 0.0.0.0

# Authentification par mot de passe

PasswordAuthentication yes
```

SSH

- Configuration

```
Authentication yes

# Authentification par clé publique

PubkeyAuthentication yes

# Connexion root

PermitRootLogin no

# Nombre max de tentatives

  MaxAuthTries 3
```

SSH

- Connection SSH

Syntaxe de base

ssh utilisateur@adresse_ip

ssh utilisateur@nom_domaine

ssh -p port utilisateur@serveur

SSH

- Connection SSH

Syntaxe de base

ssh utilisateur@adresse_ip

ssh utilisateur@nom_domaine

ssh -p port utilisateur@serveur

SSH

- Connection SSH

Exemples pratiques

```
# Connexion basique

ssh admin@192.168.1.100

# Connexion avec port spécifique

ssh -p 2222 admin@monserveur.com

# Exécution d'une commande distante

ssh admin@serveur "ls -la /var/log"

# Connexion avec verbosité pour debug

ssh -v admin@serveur
```

SSH

- Authentification par clés SSH

Principe des clés asymétriques

- **Clé privée :** Gardée secrète sur le client
- **Clé publique :** Installée sur le serveur
- **Avantages :** Plus sécurisé, pas de mot de passe à retaper

SSH

- Authentification par clés SSH

```
# Génération d'une paire de clés RSA

ssh-keygen -t rsa -b 4096 -C "mon.email@exemple.com"

# Génération avec algorithme ED25519 (recommandé)

ssh-keygen -t ed25519 -C "mon.email@exemple.com"

# Spécifier l'emplacement

ssh-keygen -t ed25519 -f ~/.ssh/ma_cle_speciale
```

SSH

- Authentification par clés SSH

```
# Méthode automatique (recommandée)

ssh-copy-id utilisateur@serveur

# Méthode manuelle

cat ~/.ssh/id_rsa.pub | ssh utilisateur@serveur "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"

# Vérification des permissions

chmod 700 ~/.ssh

chmod 600 ~/.ssh/authorized_keys
```

SSH

- Authentification par clés SSH

```
# Test de la connexion

ssh -i ~/.ssh/ma_cle_privee utilisateur@serveur

# Agent SSH pour gérer les clés

ssh-add ~/.ssh/id_rsa

ssh-add -l  # Lister les clés chargées
```

SSH

- Sécurisation du serveur SSH

```
# Changer le port par défaut
Port 2222
# Interdire la connexion root
PermitRootLogin no
# Désactiver l'authentification par mot de passe
PasswordAuthentication no
# Limiter les utilisateurs autorisés
AllowUsers admin user1 user2
# Limiter les groupes
AllowGroups sshusers
```

SSH

- Sécurisation du serveur SSH

```
# Timeout de connexion
ClientAliveInterval 300
ClientAliveCountMax 2
# Désactiver les forwards X11 si non nécessaire
X11Forwarding no
# Protocole version 2 uniquement
Protocol 2
```

SSH

- Fonctionnalités avancées::SCP

```
# Copier un fichier local vers le serveur

scp fichier.txt utilisateur@serveur:/chemin/destination/

# Copier un dossier récursivement

scp -r dossier/ utilisateur@serveur:/chemin/destination/

# Copier depuis le serveur vers local

scp utilisateur@serveur:/chemin/fichier.txt ./
```

SSH

- Fonctionnalités avancées::SFTP

```
# Session SFTP interactive
sftp utilisateur@serveur
# Commandes SFTP courantes
get fichier_distant.txt       # Télécharger
put fichier_local.txt         # Uploader
ls                            # Lister fichiers distants
lls                           # Lister fichiers locaux
cd /chemin                    # Changer répertoire distant
lcd /chemin                   # Changer répertoire local
```

SSH

- Fonctionnalités avancées::Tunneling

```
# Rediriger le port local 8080 vers le port 80 du serveur distant

ssh -L 8080:localhost:80 utilisateur@serveur

# Rendre accessible un service local depuis le serveur distant

ssh -R 9090:localhost:3000 utilisateur@serveur
```

SSH

- Configuration client

Fichier `~/.ssh/config`

```
# Configuration pour un serveur spécifique
Host monserveur
    HostName 192.168.1.100
    User admin
    Port 2222
    IdentityFile ~/.ssh/cle_monserveur
```

SSH

- Configuration client

```
# Configuration avec rebond

Host serveur-prod

    HostName 10.0.0.50

    User prod

    ProxyJump bastion.entreprise.com
```

SSH

- Configuration client

```
# Configuration globale
Host *
    ServerAliveInterval 60
    ServerAliveCountMax 3
    ControlMaster auto
    ControlPath ~/.ssh/sockets/%r@%h-%p
        ControlPersist 600
```

SSH

- BP & debug

Bonnes pratiques

- Utiliser des clés SSH plutôt que des mots de passe
- Changer le port par défaut
- Désactiver la connexion root
- Utiliser des passphrases sur les clés privées
- Mettre à jour régulièrement OpenSSH
- Surveiller les logs (`/var/log/auth.log`)
- Utiliser fail2ban pour bloquer les attaques par force brute

SSH

- BP & debug

```
# Vérifier le statut du service
sudo systemctl status ssh
# Tester la configuration
sudo sshd -T
# Logs SSH
sudo tail -f /var/log/auth.log
sudo journalctl -u ssh -f
```

SSH

- TP

 # 1

- *créer un user “imtlinux” et généré un paire de clés SSH en rsa 4096*
- *connectez vous sur le serveur SSH de votre voisin avec cet utilisateur*
- *configurer votre connexion SSH pour utiliser le serveur de votre voisin comme serveur de rebond et connectez vous sur le serveur du voisin de votre voisin*

Redirections et pipes
- Comprendre les flux entrée et sortie

bash

```
# Chaque processus Unix/Linux a 3 flux :
# STDIN (0) - Entrée standard
# - Par défaut : clavier
# - Données envoyées au programme
# STDOUT (1) - Sortie standard
# - Par défaut : écran/terminal
# - Résultats normaux du programme
# STDERR (2) - Erreur standard
# - Par défaut : écran/terminal (même que STDOUT)
# - Messages d'erreur du programme
```

Redirections et pipes

- Comprendre les flux entrée et sortie

 # Exemple visuel :

 # Clavier → [STDIN] → Programme → [STDOUT] → Écran

↓

 # [STDERR] → Écran

Redirections et pipes
- Comprendre les flux entrée et sortie

```
# Créer des fichiers de test
echo "Ligne 1" > test.txt
echo "Ligne 2" >> test.txt
echo "Ligne 3" >> test.txt
# STDOUT : sortie normale
cat test.txt                # Affiche le contenu (STDOUT)
ls -l test.txt              # Informations fichier (STDOUT)
# STDERR : messages d'erreur
cat fichier_inexistant.txt  # Message d'erreur (STDERR)
ls fichier_inexistant.txt   # Erreur (STDERR)
```

Redirections et pipes

- Comprendre les flux entrée et sortie

```
# STDIN : entrée du programme

cat                         # Attend saisie clavier (STDIN)

# Tapez quelque chose et appuyez Ctrl+D


# Programme utilisant les 3 flux

grep "mot" test.txt           # STDIN=fichier, STDOUT=résultat

grep "mot" inexistant.txt     # STDERR=erreur "fichier non trouvé"
```

Redirections et pipes
- Redirection de sortie

```
# Redirection STDOUT avec > (écrase)
ls -l > listing.txt          # Sauvegarde listing dans fichier
cat listing.txt              # Voir le résultat
date > timestamp.txt         # Écrase le contenu précédent
cat timestamp.txt
# Redirection STDOUT avec >> (ajoute)
echo "Première ligne" > log.txt
echo "Deuxième ligne" >> log.txt
echo "Troisième ligne" >> log.txt
cat log.txt                  # Voir toutes les lignes
```

Redirections et pipes
- Redirection de sortie

```
# Exemples pratiques

ps aux > processus.txt          # Sauvegarde liste processus

df -h > espace_disque.txt       # État des disques

free -h >> espace_disque.txt    # Ajouter info mémoire


# Redirection vers /dev/null (poubelle)

ls -l > /dev/null               # Sortie ignorée

echo "test" > /dev/null         # Supprime la sortie
```

Redirections et pipes
- Redirection de sortie

```
# Rediriger seulement STDERR

ls fichier_existe fichier_inexistant 2> erreurs.txt

cat erreurs.txt              # Voir les erreurs

# La sortie normale s'affiche toujours à l'écran

# Rediriger STDERR vers /dev/null (ignorer erreurs)

find / -name "*.log" 2> /dev/null    # Ignore "Permission denied"

grep -r "motif" /etc 2> /dev/null    # Recherche sans erreurs

# Rediriger STDOUT et STDERR séparément

ls fichier_existe fichier_inexistant > sorties.txt 2> erreurs.txt

cat sorties.txt              # Sorties normales

cat erreurs.txt              # Messages d'erreur
```

Redirections et pipes
- Redirection de sortie

```
# Combiner STDOUT et STDERR dans le même fichier

ls fichier_existe fichier_inexistant > tout.txt 2>&1

cat tout.txt                # Tout mélangé

# Syntaxe moderne équivalente

ls fichier_existe fichier_inexistant &> tout.txt

ls fichier_existe fichier_inexistant &>> tout.txt  # Ajouter
```

Redirections et pipes
- Redirection d'entrée

```
# Utiliser un fichier comme entrée
cat < test.txt              # Équivalent à cat test.txt
sort < test.txt             # Trier les lignes du fichier
wc -l < test.txt            # Compter lignes
# Différence subtile :
cat test.txt                # cat reçoit le nom du fichier
cat < test.txt              # cat reçoit le contenu via STDIN
# Exemple avec mail (si installé)
   mail user@example.com < message.txt
```

Redirections et pipes

- Les pipes

```
# Pipe | : STDOUT d'une commande devient STDIN de la suivante

# commande1 | commande2 | commande3

# Exemple simple

ls -l | head -5              # 5 premières lignes de ls

ps aux | grep firefox        # Processus contenant "firefox"

cat /etc/passwd | wc -l      # Nombre d'utilisateurs

# Visualisation :

# ls -l → [STDOUT] → | → [STDIN] → head -5 → [STDOUT] → écran
```

Redirections et pipes
- Pipes avec filtres courants

```
ps aux | grep bash            # Processus bash

ls -l | grep "^d"             # Seulement les dossiers

cat /etc/passwd | grep ":/bin/bash"  # Utilisateurs avec bash


# Options utiles de grep dans pipes

ps aux | grep -v grep         # Exclure grep lui-même

ps aux | grep -i firefox      # Insensible à la casse

ls -l | grep -c "\.txt"       # Compter fichiers .txt
```

Redirections et pipes
- Pipes avec filtres courants

```
ps aux | sort -k 3 -nr        # Trier par CPU (colonne 3, numérique, inverse)

ls -l | sort -k 5 -n          # Trier par taille (colonne 5)

cat /etc/passwd | sort        # Tri alphabétique simple


# Options de sort

du -h * | sort -hr            # Trier tailles (human readable, reverse)

ps aux | sort -k 4 -nr        # Trier par mémoire (%MEM)
```

Redirections et pipes
- Pipes avec filtres courants

```
# uniq nécessite des données triées !

cat fichier.txt | sort | uniq        # Éliminer doublons

cat fichier.txt | sort | uniq -c     # Compter occurrences

cat fichier.txt | sort | uniq -d     # Seulement les doublons


# Exemple pratique : extensions de fichiers

find . -name "*.*" | sed 's/.*\.//' | sort | uniq -c
```

Redirections et pipes
- Pipes avec filtres courants

```
ls -l | wc -l              # Nombre de fichiers
ps aux | wc -l             # Nombre de processus
cat fichier.txt | wc -w    # Nombre de mots
cat fichier.txt | wc -c    # Nombre de caractères
```

Redirections et pipes
- Pipes avec filtres courants

```
ls -l | wc -l              # Nombre de fichiers
ps aux | wc -l             # Nombre de processus
cat fichier.txt | wc -w    # Nombre de mots
cat fichier.txt | wc -c    # Nombre de caractères
```

Redirections et pipes
- Pipes avec filtres courants

```
# Que font ces commandes ?

ps aux | sort -k 3 -nr | head -10

ls -la | sort -k 5 -nr | head -6    # +1 pour la ligne total

who | cut -d' ' -f1 | sort | uniq

find . -name "*.*" | grep -o '\.[^.]*$' | sort | uniq -c | sort -nr

cat fichier.txt | awk '{print length, $0}' | sort -nr | head -5
```

Redirections et pipes
- Maintenant on conbine les 2

```
# Rediriger le résultat d'un pipe

ps aux | grep bash > processus_bash.txt

ls -l | sort -k 5 -nr > fichiers_par_taille.txt

# Ignorer les erreurs dans un pipe

find / -name "*.log" 2>/dev/null | head -10

# Sauvegarder ET afficher (tee)

ps aux | tee processus.txt | grep firefox

ls -l | tee listing.txt | wc -l

# tee avec ajout

echo "Nouvelle entrée" | tee -a log.txt
```

Redirections et pipes
- Maintenant on conbine les 2

```
# Mélanger STDOUT et STDERR dans le pipe

find / -name "*.conf" 2>&1 | grep -v "Permission denied"


# Séparer erreurs et résultats

find / -name "*.log" > found_logs.txt 2> search_errors.txt


# Pipeline avec log des erreurs

ps aux 2>debug.log | sort -k 3 -nr | head -5 > top_cpu.txt
```

Redirections et pipes
- Commandes utiles : CUT

```
# Extraire colonnes spécifiques

ps aux | cut -d' ' -f1,2,11        # Colonnes USER, PID, COMMAND

cat /etc/passwd | cut -d':' -f1,3  # Nom et UID

ls -l | cut -c1-10                 # 10 premiers caractères


# Exemples pratiques

who | cut -d' ' -f1                # Noms des utilisateurs connectés

df -h | cut -d' ' -f1,4            # Partition et espace libre

ip addr | grep inet | cut -d' ' -f6   # Adresses IP
```

Redirections et pipes
- Commandes utiles : AWK

```
# Syntaxe de base : awk 'pattern {action}'

ps aux | awk '{print $1, $2, $11}'     # Colonnes USER, PID, COMMAND

ps aux | awk '$3 > 1.0 {print $0}'     # Processus > 1% CPU

# Calculs

ls -l | awk '{sum += $5} END {print sum}'  # Somme des tailles

ps aux | awk '{sum += $4} END {print sum}' # Somme %MEM

# Avec conditions

df -h | awk '$5 > "80%" {print $0}'    # Partitions > 80% pleines

ps aux | awk 'NR > 1 && $3 > 0.1 {print $1, $11, $3"%"}' # CPU > 0.1%
```

Redirections et pipes
- Commandes utiles : SED

```
# Remplacement simple
ps aux | sed 's/root/admin/g'      # Remplacer "root" par "admin"
cat fichier.txt | sed 's/ /_/g'   # Remplacer espaces par underscores
# Suppression de lignes
ps aux | sed '1d'                 # Supprimer première ligne
ps aux | sed '/grep/d'            # Supprimer lignes contenant "grep"
# Extraction de lignes
sed -n '10,20p' fichier.txt       # Lignes 10 à 20
ps aux | sed -n '/firefox/p'      # Seulement lignes avec "firefox"
```

Redirections et pipes
- Commandes utiles : GREP

```
# Recherche dans un flux d'entrée

grep 'toto' /var/log/auth.log      # Recherche le user toto dans un log

grep -i   'toto' /var/log/auth.log # Recherche le user toto ou TOTO (insensible a la casse)dans un log

grep -r  'toto' /var/log/*.log # Recherche le user toto ou TOTO (insensible a la casse)dans tous les logs du repertoire
```

Redirections et pipes - Bonnes pratiques

```
# Éviter les pipes inutiles

cat fichier.txt | grep motif     # ❌ Inefficace

grep motif fichier.txt           # ✅ Mieux


cat fichier.txt | head -10       # ❌

head -10 fichier.txt             # ✅


# Ordre optimal dans les pipes

find / -name "*.log" | grep apache | head -10     # ❌ grep sur tout

find / -name "*apache*.log" | head -10            # ✅ find plus précis
```

Redirections et pipes - Bonnes pratiques

```
# Utiliser des outils spécialisés

ps aux | grep -v grep | awk '{print $1}' | sort | uniq    # ❌ Complexe

ps -eo user --no-headers | sort | uniq                    # ✅ Plus simple
```

Redirections et pipes - Debug

```
# Technique de debugging : tee intermédiaire

ps aux | tee debug1.txt | grep firefox | tee debug2.txt | wc -l

# Vérifier chaque étape

cat debug1.txt | head -5

cat debug2.txt

# Utiliser set -x dans les scripts pour tracer

set -x

ps aux | grep bash | wc -l

set +x
```

Redirections et pipes - Debug

```
# Vérifier les codes de retour

ps aux | grep "inexistant"

echo "Code de retour: $?"


# Pipeline avec gestion d'erreur

ps aux | grep bash || echo "Aucun processus bash trouvé"
```

Redirections et pipes
- TP1

 # Tout doit être redirigé dans un fichier avec le numéro de la question ex: file1.txt, file2.txt etc ..

 # 1. Créez une liste de tous vos fichiers dans votre home

 # 2. Ajoutez l'espace disque utilisé

 # 3. Tentez de lister un dossier inexistant et capturez l'erreur

 # 4. Combinez sortie normale et erreurs

 # 5. Vérifiez le contenu des fichiers créés

Redirections et pipes - TP2

 # 1. Trouvez les 5 plus gros processus par mémoire

 # 2. Comptez le nombre de processus bash en cours

 # 3. Listez tous les utilisateurs uniques connectés

 # 4. Trouvez tous les fichiers .txt dans votre home et comptez-les

 # 5. Affichez les 10 dernières lignes des logs système (si accessible)

Redirections et pipes - TP3

 # 1. afficher toutes les extensions uniques de fichiers dans /usr/bin

 # 2. Trouvez les processus qui consomment le plus de mémoire et sauvegardez

 # 3. Créez un rapport système simple avec

- *l'espace disque disponible*
- *la mémoire libre disponible*

 # 4. Affichez le rapport avec pagination(less)

 # 5. Analysez les connexions réseau (netstat ou ss)

Scripts et variables d'environnement - Comprendre les variables

Qu'est-ce qu'une variable d'environnement ?

- **Variable globale** accessible à tous les programmes lancés depuis le shell
- **Configuration système** : chemin des programmes, éditeur par défaut, langue, etc.
- **Transmission aux processus enfants** : héritées par les programmes lancés
- **Personnalisation** : adapter l'environnement à l'utilisateur

Scripts et variables d'environnement - Comprendre les variables

```
# Différence entre variable shell et variable d'environnement

MA_VARIABLE="valeur"          # Variable shell locale

export MA_VARIABLE="valeur"    # Variable d'environnement (globale)

# Test de la différence

MA_VAR_LOCALE="test"

export MA_VAR_GLOBALE="test"

# Lancer un nouveau shell

bash

echo $MA_VAR_LOCALE          # Vide (non héritée)

echo $MA_VAR_GLOBALE         # "test" (héritée)

exit
```

Scripts et variables d'environnement
- Variables systémes

```
echo $PATH          # Chemins des exécutables
echo $HOME          # Répertoire personnel
echo $USER          # Nom d'utilisateur
echo $SHELL         # Shell par défaut
echo $PWD           # Répertoire courant
echo $OLDPWD        # Répertoire précédent
echo $HOSTNAME      # Nom de la machine
echo $LANG          # Langue système
echo $TZ            # Fuseau horaire
```

Scripts et variables d'environnement - Variables configuration

```
echo $EDITOR          # Éditeur par défaut
echo $BROWSER         # Navigateur par défaut
echo $PAGER           # Pagineur par défaut (less, more)
echo $TERM            # Type de terminal
echo $DISPLAY         # Serveur X11 (interface graphique)
```

Scripts et variables d'environnement
- Variables de l'historique

```
echo $HISTSIZE          # Nombre de commandes en mémoire
echo $HISTFILE          # Fichier d'historique
echo $HISTCONTROL       # Contrôle de l'historique
```

Scripts et variables d'environnement - Variables :: Exemple

```
# Voir toutes les variables d'environnement

env | head -20

printenv | head -20

# Filtrer les variables

env | grep PATH

printenv | grep -i hist

# Variables spécifiques

echo "Utilisateur: $USER dans $HOME"

echo "Shell: $SHELL sur $HOSTNAME"

echo "Répertoire: $PWD"
```

Scripts et variables d'environnement
- Variables :: Manipulation

```
# Affichage
echo $VARIABLE_NAME          # Afficher une variable
echo ${VARIABLE_NAME}        # Syntaxe alternative (plus sûre)
echo "$VARIABLE_NAME"        # Avec guillemets (recommandé)
# Création de variables
MA_VAR="Hello World"         # Variable locale
export MA_VAR_EXPORT="Global" # Variable d'environnement
declare -x MA_DECLARE="Export" # Avec declare
# Test d'existence
echo ${MA_VAR:-"valeur par défaut"}    # Si MA_VAR vide, utilise défaut
echo ${MA_VAR:="nouvelle valeur"}      # Si MA_VAR vide, l'assigne
```

Scripts et variables d'environnement
- Variables :: la variable PATH

```
# Afficher PATH de façon lisible

echo $PATH

echo $PATH | tr ':' '\n'      # Chaque dossier sur une ligne

# Ajouter un dossier au PATH (temporaire)

export PATH="$PATH:$HOME/bin"

export PATH="/usr/local/bin:$PATH"  # En début (priorité)

# Vérifier qu'un programme est dans PATH

which python3

type python3

command -v python3
```

Scripts et variables d'environnement
- Variables :: la variable PATH

```
# Créer un script dans PATH

mkdir -p ~/bin

echo '#!/bin/bash

echo "Mon script personnel"' > ~/bin/monscript

chmod +x ~/bin/monscript

export PATH="$PATH:$HOME/bin"

monscript                    # Accessible depuis partout
```

Scripts et variables d'environnement
- Variables :: Persistance

 # Hiérarchie des fichiers de configuration :

 # /etc/profile → Système, tous utilisateurs

 # /etc/bash.bashrc → Bash système

 # ~/.profile → Personnel, tous shells

 # ~/.bashrc → Personnel, bash seulement

 # ~/.bash_profile → Personnel, bash login

 # ~/.bash_logout → Exécuté à la déconnexion

Scripts et variables d'environnement
- Variables :: Persistance

```
# Ordre de lecture (bash login) :

# 1. /etc/profile

# 2. ~/.bash_profile OU ~/.profile

# 3. ~/.bashrc (souvent appelé par .bash_profile)


# Voir quel fichier existe

ls -la ~ | grep -E "\.(profile|bashrc|bash_)"
```

Scripts et variables d'environnement
- Variables :: Persistance .bashrc

```
# Éditer .bashrc

nano ~/.bashrc

# Ajouter à la fin de .bashrc :

# ========== MES VARIABLES D'ENVIRONNEMENT ==========

# Éditeur par défaut

export EDITOR=nano

export VISUAL=nano
```

Scripts et variables d'environnement
- Variables :: Persistance .bashrc

```
# Historique amélioré
export HISTSIZE=5000
export HISTFILESIZE=10000
export HISTCONTROL=ignoreboth:erasedups
export HISTTIMEFORMAT="%d/%m/%y %T "
# Chemins personnalisés
export PATH="$PATH:$HOME/bin:$HOME/.local/bin"
# Variables de développement
export WORKSPACE="$HOME/workspace"
export PROJECTS="$HOME/Documents/projets"
```

Scripts et variables d'environnement
- Variables :: Persistance .bashrc

```
# Couleurs pour less
export LESS='-R'
export LESS_TERMCAP_mb=$'\E[1;31m'     # begin blinking
export LESS_TERMCAP_md=$'\E[1;36m'     # begin bold
export LESS_TERMCAP_me=$'\E[0m'        # end mode
export LESS_TERMCAP_se=$'\E[0m'        # end standout-mode
export LESS_TERMCAP_so=$'\E[01;44;33m' # begin standout-mode
export LESS_TERMCAP_ue=$'\E[0m'        # end underline
export LESS_TERMCAP_us=$'\E[1;32m'     # begin underline
```

Scripts et variables d'environnement
- Variables :: Persistance .bashrc

```
# Recharger la configuration
source ~/.bashrc


# Vérifier les nouvelles variables
echo $EDITOR
echo $HISTSIZE
echo $WORKSPACE
```

Scripts et variables d'environnement
- Les scripts

```
# Créer le premier script
cat > premier_script.sh << 'EOF'
#!/bin/bash
# Mon premier script bash
# Auteur: Votre nom
# Date: $(date +%Y-%m-%d)
echo "=== MON PREMIER SCRIPT ==="
echo "Bonjour $(whoami) !"
echo "Nous sommes le $(date)"
echo "Votre shell est : $SHELL"
EOF
```

Scripts et variables d'environnement

- Les scripts

```
# Rendre exécutable

chmod +x premier_script.sh

# Exécuter

./premier_script.sh
```

Scripts et variables d'environnement

- Les scripts

#!/bin/bash : Kesako ?

- L'interpréteur que vous voulez utiliser , il y en a plusieurs (shell, bash, php, python etc …)

Mon premier script bash

Auteur: Votre nom

Date: $(date +%Y-%m-%d)

- Les commentaires en hait de scripts , autant prendre la bonne pratique tout de suite , trés important
  - le nom
  - le titre
  - la date
  - plein d'autres comme la version , les arguments etc …..

Scripts et variables d'environnement
- Les scripts:: Variables

```
bash

# Script avec variables

cat > variables.sh << 'EOF'

#!/bin/bash

# Variables locales au script

nom="Alice"

age=25

ville="Paris"

# Variables d'environnement dans le script

export SCRIPT_VAR="Disponible partout"
```

Scripts et variables d'environnement

- Les scripts:: Variables

```
# Utilisation des variables

echo "Nom: $nom"

echo "Âge: $age ans"

# Calculs avec variables

annee_actuelle=$(date +%Y)

annee_naissance=$((annee_actuelle - age))

# Variables spéciales

echo "Nom du script: $0"

echo "Nombre d'arguments: $#"

echo "Tous les arguments: $@"

echo "PID du script: $$"
```

Scripts et variables d'environnement
- Les scripts:: Arguments et paramètres

```
# Variables spéciales pour les arguments

# $0 = nom du script

# $1, $2, $3... = arguments positionnels

# $# = nombre d'arguments

# $@ = tous les arguments

# $? = code de retour de la dernière commande
```

Scripts et variables d'environnement
- Les scripts:: Arguments et paramètres

```
echo "=== GESTION DES ARGUMENTS ==="
echo "Nom du script: $0"
echo "Nombre d'arguments: $#"
if [ $# -eq 0 ]; then
    echo "Aucun argument fourni."
    echo "Usage: $0 argument1 argument2 ..."
    exit 1
fi
echo "Premier argument: $1"
echo "Deuxième argument: ${2:-"Non fourni"}"
echo "Troisième argument: ${3:-"Non fourni"}"
```

Scripts et variables d'environnement
- Les scripts:: Arguments et paramètres

```
echo "Tous les arguments:"
for arg in "$@"; do
    echo "  - $arg"
done


echo "Arguments avec numérotation:"
i=1
for arg in "$@"; do
    echo "  Argument $i: $arg"
    ((i++))
done
```

Scripts et variables d'environnement

- Les scripts:: Arguments et paramètres

./arguments.sh hello world 123

Scripts et variables d'environnement
- Les scripts:: Les conditions

```
# Script avec conditions

# Test de chaîne

if [ "$nom" = "admin" ]; then

    echo "Accès administrateur accordé"

elif [ "$nom" = "Invité" ]; then

    echo "Veuillez fournir votre nom en argument"

else

    echo "Utilisateur normal: $nom"

fi



# Test numérique
```

Scripts et variables d'environnement
- Les scripts:: Les conditions

```
# Test numérique
if [ $# -gt 1 ]; then
    age=$2
    if [ "$age" -gt 18 ]; then
        echo "Vous êtes majeur ($age ans)"
    else
        echo "Vous êtes mineur ($age ans)"
    fi
fi


# Test de fichier
```

Scripts et variables d'environnement
- Les scripts:: Les conditions

```
# Test de fichier

fichier_config="$HOME/.bashrc"

if [ -f "$fichier_config" ]; then

    echo "Fichier de configuration trouvé: $fichier_config"

    echo "Taille: $(stat -c%s "$fichier_config") octets"

else

    echo "Fichier de configuration non trouvé"

fi
```

```
# Tests multiples

if [ -f "$HOME/.bashrc" ] && [ -r "$HOME/.bashrc" ]; then
```

Scripts et variables d'environnement
- Les scripts:: Les conditions

```
# Tests multiples

if [ -f "$HOME/.bashrc" ] && [ -r "$HOME/.bashrc" ]; then

    echo "Le fichier .bashrc existe et est lisible"

fi
```

Scripts et variables d'environnement
- Les scripts:: Les boucles

```
# Boucle sur arguments
for arg in "$@"; do
    echo "  - $arg"
done
# Boucle sur fichiers
echo "Fichiers .txt dans le répertoire courant:"
for fichier in *.txt; do
    if [ -f "$fichier" ]; then
        echo "  - $fichier ($(wc -l < "$fichier") lignes)"
    fi
done
```

Scripts et variables d'environnement

- Les scripts:: Les boucles

```
# Boucle numérique
echo "Compteur de 1 à 5:"
for i in {1..5}; do
    echo "  Iteration $i"
done

# Boucle C-style
echo "Carrés de 1 à 5:"
for ((i=1; i<=5; i++)); do
    echo "  $i² = $((i*i))"
done
```

Scripts et variables d'environnement
- Les scripts:: Les boucles

```
# Boucle while

compteur=1

echo "Compte à rebours:"

while [ $compteur -le 5 ]; do

    echo "  $compteur..."

    ((compteur++))

    sleep 1

done

echo "Terminé!"
```

Scripts et variables d'environnement
- Les scripts:: Les boucles

```
# Lecture de fichier ligne par ligne
if [ -f "/etc/passwd" ]; then
    echo "Premiers utilisateurs système:"
    head -5 /etc/passwd | while IFS=: read user x uid gid info home shell; do
        echo "  $user (UID:$uid) -> $shell"
    done
fi
```

Scripts et variables d'environnement

- Les scripts:: Fonction

```
# Définition de fonctions
saluer() {
    local nom=${1:-"Invité"}
    echo "Bonjour $nom, bienvenue !"
}
calculer_age() {
    local annee_naissance=$1
    local annee_actuelle=$(date +%Y)
    local age=$((annee_actuelle - annee_naissance))
    echo $age
}
```

Scripts et variables d'environnement

- Les scripts:: Fonction

```
menu() {
    echo "=== MENU PRINCIPAL ==="
    echo "1. Saluer"
    echo "2. Calculer l'âge"
    echo "4. Quitter"
    echo -n "Votre choix: "
}
```

Scripts et variables d'environnement

- Les scripts:: Fonction

```
# Programme principal

echo "=== SCRIPT AVEC FONCTIONS ==="

# Appels de fonctions

saluer

saluer "Alice"

age=$(calculer_age 1990)

echo "Une personne née en 1990 a $age ans"
```

Scripts et variables d'environnement
- TP1 Vous pouvez vous aider de l'IA

 # 1. Créez un script qui sauvegarde tous les fichiers de log présent dans /var/log/ et qui font moins de 10 M

- *# Variabiliser la limite de 10M en argument , on pourra donc choisir la limite en paramètre du script*

 # 2. Créez un script check le load average

- *si le load average est supérieur a votre nombre de processeur : renvoyer "critical"*
- *si le load average est egal a votre nombre de processeur : renvoyer "warning"*
- *sinon renvoyer "ok"*

 # 2. Transformer ces srcripts en commande pour qu'il soit accessible juste en tapant:

- *check_la pour le load average*
- *save_log pour la sauvegarde de log*

Réseau et services
- Configuration réseau basique

Comprendre la configuration réseau Linux (5 min)

- **Interface réseau** : point de connexion (eth0, wlan0, lo)
- **Adresse IP** : identifiant unique sur le réseau
- **Masque de réseau** : définit la taille du réseau local
- **Passerelle** : routeur pour sortir du réseau local
- **DNS** : serveurs de résolution de noms de domaine

Réseau et services
- Configuration réseau basique

```
# Interfaces réseau et adresses IP
ip addr show                # Commande moderne
ip a                        # Raccourci
ifconfig                    # Commande traditionnelle (si installée)
# Analyser la sortie ip addr :
# 1: lo: <LOOPBACK,UP,LOWER_UP>
#    inet 127.0.0.1/8 scope host lo
# 2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP>
#    inet 192.168.1.100/24 brd 192.168.1.255 scope global eth0
```

Réseau et services
- Configuration réseau basique

```
# Interfaces spécifiques
ip addr show eth0          # Interface ethernet
ip addr show wlan0         # Interface WiFi
ip addr show lo            # Interface loopback (127.0.0.1)


# Table de routage
ip route show              # Routes actuelles
ip r                       # Raccourci
route -n                   # Format traditionnel
```

Réseau et services
- Configuration réseau basique

```
# Exemple de sortie :

# default via 192.168.1.1 dev eth0  → Passerelle par défaut

# 192.168.1.0/24 dev eth0 scope link → Réseau local


# Serveurs DNS

cat /etc/resolv.conf          # Configuration DNS

# nameserver 8.8.8.8

# nameserver 8.8.4.4
```

Réseau et services
- Configuration réseau basique

```
# Configuration réseau (Ubuntu moderne)
ls /etc/netplan/            # Netplan (Ubuntu 18.04+)
cat /etc/netplan/*.yaml     # Configuration YAML


# Gestionnaire réseau
systemctl status NetworkManager
nmcli device status         # État des interfaces avec NetworkManager
nmcli connection show       # Connexions configurées
```

Réseau et services
- Commande diagnostique réseau: PING

```
# Ping basique

ping google.com              # Test continu (Ctrl+C pour arrêter)

ping -c 4 google.com         # 4 pings seulement

# Tests de connectivité progressifs

ping -c 1 127.0.0.1          # 1. Loopback (pile TCP/IP)

ping -c 1 192.168.1.1        # 2. Passerelle locale

ping -c 1 8.8.8.8            # 3. DNS Google (Internet)

ping -c 1 google.com         # 4. Résolution DNS + Internet

# IPv6

    ping6 -c 3 google.com
```

Réseau et services
- Commande diagnostique réseau: WGET

```
# wget - téléchargement de fichiers

wget https://example.com/file.txt          # Téléchargement simple

wget -O nouveaunom.txt https://example.com/file.txt  # Renommer

wget -c https://example.com/bigfile.iso    # Reprendre téléchargement

wget --progress=bar https://example.com/file.txt     # Barre de progression

wget -r -l 2 https://example.com/          # Récursif, 2 niveaux max


# Test de connectivité web

wget --spider -q https://google.com && echo "Site accessible"
```

Réseau et services
- Commande diagnostique réseau: CURL

```
# curl - outil polyvalent
curl https://example.com                # Afficher contenu
curl -o file.txt https://example.com/file.txt  # Sauvegarder
curl -I https://example.com             # Headers seulement
curl -s https://example.com             # Silencieux
curl -L https://example.com             # Suivre redirections
# Tests API avec curl
curl -X GET https://api.github.com/users/octocat
curl -H "Accept: application/json" https://api.example.com/data
# Tester vitesse/temps de réponse
   curl -w "@curl-format.txt" -s -o /dev/null https://google.com
```

Réseau et services
- Analyse réseau avancé

```
# Trace de route
traceroute google.com        # Chemin des paquets (si installé)
tracepath google.com         # Alternative (souvent pré-installé)
mtr google.com               # Traceroute continu (si installé)
# Résolution DNS
nslookup google.com          # Interrogation DNS basique
dig google.com               # Plus détaillé (si installé)
dig @8.8.8.8 google.com      # Serveur DNS spécifique
# Tests de ports
telnet google.com 80         # Test connexion port 80
nc -zv google.com 80         # netcat (si installé)
```

Réseau et services
- Analyse réseau avancé

```
# Exemple de diagnostic complet

echo "=== DIAGNOSTIC RÉSEAU ==="

echo "Interface réseau:"

ip addr show | grep "inet " | grep -v "127.0.0.1"

echo "Passerelle:"

ip route show | grep default

echo "DNS:"

cat /etc/resolv.conf | grep nameserver

echo "Test connectivité:"

ping -c 1 8.8.8.8 && echo "Internet OK" || echo "Problème Internet"
```

Réseau et services
- Services et démons

Qu'est-ce qu'un service/démon ?

- **Service** : programme qui s'exécute en arrière-plan
- **Démon** : service Unix/Linux (souvent terminé par 'd')
- **Systemd** : gestionnaire de services moderne (Ubuntu 15.04+)
- **Unit files** : fichiers de configuration des services
- **États** : active, inactive, enabled, disabled, failed

Réseau et services
- Services et démons

```
# Exemples de services courants :

# - sshd : serveur SSH

# - NetworkManager : gestion réseau

# - cron : tâches planifiées

# - apache2/nginx : serveurs web

# - mysql : base de données

# - bluetooth : connectivité Bluetooth
```

Réseau et services
- Services: systemctl

```
# Lister tous les services
systemctl list-units --type=service          # Services actifs
systemctl list-units --type=service --all    # Tous les services
systemctl list-unit-files --type=service     # Fichiers de service
# Filtrer les services
systemctl list-units --type=service --state=active    # Actifs seulement
systemctl list-units --type=service --state=failed    # En échec
systemctl list-units --type=service | grep ssh        # Contenant "ssh"
```

Réseau et services

- Services: systemctl

```
# État d'un service spécifique
systemctl status ssh                # Service SSH
systemctl status NetworkManager     # Gestionnaire réseau
systemctl status cron               # Planificateur de tâches
```

Réseau et services
- Services: systemctl

```
# Analyse de la sortie systemctl status :
# ● ssh.service - OpenBSD Secure Shell server
#    Loaded: loaded (/lib/systemd/system/ssh.service; enabled; vendor preset: enabled)
#    Active: active (running) since Mon 2023-12-15 10:30:00 UTC; 2h 15min ago
#    Main PID: 1234 (sshd)
#      Tasks: 1 (limit: 1108)
#     Memory: 2.5M
#     CGroup: /system.slice/ssh.service
#             └─1234 /usr/sbin/sshd -D
```

Réseau et services
- Services: systemctl

```
# Vérifier si un service est actif
systemctl is-active ssh            # active/inactive
systemctl is-enabled ssh           # enabled/disabled
systemctl is-failed ssh            # failed ou pas
# Services les plus importants à connaître
echo "=== SERVICES SYSTÈME IMPORTANTS ==="
for service in ssh NetworkManager cron systemd-resolved; do
    status=$(systemctl is-active $service 2>/dev/null)
    enabled=$(systemctl is-enabled $service 2>/dev/null)
    echo "$service: $status ($enabled)"
    done
```

Réseau et services
- Services: systemctl

```
# Démarrer un service
sudo systemctl start ssh
sudo systemctl start apache2          # Si installé
# Arrêter un service
sudo systemctl stop ssh
sudo systemctl stop apache2
# Redémarrer un service
sudo systemctl restart ssh            # Arrêt complet puis démarrage
sudo systemctl reload ssh             # Recharger config sans arrêter
sudo systemctl reload-or-restart ssh  # Reload si possible, sinon restart
```

Réseau et services
- Services: systemctl

```
# Vérifier l'état après action

sudo systemctl restart ssh

systemctl status ssh

# Exemple pratique avec un service web (si installé)

# sudo apt install apache2

# sudo systemctl start apache2

# curl http://localhost              # Tester le serveur

# sudo systemctl stop apache2

# curl http://localhost              # Devrait échouer
```

Réseau et services
- Services: systemctl

```
# Activer au démarrage (enable)
sudo systemctl enable ssh          # Démarre automatiquement au boot
sudo systemctl enable apache2
# Désactiver au démarrage (disable)
sudo systemctl disable apache2     # Ne démarre pas automatiquement
# Vérifier l'état d'activation
systemctl is-enabled ssh           # enabled/disabled
systemctl is-enabled apache2
# Masquer un service (empêche tout démarrage)
sudo systemctl mask apache2        # Impossible de démarrer
sudo systemctl unmask apache2      # Retirer le masque
```

Réseau et services
- Services: systemctl

```
# Lister les services activés au démarrage
systemctl list-unit-files --type=service --state=enabled
# Services par cible (target = ancien runlevel)
systemctl list-dependencies multi-user.target    # Services mode multi-utilisateur
systemctl list-dependencies graphical.target     # Services mode graphique
# Exemple de configuration complète d'un service
echo "=== CONFIGURATION SERVICE SSH ==="
sudo systemctl enable ssh          # Activer au démarrage
sudo systemctl start ssh           # Démarrer maintenant
systemctl status ssh               # Vérifier l'état
systemctl is-enabled ssh           # Confirmer l'activation
  systemctl is-active ssh          # Confirmer le fonctionnement
```

Réseau et services
- Services: Logs

```
# Logs avec journalctl (systemd)
journalctl                    # Tous les logs (q pour quitter)
journalctl -n 20              # 20 dernières entrées
journalctl -f                 # Suivi en temps réel (comme tail -f)
# Logs d'un service spécifique
journalctl -u ssh             # Logs du service SSH
journalctl -u ssh -n 10       # 10 dernières entrées SSH
journalctl -u ssh --since "1 hour ago"  # Depuis 1 heure
journalctl -u ssh --since "2023-12-15 10:00:00"  # Depuis date précise
```

Réseau et services
- Services: Logs

```
# Filtres temporels
journalctl --since yesterday          # Depuis hier
journalctl --since "2023-12-15"       # Depuis date
journalctl --since "10:00" --until "11:00"  # Entre 10h et 11h
journalctl -b                         # Depuis le dernier boot
journalctl -b -1                      # Boot précédent
# Niveaux de priorité
journalctl -p err                     # Erreurs seulement
journalctl -p warning                 # Avertissements et plus graves
journalctl -p info                    # Informations et plus graves
```

Réseau et services
- Fichers importants

Fichiers de configuration importants :

- `/etc/resolv.conf` - Configuration DNS
- `/etc/netplan/*.yaml` - Configuration réseau (Ubuntu moderne)
- `/etc/systemd/system/` - Services personnalisés
- `/lib/systemd/system/` - Services système
- `/var/log/` - Logs traditionnels

Réseau et services
- Choses a connaitre

```
# SSH - Accès distant

systemctl is-active ssh && echo "  - État: Actif" || echo "  - État: Inactif"

# NetworkManager - Gestion réseau

systemctl is-active NetworkManager && echo "  - État: Actif" || echo "  - État: Inactif"

# systemd-resolved - DNS

systemctl is-active systemd-resolved && echo "  - État: Actif" || echo "  - État: Inactif"
```

Réseau et services
- Choses a connaitre

```
# Exemple de ports couramment utilisés
echo "22   - SSH"
echo "80   - HTTP"
echo "443  - HTTPS"
echo "21   - FTP"
echo "25   - SMTP (mail)"
echo "53   - DNS"
echo "3306 - MySQL"
echo "5432 - PostgreSQL"
# Vérifier les ports en écoute
ss -tuln | grep LISTEN | head -5
```

Réseau et services
- TP1

 # Faites un script de diagnostique réseau qui teste que :

- *Les interfaces réseaux sont up*
- *on arrive bien a joindre la gateway*
- *on sait sortir sur internet*
- *on sait résoudre google.com*

Réseau et services
- TP2

 # Faites un script de diagnostique des services qui teste que :

- *Le service ssh est démarré et activé au démarrage*
- *que le service http est démarré si ce n'est pas le cas on installe apache2 et on l'active au démarrage*

Réseau et services
- TP3

 # Crée un service qui écoute sur le port 3000 (vous pouvez vous faire aider de l'IA)

Surveillance système et dépannage
- Surveillance de l'espace disque

```
# Affichage de base
df                      # Espace en blocs (peu lisible)
df -h                   # Format humain (K, M, G, T)
df -H                   # Base 1000 au lieu de 1024
# Analyser la sortie de df -h :
# Filesystem      Size  Used Avail Use% Mounted on
# /dev/sda1        20G   15G  4.2G  79% /
# /dev/sda2       100G   45G   50G  48% /home
# tmpfs           2.0G     0  2.0G   0% /dev/shm
```

Surveillance système et dépannage
- Surveillance de l'espace disque

```
# Options utiles
df -h /                    # Partition racine seulement
df -h /home                # Partition home seulement
df -T                      # Afficher le type de système de fichiers
df -i                      # Inodes (nombre de fichiers) au lieu de l'espace

# Surveillance automatique
df -h | awk '$5 > 80 {print "ALERTE: " $0}'    # Partitions > 80%
df -h | grep -E "8[0-9]%|9[0-9]%|100%"         # Regex pour > 80%
```

Surveillance système et dépannage
- Surveillance de l'espace disque

```
# Script de surveillance espace disque
#!/bin/bash
echo "=== SURVEILLANCE ESPACE DISQUE ==="
echo "Date: $(date)"
echo
# Seuils d'alerte
WARNING_THRESHOLD=80
CRITICAL_THRESHOLD=90
df -h | grep -E "^/dev/" | while read line; do
    partition=$(echo $line | awk '{print $1}')
    use_percent=$(echo $line | awk '{print $5}' | tr -d '%')
```

Surveillance système et dépannage - Surveillance de l'espace disque

```
    mount_point=$(echo $line | awk '{print $6}')


    if [ "$use_percent" -ge "$CRITICAL_THRESHOLD" ]; then
        echo "🔴 CRITIQUE: $partition ($mount_point) à ${use_percent}%"
    elif [ "$use_percent" -ge "$WARNING_THRESHOLD" ]; then
        echo "🟡 ATTENTION: $partition ($mount_point) à ${use_percent}%"
    else
        echo "🟢 OK: $partition ($mount_point) à ${use_percent}%"
    fi
done
```

Surveillance système et dépannage
- Surveillance de l'espace disque:: DU

```
# Taille des dossiers

du -h /home/user/              # Taille totale du dossier

du -h --max-depth=1 /home/user/ # Seulement premier niveau

du -sh /home/user/             # Résumé seulement (-s = summarize)

# Analyser l'utilisation actuelle

du -h --max-depth=1 . | sort -hr    # Dossiers triés par taille

du -sh * | sort -hr                 # Tous les éléments du répertoire courant

# Trouver les plus gros fichiers

find . -type f -exec du -h {} + | sort -hr | head -20
```

Surveillance système et dépannage
- Surveillance de l'espace disque:: DU

```
# Analyser un dossier spécifique

echo "=== ANALYSE DE /var/log ==="

sudo du -sh /var/log/*/ 2>/dev/null | sort -hr | head -10


echo "=== PLUS GROS FICHIERS DANS /tmp ==="

   sudo find /tmp -type f -exec du -h {} + 2>/dev/null | sort -hr | head -10
```

Surveillance système et dépannage
- Surveillance de la mémoire:: FREE

```
# Affichage de base
free                    # En kilo-octets
free -h                 # Format humain (M, G)
free -m                 # En méga-octets
free -g                 # En giga-octets
```

Surveillance système et dépannage
- Surveillance de la mémoire:: FREE

```
# Analyser la sortie de free -h :
#               total        used        free      shared  buff/cache   available
# Mem:           7.7G        2.1G        1.2G        156M        4.4G        5.3G
# Swap:          2.0G          0B        2.0G

# Colonnes importantes :
# total     : mémoire physique totale
# used      : mémoire utilisée par les processus
# free      : mémoire complètement libre
# buff/cache: mémoire utilisée pour cache/buffers (récupérable)
# available : mémoire réellement disponible (free + récupérable)
```

Surveillance système et dépannage
- Surveillance de la mémoire:: FREE

```
# Surveillance continue

free -h -s 5                # Actualisation toutes les 5 secondes

watch -n 2 "free -h"        # Avec watch, actualisation toutes les 2s


# Calcul du pourcentage d'utilisation

free | awk 'NR==2{printf "Mémoire utilisée: %.2f%%\n", $3*100/$2}'


# Informations détaillées sur la mémoire

cat /proc/meminfo | head -20    # Statistiques détaillées

cat /proc/meminfo | grep -E "(MemTotal|MemFree|MemAvailable|Buffers|Cached)"
```

Surveillance système et dépannage - Analyse des processus

```
# Top 10 processus par mémoire
ps aux --sort=-%mem | head -11

# Processus utilisant plus de 100MB
ps aux | awk '$6 > 100000 {print $2, $6/1024 "MB", $11}' | head -10

# Mémoire utilisée par utilisateur
ps aux | awk '{mem[$1] += $6} END {for (user in mem) printf "%s: %.1fMB\n", user, mem[user]/1024}' | sort -k2 -nr

# Surveiller un processus spécifique
watch -n 2 "ps aux | grep firefox | grep -v grep"
```

Surveillance système et dépannage - Analyse des processus

```
# Informations sur le swap

echo "=== UTILISATION DU SWAP ==="

free -h | grep Swap

swapon --show                # Partitions swap actives

cat /proc/swaps              # Alternative
```

Surveillance système et dépannage - Commande dépannage courantes

```
# Charge système
uptime                    # Load average
cat /proc/loadavg         # Charge détaillée + processus actifs
# CPU et processus
top                       # Vue d'ensemble temps réel
htop                      # Version améliorée (si installé)
ps aux --sort=-%cpu | head -10 # Top CPU
# I/O disque
iostat 2 5                # Statistiques I/O (si installé avec sysstat)
iotop                     # Top I/O temps réel (si installé)
```

Surveillance système et dépannage - Commande dépannage courantes

```
# Réseau
ss -tuln                # Connexions réseau
netstat -tuln           # Alternative (si installé)
ss -s                   # Statistiques réseau

# Température et hardware (si disponible)
sensors                 # Températures (si lm-sensors installé)
lscpu                   # Informations CPU
lsblk                   # Disques et partitions
lsusb                   # Périphériques USB
lspci                   # Périphériques PCI
```

Surveillance système et dépannage
- Commande dépannage courantes

```
# Tests réseau
ping -c 4 8.8.8.8          # Test connectivité Internet
dig google.com             # Test DNS
nslookup google.com        # Alternative DNS
# Processus et ports
lsof -i :22                # Qui utilise le port 22
lsof -i :80                # Port 80
fuser -v 22/tcp            # Alternative pour voir qui utilise un port
# Fichiers ouverts
lsof | head -20            # Tous les fichiers ouverts
lsof /var/log/syslog       # Qui accède au fichier syslog
```

Surveillance système et dépannage - TP1

 # Faites un script de surveillance des log systèmes (utiliser journalctl -p err) sur la dernière heure

Surveillance système et dépannage - TP2

 # Faites un script qui affiche

- *la date actuelle*
- *l'uptime du serveur*
- *le nom de l'OS et sa version*
- *la mémoire disponible (swap et ram)*
- *la taille du disque restante*
- *les services qui tourne actuellement*

TP de fin de cours

 # Pour ce Tp de fin on va:

- *déployer une nouvelle VM Ubuntu server (en NAT pour accéder a internet)*
- *Installer iptables dessus et bloquer tous les accés sauf ceux pour http, https et ssh*
- *installer l'outils GLPI qui est un outils de gestion de parc et de ticketing*
    - *respecter les points suivants:*
        - *créer des user spécifiques pour chaque services*
        - *créer des sauvegardes de la base de données et du code*
        - *faire un rotation des logs*
        - *installer mysql, apache2 et php*
        - *Ne pas utiliser docker*