Auto-formation 1

Bases et concepts de GNU/LINUX

1.Rôle des répertoires système se situant à la racine :

/bin (binary)Contient les commandes essentielles accessibles à tous les utilisateurs (ex. : ls, cp, mv). Ces commandes sont disponibles même si /usr n'est pas monté.

/boot Contient les fichiers de démarrage du système (ex. : noyau, GRUB).

Ce répertoire est parfois situé sur une partition dédiée.

/dev (devices)Contient les fichiers de périphériques (ex : disques, claviers,terminaux).

Géré dynamiquement par **udev**.

/etc (editable text configuration)Contient tous les fichiers de configuration du système.(ex :/fstab, /passwd, /hostname, /network/interfaces, /ssh, /apache2).

/home Contient les répertoires personnels des utilisateurs. Chaque utilisateur a ses fichiers personnels, ses paramètres, ses téléchargements, etc.

/lib Contient les bibliothèques partagées pour les programmes dans /bin et /sbin.

/lib64 est utilisé pour les bibliothèques 64 bits sur les systèmes 64 bits.

/media (raccord)Point de montage pour les médias amovibles (clés USB, CD, etc.).

Généralement géré automatiquement par des environnements de bureau comme GNOME ou KDE.

/mnt (raccord system) Utilisé comme point de montage temporaire pour les administrateurs (par ex., pour monter une partition manuellement).

/opt (optional)Contient les applications tierces installées manuellement.

Exemple : /opt/google/chrome/

/proc Répertoire spécial contenant des fichiers virtuels représentant les processus en cours et l'état du noyau (ex: /cpuinfo, /meminfo, /[pid]).

/root Le dossier personnel de l'administrateur système (root),(super-utilisateur).

Différent de /home/root : celui-ci est à part pour des raisons de sécurité .

/run Contient des fichiers temporaires utilisés par le système en cours d'exécution.

Nettoyé à chaque redémarrage

/sbin (sudo)Contient les commandes systèmes critiques utilisées pour le démarrage,

la récupération, la maintenance. Réservé aux administrateurs (root,sudoers)

/srv (services) Contient les données de service (serveurs web /http, FTP /ftp , etc.).

/sys Fournit des informations sur les périphériques et le noyau (système virtuel).

(crée après /proc pour une version standardisée et plus structure)

/tmp Espace pour stocker des fichiers temporaires utilisés par les programmes.

/usr Contient des programmes et données utilisateur (non essentiels au boot).

/var Contient les fichiers variables qui changent fréquemment (logs, spools, caches, etc.).

2.Comprendre comment les systèmes gère leurs droits d'accès aux ressources et d'exécution inter-services et inter-processus.

-Chaque fichier/répertoire a un proprietaire(user) et un groupe. Des droits pour chacun

Ainsi qu'aux autres utilisateur : lecture (r), écriture (w), exécution (x).

Exemple : -rwxr-xr-- 1 alice dev 1234 date fichier.sh

Alice = proprietaire, dev = groupe,

rwx =droits de alice (user)

r-x = droits de dev (groupe)

r = droit des autres (user hors alice et hors dev)

parfois utilisé sous la forme de chiffres avec la règle 4, 2,1(w, r, x)

Exemple 732 = wrx proprietaire, rx groupe, r pour le autres.

Il existe plusieurs commande de gestions des droits :

chmod → changer les permissions (chmod 755 script.sh)

chown → changer le propriétaire (chown bob script.sh)

chgrp → changer le groupe (chgrp dev script.sh)

Permissions spéciales

- SUID : exécuter avec les privilèges du propriétaire (chmod u+s fichier)
- SGID : pour le groupe, souvent utilisé pour les répertoires partagés
- Sticky bit : empêche la suppression de fichiers par d'autres utilisateurs (utilisé dans /tmp)

-Communication et exécution inter-processus

Dans un système d'exploitation multitâche comme Linux, les processus doivent souvent

interagir entre eux pour échanger des données, se synchroniser ou collaborer sur une tâche commune. Ce besoin est satisfait par deux concepts fondamentaux : l'exécution inter-processus et la communication inter-processus (IPC – Inter-Process Communication).

Communication Inter-Processus (IPC)

Linux fournit plusieurs mécanismes d'IPC, classés selon leur nature : mémoire, message ou fichier. Ces mécanismes permettent l'échange d'informations entre processus indépendants ou apparentés.

Mécanismes d'inter-processus (IPC)

1. **Fichiers** : fichiers temporaires dans /tmp ou autres répertoires.
2. **Pipes** :

   Anonymes : Les pipes sont unidirectionnels et utilisés pour communiquer entre un processus parent et son enfant.

   Només (FIFOs) : Les FIFOs sont similaires aux pipes, mais nommés et accessibles par des processus non apparentés via le système de fichiers
3. **Sockets UNIX** : Les sockets UNIX permettent une communication bidirectionnelle entre processus locaux, via un fichier spécial.
4. **POSIX /**

   Message Queues : Les queues de messages permettent l'envoi de messages formatés entre processus.

   Shared Memory : La mémoire partagée permet aux processus de lire/écrire dans un espace mémoire commun. Très rapide, elle nécessite toutefois une synchronisation stricte (semaphores, mutex).

   Semaphores : structures IPC avancées (ipcs pour voir, ipcrm pour supprimer)
5. **Signals** : signaux pour envoyer des commandes simples à un processus (kill -SIGTERM pid)

Comparaison des Méthodes IPC

Mécanisme	Type	Direction	Vitesse	Complexité	Usage typique
Pipe	Flux	Uni	Moyenne	Faible	Parent ↔ Enfant
FIFO	Fichier	Uni	Moyenne	Faible	Processus indépendants

Mécanisme	Type	Direction	Vitesse	Complexité	Usage typique
Mémoire partagée	Mémoire	Bi	Très rapide	Élevée	Échange rapide de données
Message Queue	Messages	Bi	Moyenne	Moyenne	Communication structurée
Signaux	Signal	Uni	Très rapide	Très simple	Notifications
Socket UNIX	Réseau local	Bi	Bonne	Moyenne	Client ↔ Serveur local

Contrôle des droits IPC

Les objets IPC ont des **droits d'accès** similaires aux fichiers (propriétaire, groupe, permissions).

Seuls les utilisateurs avec les droits adéquats peuvent lire, écrire ou modifier une ressource IPC.

Exécution Inter-Processus

L'exécution inter-processus concerne la manière dont les processus sont créés, gérés et synchronisés dans un système Linux.

Création de processus

La création de processus repose principalement sur la fonction fork(), qui duplique le processus courant. Le nouveau processus, appelé processus fils, hérite de l'environnement du parent. Il est également possible de remplacer un processus par un autre programme via exec(), ou d'utiliser clone() pour des cas spécifiques comme les threads ou les conteneurs.

Synchronisation

La synchronisation permet d'éviter les conflits d'accès aux ressources partagées. Linux propose plusieurs mécanismes :

wait() ou waitpid() pour que le parent attende la fin du fils.

Sémaphores et mutex pour synchroniser des sections critiques.

3.Comprendre le service apt de gestion de paquets

Un **paquet** est un **fichier compressé** qui regroupe **tout ce qu'il faut pour installer, configurer et exécuter un logiciel** sur Linux. Il permet une installation automatisée, fiable et maintenable grâce aux gestionnaires de paquets comme **APT**.

Sous Linux, la gestion des logiciels repose sur des gestionnaires de paquets. Dans les distributions basées sur Debian (comme Ubuntu, Linux Mint, etc.), le système de gestion de paquets repose principalement sur l'outil APT (Advanced Package Tool). Il permet d'installer, mettre à jour, configurer et supprimer facilement des logiciels.

Qu'est-ce que APT ?

APT (Advanced Package Tool) est un ensemble d'outils en ligne de commande permettant d'interagir avec le système de paquets .deb utilisé par Debian et ses dérivés.

APT automatise plusieurs tâches :

- Téléchargement de paquets depuis des dépôts distants
- Résolution automatique des dépendances
- Installation ou suppression de logiciels
- Mise à jour du système

Fonctionnement d'APT

APT s'appuie sur deux composants essentiels :

- **Les dépôts** : serveurs en ligne contenant les paquets disponibles.
- **La base de données locale** : liste des paquets disponibles et installés.

Chaque paquet contient :

- Les fichiers binaires (le logiciel)
- Les métadonnées (nom, version, dépendances, etc.)

Exemple d'utilisation complète

```
sudo apt update              # Actualise les listes de paquets
sudo apt install vlc         # Installe le lecteur VLC
sudo apt upgrade             # Met à jour tous les paquets
sudo apt remove vlc          # Supprime VLC (mais garde la config)
sudo apt autoremove          # Nettoie les dépendances inutiles
```

APT et autres outils

Outil	Fonction	Utilisation principale
apt	Interface utilisateur simplifiée	Usage courant (depuis Ubuntu 16.04+)
apt-get	Interface plus bas-niveau	Scripts, rétrocompatibilité
dpkg	Gestion des paquets localement	Installation de .deb sans dépendances

APT est un outil central pour les utilisateurs et administrateurs de systèmes Debian-based. Il simplifie considérablement la gestion des logiciels grâce à une interface cohérente et automatisée. Maîtriser APT permet de maintenir un système à jour, propre et sécurisé.

Mise en pratique de certains des points vu précédemment :

Nous allons faire un exemple pratique d'un service web Apache qui interagit avec une base de données MariaDB et un moteur Php sur un système Linux, en mettant en lumière la gestion des droits, l'isolation inter-services et la communication entre processus.

1. Préparation :

Creatien d'un utilsateur et de son groupe

sudo groupadd apachers

sudo useradd -aG apache Auser

sudo passwd Auser

auser (mot de passe simple car test)

Installation des paquets avec APT

sudo apt update # Mise à jour des listes de paquets

sudo apt install apache2 mariadb-server php libapache2-mod-php php-mysql

APT va automatiquement :

APT va automatiquement :

- Télécharger les paquets .deb

- Résoudre et installer les dépendances (ex : libapache2-mod-php)
- Lancer les services (via systemd)
- Créer les fichiers de configuration dans /etc/

2. Répertoires utilisés dans l'arborescence Linux

Répertoire	Rôle
`/etc/apache2/`	Configurations du serveur Apache
`/var/www/html/`	Dossier racine du site Web (accessible par Apache)
`/etc/mysql/`	Configuration de MariaDB
`/var/lib/mysql/`	Stockage physique des bases de données
`/var/log/`	Logs des services (`apache2`, `mysql`, etc.)
`/srv/`	Peut accueillir des services web/FTP personnalisés

3. Droits et sécurité

Gestion des utilisateurs/services

- Apache tourne sous l'utilisateur **www-data**
- MariaDB tourne souvent sous l'utilisateur **mysql**

ps aux | grep apache2

ps aux | grep mysql

Permettent de vérifier si les processus apache2 et mysql sont en cours d'exécution

ps : affiche la liste des processus en cours.

a : affiche les processus de tous les utilisateurs.

u : affiche l'utilisateur associé au processus.

x : affiche aussi les processus sans terminal de contrôle

Le | (pipe) envoie la sortie de ps aux dans la commande grep.

grep apache2 cherche les lignes contenant "apache2".

grep mysql cherche les lignes contenant "mysql".

Isolation inter-services

Apache et MariaDB tournent **dans des processus séparés**

Chacun a des **droits limités** grâce à leur propre utilisateur système

Le site web ne peut accéder à MariaDB **que par le port local 3306** et **grâce aux droits PHP** configurés

Droits sur les fichiers

Creation d’un fichier test dans le repertoir html

```
sudo mkdir -p /var/www/html
```

```
sudo nano /var/www/html/test.php
```

```
<?php
$mysqli = new mysqli("localhost", "utilisateurweb", "auser", "site1");
if ($mysqli->connect_error) {
   die("Erreur de connexion : " . $mysqli->connect_error);
}
echo "Connexion réussie à MariaDB !";
?>
```

si Auser n’est pas propriétaire → Apache pourrait ne pas lire ce fichier.

```
sudo chown -R Auser:apachers /var/www/html
```

```
sudo chmod -R 770 /var/www/html
```

droit d’écriture de lecture et d’exécution a Auser et son groupe apachers sur le répertoire /var/www/html et ses sous-répertoires et fichiers.

Creation site1.local

```
sudo mkdir -p /var/www/site1
```

```
sudo nano /etc/apache2/sites-available/site1.local.conf
```

```
<VirtualHost *:80>
   ServerName site1.local
   DocumentRoot /var/www/site1

   <Directory /var/www/site1>
      AllowOverride All
      Require all granted
   </Directory>

   ErrorLog ${APACHE_LOG_DIR}/site1_error.log
   CustomLog ${APACHE_LOG_DIR}/site1_access.log combined
</VirtualHost>
```

Activer le site et redémarrer Apache

sudo a2ensite site1.local.conf

sudo systemctl reload apache2

Ajouter site1.local au fichier hosts

Édite le fichier /etc/hosts :

sudo nano /etc/hosts

Ajoute cette ligne :

127.0.0.1 site1.local

Tester sur le navigateur

http://site1.local pour voir '' Bienvenue sur site1.local''

Structure de répertoires et bonnes pratiques

Dossier	Utilité
/etc/apache2/	Fichiers de configuration Apache (VirtualHost, modules, etc.)
/etc/mysql/	Configuration de MariaDB
/var/www/html/	Dossier **racine du site web**
/var/lib/mysql/	**Données réelles** des bases MariaDB (accès restreint à mysql)
/var/log/apache2/	Logs d'accès et d'erreurs Apache
/var/log/mysql/	Logs du moteur MariaDB

Création d'une base de données

sudo mysql

CREATE DATABASE site1;

CREATE USER 'Auser'@'localhost' IDENTIFIED BY 'auser';

GRANT ALL PRIVILEGES ON site1.* TO 'Auser'@'localhost';

FLUSH PRIVILEGES;

EXIT;

Analyse inter-processus

Processus en cours :

ps -u Auser # Affiche les processus Apache

ps -u mysql # Affiche les processus MariaDB

Communication :

Apache (Auser) → exécute **PHP**

PHP → établit une **connexion réseau locale (localhost:3306)** à **MariaDB**
→ Communication via **socket réseau local** (inter-processus)

Vérification des services :

systemctl status apache2

systemctl status mariadb

redémarrer le service apache

sudo systemctl restart apache2

Ensuite, accédez à http://localhost/test.php pour voir ‘’Connexion réussie à MariaDB !’’

Conclusion

Ce projet montre la **mise en place d’un service web complet sous Linux**, mettant en lumière plusieurs concepts :

Concept	**Mise en application**
Gestion des paquets	via apt install
Structure du système de fichiers	/etc, /var, /srv, /usr/bin
Droits d’accès	utilisateurs Auser, mysql
Isolation inter-services	Apache, PHP et MariaDB séparés
Communication inter-processus	socket local entre PHP et MariaDB