Révisions du module M4 bases de GNU/linux

Dans le cadre de notre apprentissage de l'administration des systèmes Linux, la maîtrise des compétences fondamentales liées à la gestion et à la sécurisation d'un serveur est essentielle. Ces connaissances permettent non seulement d'assurer le bon fonctionnement des systèmes, mais aussi de garantir la sécurité, la fiabilité et la performance des infrastructures.

Ce devoir de révision a pour objectif de consolider les notions indispensables à toute personne souhaitant administrer un serveur Linux de manière efficace et sécurisée. Il s'agit de revoir des points clés tels que l'exploitation d'un serveur en ligne de commande, la gestion des utilisateurs, la surveillance des systèmes et des journaux, ainsi que l'application des mises à jour.

En parallèle, cette révision permet de mieux comprendre la philosophie Unix et l'importance de l'écosystème open source, tout en faisant la distinction entre les principales familles de distributions Linux et leurs usages respectifs.
Enfin, la sécurité occupe une place centrale, avec des notions essentielles telles que la gestion des permissions, les mécanismes d'authentification, et les bonnes pratiques en matière d'administration.

L'ensemble de ces compétences constitue un socle solide pour progresser dans l'administration des systèmes et réussir la prochaine évaluation.

Philosophie Unix

La philosophie Unix repose sur des principes clés :

- **Faire une seule chose et la faire bien** : chaque outil ou programme Unix est conçu pour accomplir une tâche précise efficacement.
- **Simplicité** : les outils restent simples à utiliser, évitant la complexité inutile.
- **Modularité** : les programmes peuvent être combinés entre eux via des mécanismes comme les pipes pour créer des fonctions plus complexes.
- **Transparence** : les systèmes et outils favorisent la clarté, permettant aux utilisateurs de comprendre facilement ce qui se passe.

Cette philosophie a profondément influencé la conception du système GNU/Linux.

Logiciel libre et open source

- **Définition** :
  Un logiciel libre garantit à l'utilisateur la liberté d'exécuter, copier, modifier et distribuer le logiciel. L'open source est une approche similaire, mettant l'accent sur la disponibilité gratuite du code source.
- **Avantages** :
  - Liberté d'utilisation et d'adaptation selon les besoins.
  - Sécurité améliorée grâce à la transparence du code.
  - Collaboration et innovation grâce aux communautés actives.
  - Absence de coûts de licence.
- **Importance des communautés** :
  Les projets libres sont souvent soutenus par des communautés internationales qui

développent, maintiennent et partagent le savoir. Ces communautés sont un moteur d'innovation et de support, ayant permis la création de nombreuses distributions Linux.

Principales distributions Linux

Distribution	Facilité	Stabilité	Nouveautés	Public cible	Usage typique
Ubuntu	Très facile	Stable (LTS)	Moyenne	Débutants, grand public	PC, serveurs web
Debian	Moyenne	Très stable	Moins fréquentes	Serveurs, utilisateurs avancés	Serveurs, base d'autres distros
Fedora	Moyenne	Bonne	Très récente	Développeurs, testeurs	Développement, testing
CentOS / Alma / Rocky	Moyenne	Très stable	Peu fréquentes	Entreprises	Serveurs en production
Arch Linux	Difficile	Variable	Très récente	Utilisateurs avancés	Systèmes personnalisés
Linux Mint	Très facile	Stable	Moyenne	Débutants Windows	PC personnel
openSUSE	Moyenne	Bonne	Variable	Professionnels, devs	Poste de travail, serveurs
Kali Linux	Moyenne	Stable	Moyenne	Experts en sécurité	Tests de pénétration, audit
Red Hat	Moyenne à difficile	Très stable	Moins fréquentes	Grandes entreprises	Serveurs critiques, cloud, entreprise

En résumé, Unix, créé dans les années 1970, a posé les bases d'un système multitâches et multi-utilisateurs, inspirant toute une génération de systèmes d'exploitation. Le projet GNU, initié en 1983 par Richard Stallman, a apporté une dimension essentielle avec sa vision d'un système libre, en développant tous les outils nécessaires à un OS compatible Unix, à l'exception du noyau. C'est avec la création du noyau Linux par Linus Torvalds en 1991, combiné aux outils GNU, que le système complet GNU/Linux a vu le jour, incarnant pleinement la philosophie Unix et les principes du logiciel libre.

GNU/Linux est un système d'exploitation libre basé sur deux éléments :

Le noyau Linux : Partie centrale qui gère le matériel (mémoire, processeur, disque, périphériques, réseau…).

Les outils GNU : Programmes essentiels qui permettent d'utiliser le système (shell, compilateurs, éditeurs, commandes…).

Fonctionnement Global **:**

1. **Démarrage (Boot) :**

    - L'ordinateur charge un programme d'amorçage (comme GRUB).
    - GRUB lance le **noyau Linux**.
2. **Noyau Linux :**
    - Active les pilotes matériels (USB, disque, réseau…).
    - Gère la mémoire, les processus et le matériel.
    - Lance le processus principal (init ou systemd).
3. **Système GNU et Services :**
    - Les services de base démarrent (réseau, affichage, etc.).
    - Les utilitaires GNU (bash, ls, cp, etc.) permettent à l'utilisateur d'interagir avec le système.
4. **Interface Utilisateur :**
    - Soit en mode console (terminal).
    - Soit via une interface graphique (en lançant un serveur graphique comme **X.org** ou **Wayland** + un environnement de bureau comme GNOME, KDE…).
5. **Système de Fichiers :**
    - Linux accède aux fichiers via son arborescence (/).
    - Chaque fichier ou dossier joue un rôle précis.

Le système Linux repose sur une arborescence de fichiers hiérarchique où tout est fichier : fichiers classiques, périphériques, processus, service, etc.

Principaux répertoires racine de linux

Répertoire	Rôle
/	Racine du système. Tous les fichiers et répertoires partent d'ici.
/bin	Contient les **commandes essentielles** (ex. : ls, cp, mv, rm). Accessible même en mode de secours.
/sbin	Similaire à /bin, mais réservé aux **commandes d'administration** système (ex. : fsck, reboot).
/etc	Contient les **fichiers de configuration** du système et des applications (ex : /etc/hosts, /etc/fstab).
/home	Dossiers personnels des utilisateurs (ex : /home/toto). Les fichiers personnels y sont stockés.
/root	Dossier personnel du **superutilisateur** (root).
/lib	Bibliothèques essentielles au fonctionnement du système et des commandes de base.
/lib64	Bibliothèques 64 bits sur certains systèmes.
/usr	Applications et fichiers partagés accessibles à tous les utilisateurs.
/usr/bin	Programmes non essentiels pour l'administration, mais utiles aux utilisateurs.
/usr/sbin	Utilitaires système pour administrateurs.
/var	Données variables, comme les logs (/var/log), les mails, les impressions.
/tmp	Fichiers temporaires (vidé au redémarrage).
/opt	Logiciels additionnels installés manuellement.
/mnt et /media	Points de montage pour les périphériques amovibles (clé USB, disque externe).
/dev	Fichiers représentant les **périphériques matériels** (disques, ports USB, etc.).

Répertoire	Rôle
/proc	Répertoire virtuel contenant des informations sur les **processus et le noyau**.
/sys	Interface pour interagir avec le noyau et le matériel.
/srv	Données propres aux services réseau (sites web, FTP, etc.).

Le système de fichiers Linux repose sur une organisation claire et hiérarchisée des répertoires racine, chacun ayant un rôle précis. Le dossier /boot contient les fichiers essentiels au démarrage, tandis que /proc et /sys permettent au noyau de gérer les processus et le matériel. Les commandes essentielles sont réparties entre /bin et /sbin, et chaque utilisateur dispose d'un espace personnel dans /home. La configuration du système se trouve dans /etc, les applications standards dans /usr et les applications supplémentaires dans /opt. Enfin, les fichiers variables et journaux sont centralisés dans /var. Cette structure rigoureuse assure à la fois la stabilité, la modularité et la facilité d'administration du système Linux.

Principales commande linux

Il existe une très grande quantité de commandes (binaires) qui permettent de naviguer, gérer les fichiers, surveiller les processus et administrer les services et nous allons ici énumérer et revoir les principales. Les arguments précisent à une commande ce qu'elle doit faire ou sur quoi agir, ce qui est parfois obligatoire sur certaines commandes. Sans eux, la commande utilise des valeurs par défaut qui ne correspondent pas toujours à ce que l'on veut.

Commandes Essentielles :

Commande	Description	Exemple
ls	Lister le contenu d'un répertoire	ls -l /var/log
cd	Changer de répertoire	cd /etc
pwd	Affiche le chemin du répertoire courant	pwd
cp	Copier un fichier ou dossier	cp fichier.txt /tmp/
mv	Déplacer ou renommer un fichier/dossier	mv fichier.txt /var/
rm	Supprimer un fichier ou répertoire	rm fichier.txt rm -r dossier/
nano	Éditeur de texte simple en terminal	nano /etc/hosts
vim	Éditeur de texte avancé	vim /etc/ssh/sshd_config
cat	Affiche le contenu d'un fichier	cat /etc/os-release
tail	Affiche la fin d'un fichier	tail -n 20 /var/log/syslog
head	Affiche le début d'un fichier	head -n 10 fichier.log
grep	Recherche dans les fichiers	grep "erreur" /var/log/syslog
mkdir	créer un repertoire	mkdir /var/www/html/snake

Gestion des processus

Commande	Description	Exemple
ps	Liste les processus en cours	ps aux
top	Affiche les processus en temps réel (utilisation CPU/RAM)	top

Commande	Description	Exemple
htop	Version améliorée de top (nécessite installation)	htop
kill	Termine un processus par son PID	kill 1234
killall	Termine tous les processus d'un nom spécifique	killall firefox
nice	Lance un processus avec une priorité spécifique	nice -n 10 script.sh
renice	Change la priorité d'un processus en cours	renice -n 5 -p 1234

Utilisation des services

Commande	Description	Exemple
systemctl status	Affiche l'état d'un service	systemctl status sshd
systemctl start	Démarre un service	systemctl start apache2
systemctl stop	Arrête un service	systemctl stop apache2
systemctl restart	Redémarre un service	systemctl restart apache2
systemctl enable	Active le démarrage auto d'un service	systemctl enable apache2
systemctl disable	Désactive le démarrage auto d'un service	systemctl disable apache2
systemctl reload	Recharge la configuration d'un service	systemctl reload nginx

Surveillance des événements et des journaux sous Linux

La surveillance des événements système et l'analyse des journaux (logs) sont essentielles pour assurer la bonne santé et la sécurité d'un système Linux. Ces logs permettent de comprendre le fonctionnement du système, détecter des anomalies, des erreurs, ou des tentatives d'intrusion.

Les fichiers de logs principaux

Sous Linux, la plupart des fichiers de logs se trouvent dans le répertoire /var/log/. Chaque fichier correspond à un type d'événement ou un service particulier. Par exemple :

- **/var/log/syslog** :Contient les messages généraux du système, comme les événements liés au démarrage, aux services, et aux opérations système courantes.
- **/var/log/auth.log**: Enregistre toutes les tentatives d'authentification, y compris les connexions réussies ou échouées, ainsi que les utilisations de commandes avec privilèges comme sudo.
- **/var/log/kern.log** : Regroupe les messages provenant du noyau Linux, notamment les erreurs matérielles ou les alertes critiques.
- **/var/log/apache2/**: Dossier où sont stockés les logs des serveurs web Apache, contenant les accès et erreurs HTTP.

Ces fichiers sont en général des fichiers texte que l'on peut lire avec un éditeur ou une commande.

Les commandes pour consulter et surveiller les logs

Pour analyser ces journaux, plusieurs commandes sont utilisées couramment :

- **journalctl** : Outil centralisé sous les systèmes utilisant systemd, qui permet d'accéder aux logs du système et des services. Il offre une lecture complète, triée et filtrée des événements.
- **tail -f** : Permet de suivre en temps réel les ajouts dans un fichier de log. Par exemple, tail -f /var/log/syslog affiche les derniers messages et continue d'afficher ceux qui arrivent.
- **less** : Pour lire un fichier de logs de manière paginée et navigable.
- **grep** : Pour rechercher un mot-clé ou un motif précis dans un fichier de logs.

La surveillance des fichiers de logs est un pilier fondamental de l'administration système Linux. Une bonne maîtrise de ces outils et fichiers permet d'assurer la stabilité, la sécurité et la performance du système.

Mises à jour et gestion des paquets sous Linux

Rôle des paquets

Un **paquet** est un fichier contenant un logiciel (programme, bibliothèque, ou outil) avec ses fichiers et ses informations d'installation. Le système de gestion de paquets permet d'installer, mettre à jour, et supprimer ces logiciels facilement, en gérant les dépendances entre eux.

- **dpkg** :
  C'est le gestionnaire de paquets de bas niveau. Il installe ou supprime directement un paquet .deb.
  Exemple :

```
sudo dpkg -i paquet.deb  # Installer un paquet local
sudo dpkg -r paquet      # Supprimer un paquet
```

Ne gère pas automatiquement les dépendances.

- **apt-get** :
  Outil en ligne de commande plus complet, qui télécharge les paquets depuis les dépôts, gère les dépendances, et installe ou met à jour les logiciels.
  Exemple :

```
sudo apt-get update      # Met à jour la liste des paquets
sudo apt-get upgrade     # Met à jour les paquets installés
```

- **apt** :
  Interface simplifiée et conviviale qui regroupe les fonctions essentielles d'apt-get et d'apt-cache. Elle est recommandée pour un usage courant.
  Exemple :

```
sudo apt update          # Mise à jour de la liste des paquets
sudo apt upgrade         # Mise à jour des paquets installés
sudo apt install paquet  # Installer un paquet
```

Rôle des mises à jour sous Linux

Les mises à jour sous Linux consistent à installer de nouvelles versions des **paquets** logiciels qui composent le système.
Mettre à jour les paquets permet de :

- Corriger des bugs et erreurs dans les logiciels.
- Améliorer les fonctionnalités et la performance.
- Appliquer des correctifs de sécurité pour protéger le système contre les vulnérabilités.

Le gestionnaire de paquets automatise le téléchargement, la résolution des dépendances et l'installation de ces mises à jour, garantissant ainsi que le système reste fonctionnel et sécurisé.

Principes des partages et des permissions sous Linux

Création et gestion des utilisateurs et groupes

- **Utilisateurs** : Chaque personne ou service accédant au système a un compte utilisateur.
- **Groupes** : Permettent de regrouper plusieurs utilisateurs pour gérer plus facilement les droits.

Commandes principales :

- adduser / useradd : Créent un nouvel utilisateur.
- groupadd : Crée un nouveau groupe.
- passwd : Modifie ou initialise le mot de passe d'un utilisateur.
- usermod : Modifie les paramètres d'un utilisateur existant (groupe, shell, etc.).
- deluser : Supprime un utilisateur.

Gestion des permissions

Les permissions sur les fichiers et dossiers peuvent être exprimées de deux façons : en lettres (rwx) ou en chiffres (notation octale).

Correspondance entre lettres et chiffres

Chaque droit correspond à une valeur numérique :

- **Lecture (r)** = 4 Permet de lire le contenu.
- **Écriture (w)** = 2 Permet de modifier le contenu.
- **Exécution (x)** = 1 Permet d'exécuter un fichier ou d'entrer dans un dossier.

Pour chaque catégorie (propriétaire, groupe, autres), on additionne ces valeurs selon les droits accordés.

Commandes principales :

- chmod : Modifie les permissions des fichiers/dossiers.
- chown : Change le propriétaire et/ou le groupe d'un fichier/dossier.
- umask : Définit les permissions par défaut lors de la création de nouveaux fichiers (avec une lecture inversé).

Exemple :

1. Changer le propriétaire et le groupe du répertoire 'apach'

```
sudo chown saiki:apache /chemin/vers/apach
```

2. Appliquer les permissions demandées :

```
sudo chmod 750 /chemin/vers/apach
```

« 3. Verification des droit et permitions du repertoire apache

```
ls -ld /chemin/vers/apach
```

réponse du terminal :

```
drwxr-x---  2 saiki apache 4096 Jul  3 12:34 /chemin/vers/apach
```

Permissions avancées : ACL (Access Control Lists)

Les ACL (Listes de Contrôle d'Accès) sont une extension du système classique de permissions Linux. Elles permettent d'attribuer des droits d'accès plus fins et précis sur des fichiers ou dossiers, en dépassant la limite des trois catégories traditionnelles (propriétaire, groupe, autres).
Le système classique (propriétaire, groupe, autres) peut être limité quand :

- Plusieurs utilisateurs ou groupes doivent avoir des permissions différentes sur un même fichier.
- On veut donner des droits spécifiques à certains utilisateurs sans changer le groupe principal du fichier.

Les ACL permettent donc de gérer des scénarios complexes de permissions.

Exemple :
setfacl -m u:mansah:rx fichier.txt
accorde à l'utilisateur **mansah** les droits de lecture et d'exécution sur **fichier.txt**, en plus des permissions classiques déjà définies.

Gestion des partages de fichiers
Pour permettre l'accès et le partage de fichiers entre plusieurs machines ou utilisateurs, plusieurs solutions sont utilisées selon les besoins et les environnements :
1. NFS (Network File System)

- Permet de partager des fichiers entre systèmes Linux et Unix.
- Fonctionne en montant un dossier distant sur un client comme s'il était local.
- Idéal pour les environnements homogènes Linux/Unix, notamment en entreprise ou dans les réseaux internes.

2. Samba

- Permet le partage de fichiers entre Linux et Windows.
- Implémente le protocole SMB/CIFS utilisé par Windows pour le partage réseau.
- Utile pour intégrer des machines Linux dans un réseau Windows ou partager des dossiers avec des utilisateurs Windows.

3. SSHFS (SSH Filesystem)

- Monte un répertoire distant accessible via SSH comme un dossier local.
- Simple à configurer, sécurisé grâce au chiffrement SSH.
- Pratique pour accéder à des fichiers à distance sans configurer de serveur de partage complexe.

Sécurité informatique sous Linux

La sécurité sous Linux repose sur une combinaison de bonnes pratiques d'administration, d'une gestion rigoureuse des droits et privilèges, ainsi que sur l'utilisation de mécanismes solides d'authentification et de protection réseau. Ces éléments permettent de limiter les risques d'intrusion, de fuite de données ou d'attaques malveillantes.

Bonnes pratiques d'administration

- **Politiques de mots de passe forts**
  Imposer des mots de passe complexes (longueur, caractères spéciaux) et forcer leur renouvellement régulier.
- **Mises à jour régulières**
  Installer les correctifs de sécurité via le gestionnaire de paquets pour corriger vulnérabilités et failles.

- **Restriction des accès SSH et clés SSH**
  Désactiver l'accès par mot de passe, utiliser uniquement les clés SSH. Limiter les utilisateurs autorisés à se connecter via SSH. Modifier le port SSH par défaut pour réduire les attaques automatisées.
- **Pare-feu**
  Configurer un pare-feu (comme ufw) pour contrôler précisément les connexions réseau autorisées ou bloquées. Par exemple, fermer tous les ports non nécessaires.
- **Proxy**
  Utiliser un proxy (comme Squid ou Apache et Nginx en reverse proxy) pour filtrer, contrôler et optimiser l'accès aux ressources web, améliorer la sécurité et la gestion du trafic.
- **Fail2ban**
  Mettre en place Fail2ban pour surveiller les journaux système, détecter les tentatives d'intrusion (comme les attaques par force brute sur SSH ou php) et bloquer automatiquement les adresses IP suspectes via le pare-feu.
- **Utilisation d'un VPN**
  Pour sécuriser les communications réseau sensibles, créer un tunnel VPN chiffré (comme WireGuard) limitant l'exposition directe des services à Internet.

Principe du moindre privilège

- **Limitation des droits à l'essentiel**
  Ne jamais utiliser le compte root pour des tâches courantes. Créer des utilisateurs avec seulement les permissions nécessaires.
- **Utilisation prudente de sudo**
  Accorder l'accès sudo uniquement aux utilisateurs de confiance, configurer précisément les commandes autorisées et auditer les accès.
- **Fichier .bashrc**
  Personnaliser l'environnement shell de chaque utilisateur. Utiliser .bashrc pour définir des alias, variables, ou alertes de sécurité à chaque connexion.

Mécanismes d'authentification

- **Authentification locale**
  Gérée via les fichiers /etc/passwd (informations utilisateur) et /etc/shadow (mots de passe chiffrés). PAM (Pluggable Authentication Modules) permet de configurer et renforcer ces mécanismes.
- **Authentification distante**
  Via SSH (clé publique/privée), LDAP, Kerberos ou RADIUS selon le contexte et la taille du réseau.

Cette approche globale garantit une sécurité renforcée, adaptée à tous les types d'environnements Linux, du poste utilisateur au serveur en production.

Ce compt rendu de révision a permis de consolider les connaissances fondamentales indispensables à l'administration d'un système GNU/Linux. Nous avons vu que maîtriser la philosophie Unix, comprendre les différents composants du système, et savoir utiliser les commandes principales est essentiel pour gérer efficacement un serveur ou un poste Linux. La gestion des utilisateurs, des permissions et des services ainsi que la surveillance des logs sont au cœur de la maintenance et de la sécurisation du système. Par ailleurs, la gestion des

paquets et les mises à jour régulières garantissent la stabilité et la protection face aux vulnérabilités.
Enfin, l'aspect sécurité ne peut être négligé : appliquer le principe du moindre privilège, utiliser des mécanismes d'authentification solides et déployer des outils comme les pare-feu, Fail2ban ou VPN contribuent à renforcer la résilience des infrastructures Linux face aux menaces.
Ces connaissances constituent un socle solide pour toute personne souhaitant évoluer dans le domaine de l'administration système, et préparer efficacement les évaluations à gvenir.