📚 RÉSUMÉ COMPLET ASSEU – EXAMEN LUNDI

Administration & Sécurité des SE (Unix)

⭐ PRIORITÉS EXAMEN (Probabilité haute)

Ordre d'importance:

1. **Disques/MBR/GPT/LVM/Montage/Quotas** → 25-30% de l'examen
2. **Utilisateurs/Groupes/Droits/SGID/Sticky bit** → 20-25%
3. **Shell & Commandes (find/grep/cut/sort/pipes)** → 15-20%
4. **Systemd (units, services, dépendances)** → 10-15%
5. **UEFI/efibootmgr/GRUB2** → 10-15%

CHAPITRE 1 : INTRODUCTION & SHELL

1.1 Composants d'un OS

- **Noyau (Kernel)** : gère CPU, RAM, périphériques
- **Shell** : interpréteur de commandes (interface utilisateur ↔ kernel)
- **Système de fichiers** : organisation hiérarchique (FHS)

1.2 Shell : Rôle & Types

Définition : Programme qui :

- Affiche le prompt `$` (user) ou `#` (root)
- Lit les commandes au clavier
- Les analyse et les exécute
- Revient au prompt

Types courants :

- `sh` (Bourne Shell) - par défaut
- `bash` (Bourne Again Shell)
- `ksh` (Korn Shell)

- `csh` / `tcsh` (C Shell)

1.3 Commandes essentielles

Commande	Fonction
`pwd`	Afficher le répertoire courant
`cd`, `cd ~`, `cd /`	Changer de répertoire
`ls -la`	Lister fichiers + détails
`mkdir dir`	Créer un répertoire
`cp source dest`	Copier
`mv source dest`	Déplacer/renommer
`rm fichier`	Supprimer un fichier
`rm -r dir`	Supprimer un répertoire
`cat fichier`	Afficher contenu
`echo "texte"`	Afficher du texte

1.4 Variables d'environnement principales

```
echo $PATH      # Chemins des exécutables
echo $HOME      # Répertoire personnel
echo $SHELL     # Shell par défaut
echo $USER      # Nom de l'utilisateur
echo $PWD       # Répertoire courant
```

Créer une variable d'environnement :

```
export MYVAR=valeur
```

CHAPITRE 2 : PERMISSIONS & DROITS D'ACCÈS

2.1 Types d'utilisateurs

- **Root (UID=0)** : Super-administrateur, contrôle total
- **Utilisateurs applicatifs (UID < 1000)** : comptes système (mail, ftp, etc.)
- **Utilisateurs simples (UID ≥ 1000)** : humains normaux, home dans `/home`

Règle : Défini dans `/etc/login.defs`

2.2 Permissions : rwx pour User/Group/Others

Sur un fichier :

- `r` (4) = Lire
- `w` (2) = Écrire
- `x` (1) = Exécuter

Sur un répertoire :

- `r` = Afficher le contenu
- `w` = Supprimer/renommer/créer des fichiers
- `x` = Accès/traverser le répertoire

2.3 Changer les droits (chmod)

Méthode littérale :

```
chmod u+x fichier       # Ajouter exécution au propriétaire
chmod g-w fichier       # Enlever écriture au groupe
chmod o=r fichier       # Autres = lecture seule
chmod a+r fichier       # Tous = ajout lecture
```

Méthode octale :

```
chmod 755 fichier       # u=rwx(7) g=rx(5) o=rx(5)
chmod 644 fichier       # u=rw(6) g=r(4) o=r(4)
chmod 777 fichier       # Tous les droits (rare!)
```

- `r = 4`, `w = 2`, `x = 1` → additionner pour chaque catégorie

2.4 Changer propriétaire/groupe

```
chown newuser fichier           # Changer propriétaire
chgrp newgroup fichier          # Changer groupe
chown user:group fichier        # Les deux
```

2.5 Droits étendus (TRÈS IMPORTANT EXAMEN)

SUID (Set User ID) - sur un fichier

- Exécution avec droits du propriétaire
- Exemple : `passwd` peut modifier `/etc/shadow` pour un user normal
- Notation octale : `4xxx` (ex: `4755`)
- Symbole : `u+s` → affiche `s` à la place de `x` du propriétaire

```
chmod 4755 script.sh   # Ajouter SUID
ls -l script.sh        # Affiche : -rwsr-xr-x
```

SGID (Set Group ID) - sur un répertoire

- **Fichiers créés dans un répertoire héritent du groupe du répertoire**
- Notation octale : `2xxx` (ex: `2755`)
- Symbole : `g+s`

Cas d'usage : Répertoire partagé où tous les fichiers doivent avoir le groupe du répertoire

```
mkdir /shared_dir
chgrp team /shared_dir
chmod g+s /shared_dir          # SGID appliqué
chmod 770 /shared_dir          # u=rwx, g=rwx, o=---
# Maintenant : user1 crée un fichier → groupe = team (pas user1!)
```

Sticky Bit - sur un répertoire

- **Seul le propriétaire du fichier (ou root) peut le supprimer**
- Notation octale : `1xxx` (ex: `1777`)
- Symbole : `o+t`
- Utilisé sur `/tmp` pour empêcher les suppression croisées

```
chmod 1777 /tmp        # Tous peuvent créer, mais seul le proprio peut supprimer son fichier
ls -ld /tmp            # Affiche : drwxrwxrwt
```

2.6 umask (permissions par défaut)

Lors de la création d'un fichier/répertoire, les permissions par défaut sont :

- **Fichier** : 666 − umask
- **Répertoire** : 777 − umask

```
umask                   # Afficher umask courant (ex: 0022)
umask 0077              # Modifier le umask
# Avec umask 0022 :
# - Fichier créé : 666 - 022 = 644 (rw-r--r--)
# - Répertoire créé : 777 - 022 = 755 (rwxr-xr-x)
```

CHAPITRE 3 : GESTION DES UTILISATEURS & GROUPES

3.1 Fichiers systèmes (CRITIQUE POUR EXAMEN)

`/etc/passwd` (lisible par tous)

Format : `login:x:UID:GID:commentaire:home:shell`

Exemple :

```
root:x:0:0:root:/root:/bin/bash
user1:x:1000:1000:User One:/home/user1:/bin/bash
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
```

`/etc/shadow` (lisible par root seul)

Format : `login:password_crypté:lastchange:minage:maxage:warning:inactivity:expiration:reserved`

```
root:$6$hash...:19000:0:99999:7::
user1:!:18000:15:90:7:30:
# "!" ou "*" = compte verrouillé/sans password
```

Champs importants :

- Champ 2 : Mot de passe crypté (! = verrouillé, * = pas de login possible)
- Champ 3 : Jours depuis 1970 du dernier changement
- Champ 4 : Jours minimum avant changement (0 = n'importe quand)
- Champ 5 : Jours maximum avant obligation de change
- Champ 6 : Jours d'avertissement avant expiration
- Champ 7 : Jours d'inactivité avant désactivation
- Champ 8 : Date expiration compte (jours depuis 1970)

`/etc/group` (lisible par tous)

Format : `groupname:x:GID:secondary_members`

```
users:x:100:user1,user2
sudo:x:27:
admins:x:1001:alice,bob
```

`/etc/gshadow` (lisible par root seul)

Format : `groupname:password:administrators:members`

```
admins:!:root:alice,bob
# "!" = pas de password/groupe verrouillé pour newgrp
```

3.2 Création d'utilisateurs & groupes

Créer un groupe

```
groupadd groupname          # Groupe simple
groupadd -g 1005 groupname  # Avec GID spécifique
```

Créer un utilisateur

```
useradd -m -c "Commentaire" -s /bin/bash username
# -m = créer home dans /home/username
# -c = commentaire (GECOS)
# -s = shell par défaut

useradd -u 1003 -g groupname -G grp2,grp3 username
# -u = UID spécifique
# -g = groupe primaire
# -G = groupes secondaires (ajout)
```

Définir mot de passe

```
passwd username
# Ou avec script :
echo "password" | passwd --stdin username
```

Politique de mot de passe

```
chage -m 15 -M 90 -W 7 -I 30 username
# -m = min days before change (15 days)
# -M = max days before must change (90 days)
# -W = warning days before expiration (7 days)
# -I = inactivity days before disable (30 days)

# Ou avec passwd :
passwd -x 60 -w 5 username  # 60 jours max, warning 5 jours avant
```

Supprimer/modifier utilisateurs

```
usermod -g newgroup username    # Changer groupe primaire
usermod -aG newgroup username   # Ajouter groupe secondaire (sans perdre les autres)
usermod -s /bin/zsh username    # Changer shell
userdel -r username             # Supprimer user + son home
groupdel groupname              # Supprimer groupe
```

3.3 Groupe privé par utilisateur (UPG)

Par défaut, lors de la création d'un utilisateur, un groupe du même nom est créé (groupe primaire).

```
useradd user1
# Crée : user1 (user) + groupe user1 (group)
```

3.4 Administrateurs de groupe (gpasswd)

```
gpasswd -A alice,bob admins     # alice et bob = admins du groupe admins
newgrp admins                    # Un membre peut changer temporairement de groupe primaire
```

3.5 Compte verrouillé & déverrouillé

```
# Vérifier : /etc/shadow a "!" au champ du password
# User ne peut pas se connecter
```

CHAPITRE 4 : GESTION DES DISQUES

4.1 Inode (très important)

Structure de données contenant les métadonnées d'un fichier :

- Numéro d'inode unique
- Propriétaire, groupe, droits
- Taille en octets
- Dates (création, modification)
- **Nombre de liens physiques**

```
ls -i fichier           # Afficher n° inode
stat fichier            # Détails complets incluant inode
df -i                   # Afficher inodes disponibles/utilisés
```

Important : Le nombre d'inodes max est **défini au formatage** → limite du nombre de fichiers!

4.2 Partitionnement : MBR vs GPT

Critère	MBR	GPT
Partitions max	4 primaires	128+ partitions
Taille max partition	2,2 To	9,4 Zettaoctets

Critère	MBR	GPT
Adressage	CHS (historique)	LBA (logique)
Table de partitions	Début du disque	Début + fin du disque (backup)
Mode firmware	BIOS	UEFI

Décision :

- **< 2,2 To + ≤ 4 partitions** → MBR acceptable
- **> 2,2 To OU > 4 partitions → GPT obligatoire**

4.3 Structure GPT (TRÈS IMPORTANT EXAMEN)

```
LBA 0        : Protective MBR (protection contre outils anciens)
LBA 1-33     : En-tête GPT primaire + table partitions
LBA 34+      : Données utilisateur
LBA fin-33   : Table partitions (sauvegarde)
LBA fin      : En-tête GPT (sauvegarde)
```

Outils GPT :

```
gdisk -l /dev/sdb              # Afficher partitions GPT
gdisk /dev/sdb                 # Interactif (créer/modifier)
```

4.4 Nommage des disques sous Linux

- **IDE** : `hda` , `hdb` , `hdc` , `hdd`
- **SATA/SCSI/USB** : `sda` , `sdb` , `sdc` , ...
- **Partitions** : `sda1` , `sda2` (numéro ajouté)

```
fdisk -l /dev/sda              # Lister partitions
```

4.5 Systèmes de fichiers

Type	Caractéristiques
ext2	Ancien, aucun journal
ext3	ext2 + journal
ext4	ext3 amélioré, moderne
XFS	Journalisé, gros fichiers (9 Exabytes+)
ReiserFS	Journalisé, petits fichiers
FAT32 / NTFS	Windows

Journalisation : Trace les opérations en cours pour récupération en cas d'arrêt brutal.

4.6 Créer/formater partitions

```
# Créer partition GPT
gdisk /dev/sdb
# Taper : n → (entrées) → w

# Formater
mkfs.ext4 /dev/sdb1             # ext4
mkfs.ext3 /dev/sdb1             # ext3
mke2fs -t ext2 /dev/sdb1        # ext2 (ancienne méthode)
mkfs -t XFS /dev/sdb1           # XFS
mkfs.vfat /dev/sdb1             # FAT32

# Vérifier/réparer
fsck /dev/sdb1                  # À faire sur partition non montée!
fsck -i /dev/sdb1               # Réparer interactivement
```

4.7 Montage temporaire vs permanent

Montage temporaire

```
mkdir -p /mnt/partition1
mount /dev/sdb1 /mnt/partition1
mount                           # Afficher tous les montages
df -h                           # Statistiques (disques montés)
umount /mnt/partition1          # Démonter
```

Montage permanent : `/etc/fstab`

Format : `device mount_point fstype options dump pass`

```
# /etc/fstab
/dev/sdb1       /mnt/data   ext4    defaults    0   0
UUID=xxx        /home       ext4    defaults    0   1
/dev/sdb2       /backup     ext3    rw,noauto   0   0
```

Signification des colonnes :

1. Partition (`/dev/sdb1` ou `UUID=...`)
2. Point de montage (doit exister)
3. Type de FS
4. Options : `ro` (lecture seule), `rw` (lecture-écriture), `auto` , `noauto` , `user` , `defaults`
5. Dump (0 = pas de sauvegarde)
6. Pass (ordre vérification au boot : 0 = non, 1 = root, 2+ = autres)

Après modification :

```
mount -a                        # Monter tous les FS de /etc/fstab
```

4.8 LVM (Logical Volume Manager)

Concept : Abstraction entre les partitions physiques et logiques.

Hiérarchie :

```
PV (Physical Volume) ← partitions réelles
    ↓
VG (Volume Group) ← groupe de PV
    ↓
LV (Logical Volume) ← partitions logiques (utilisables)
    ↓
Système de fichiers
```

Commandes LVM

```
# 1. Installer
apt install lvm2

# 2. Créer volumes physiques
pvcreate /dev/sdb /dev/sdc
pvcreate /dev/sdd

# 3. Créer groupe de volumes
vgcreate vg1 /dev/sdb /dev/sdc
vgcreate VG_exam /dev/sdd

# 4. Créer volumes logiques
lvcreate -n lv1 -L 4G vg1       # 4 Go
lvcreate -n lv_exam -L 5G VG_exam

# 5. Vérifier
pvdisplay                       # Détails PV
vgdisplay vg1                   # Détails VG
lvscan                          # Lister tous les LV

# 6. Formater + monter
mkfs.ext4 /dev/vg1/lv1
mount /dev/vg1/lv1 /mnt/data

# 7. Montage permanent dans /etc/fstab
/dev/vg1/lv1    /mnt/data   ext4    defaults    0   0
```

4.9 Quotas disque

Limiter l'utilisation disque par utilisateur ou groupe.

Deux limites :

- **Soft** : Limite douce (warning, mais peut dépasser)
- **Hard** : Limite dure (impossible de dépasser)

Mettre en place les quotas

```
# 1. Ajouter dans /etc/fstab
/dev/sdb1       /data   ext4    defaults,usrquota,grpquota   0   0

# 2. Remonter
mount -o remount /data

# 3. Installer les quotas
quotacheck -a                   # Générer fichiers quota
quotaon /data                   # Activer

# 4. Éditer les quotas utilisateur
edquota user1
# Modifie 2 limites : blocs (espace) et inodes (nombre de fichiers)
# blocks soft=1000 hard=2000 (en blocs de 1K par défaut)
# inodes soft=100 hard=150

# 5. Éditer période de grâce (durée avant que soft devienne hard)
edquota -T user1

# 6. Quotas de groupe
edquota -g admins
```

Exemple : user1 peut créer max 20 fichiers, avec 3 jours de grâce pour 5 fichiers supplémentaires.

CHAPITRE 5 : REDIRECTION E/S & TUBES

5.1 Redirection standard

Notation	Signification
0	Entrée standard (stdin)
1	Sortie standard (stdout)
2	Sortie d'erreur (stderr)

```
ls > result.txt             # Sortie → fichier (écrase)
ls >> result.txt            # Sortie → fichier (ajoute)
ls 2> errors.txt            # Erreurs → fichier
ls 1>out.txt 2>err.txt      # Sortie + erreurs séparées
ls &> all.txt               # Tout dans un fichier
command < input.txt         # Entrée depuis fichier
command > /dev/null 2>&1    # Tout vers néant (discarter)
```

5.2 Pipes (tubes)

La sortie d'une commande devient l'entrée d'une autre.

```
cat /etc/passwd | grep user1       # Chercher "user1" dans passwd
ls -la | wc -l                     # Compter les lignes
ls | sort | uniq                   # Trier puis supprimer doublons
```

5.3 Commandes filtres essentielles

find

```
find /home -type f -name "*.txt"            # Fichiers .txt
find /etc -type d -perm 755                 # Répertoires avec 755
find / -type f -size +100M                  # Fichiers > 100 Mo
find / -type f -name "*.log" -mtime -7      # Fichiers modifiés < 7 jours
find /usr -type f -size +5M                 # Fichiers > 5 Mo
```

Wildcards :

- `*` = toute chaîne (même vide)
- `?` = un caractère quelconque
- `[abc]` = l'un de a, b, c
- `[!abc]` = n'importe quoi sauf a, b, c

grep

```
grep "pattern" file                # Chercher pattern
grep -i "pattern" file             # Case insensitive
grep -v "pattern" file             # Lignes ne contenant pas pattern
grep -n "pattern" file             # Avec numéros de ligne
grep -r "pattern" /dir             # Récursif
grep -c "pattern" file             # Compter occurrences
```

cut

```
cut -d: -f1 /etc/passwd            # 1er champ séparé par ":"
cut -d: -f1,6,7 /etc/passwd        # Champs 1, 6, 7
cut -c1-5 file                     # Caractères 1 à 5
echo "12345" | cut -c2-4           # Résultat: "234"
```

sort

```
sort file                           # Trier alphabétiquement
sort -n file                        # Trier numériquement
sort -r file                        # Trier inversé
sort -t: -k3 file                   # Trier par le 3e champ (sep ":")
sort -t: -k3n file                  # Trier numériquement par 3e champ
```

wc

```
wc -l file                          # Compter lignes
wc -w file                          # Compter mots
wc -c file                          # Compter caractères
wc -l < file                        # Compter lignes (stdin)
```

head / tail

```
head -10 file                       # 10 premières lignes
tail -10 file                       # 10 dernières lignes
tail -f /var/log/syslog             # Suivre en direct (logs)
```

CHAPITRE 6 : SCRIPTS SHELL

6.1 Structure d'un script

```
#!/bin/bash                         # Shebang (doit être 1ère ligne)

# Variables
nom="Jean"
variable=$(commande)                # Assignation résultat commande

# Affichage
echo "Bonjour $nom"                 # Entre guillemets : interprète variables
echo 'Hello $nom'                   # Entre quotes : littéral
echo -e "Ligne1\nLigne2"            # -e pour newlines

# Saisie utilisateur
read -p "Votre nom: " nom           # -p pour prompt
read var1 var2                      # 2 variables

# Exécution
chmod +x script.sh
./script.sh
bash script.sh
```

6.2 Paramètres positionnels

```
# Lancer : ./script.sh arg1 arg2 arg3

$0              # Nom du script
$1, $2, ...     # Arguments 1, 2, ...
$#              # Nombre total d'arguments
$*              # Tous les arguments
```

6.3 Structures conditionnelles (if)

```
if [ condition ]; then
    commandes
elif [ autre_condition ]; then
    commandes
else
    commandes
fi

# Conditions sur fichiers :
test -e /etc/passwd && echo "existe"    # -e : existe
test -f /etc/passwd && echo "fichier"   # -f : fichier ordinaire
test -d /etc && echo "dossier"          # -d : répertoire
test -r /etc/passwd && echo "lisible"   # -r : readable
test -w fichier && echo "writable"      # -w : writable
test -x script && echo "executable"     # -x : executable

# Conditions sur nombres :
[ $num -eq 5 ]      # Égal
[ $num -ne 5 ]      # Différent
[ $num -lt 5 ]      # Inférieur à
[ $num -gt 5 ]      # Supérieur à
[ $num -le 5 ]      # ≤
[ $num -ge 5 ]      # ≥

# Conditions sur strings :
[ -z "$str" ]       # String vide
[ -n "$str" ]       # String non vide
[ "$str1" = "$str2" ]   # Égal
[ "$str1" != "$str2" ]  # Différent

# Combinaisons :
[ -f file ] && [ -x file ]      # AND (&&)
[ -f file ] || [ -d file ]      # OR (||)
[ ! -f file ]                   # NOT (!)
```

6.4 Boucles

for

```
for var in liste; do
    echo $var
done

# Exemple :
for i in 1 2 3 4 5; do echo $i; done
for file in *.txt; do cat $file; done
for i in $(seq 1 10); do echo $i; done
```

while / until

```
while condition; do
    commandes
done

until condition; do     # Inverse de while
    commandes
done
```

6.5 Fonctions

```
function myfunction() {
    echo "Argument 1: $1"
    return 0
}

# Ou :
myfunction() {
    echo "Hello"
    return 0
}

# Appel :
myfunction arg1 arg2
echo $?          # Code de retour (0 = succès)
```

CHAPITRE 7 : SÉQUENCE DE DÉMARRAGE (UEFI/GRUB)

7.1 BIOS vs UEFI

Aspect	BIOS	UEFI
Firmware	Ancien (années 90)	Moderne (2006+)

Aspect	BIOS	UEFI
Partition max	MBR = 2,2 To	GPT = 9,4 Zo
Partitions	Max 4 primaires	Max 128+
Espace	1 Mo	Extensible
Secure Boot	Non	Oui
Démarrage	Lent	Rapide

7.2 UEFI & NVRAM

UEFI : Firmware moderne stocké dans une mémoire flash.

NVRAM : Non-Volatile RAM qui stocke les variables de démarrage entre les redémarrages.

```
efibootmgr -v                       # Afficher entrées de boot EFI
efibootmgr -o 0005,0004,0000        # Changer l'ordre de boot
```

Données NVRAM :

- `BootCurrent` : Entrée actuellement utilisée
- `BootOrder` : Ordre de recherche
- `Boot####` : Entrées disponibles (#### = hex)

7.3 Structure GPT (rappel)

```
LBA 0       : Protective MBR
LBA 1-33    : En-tête GPT + table des partitions
LBA 34+     : Données
Fin-33      : Sauvegarde table
Fin         : Sauvegarde en-tête
```

Outils :

```
gdisk -l /dev/sda               # Afficher partitions GPT
gdisk /dev/sda                  # Éditer interactif
```

7.4 EFI System Partition (ESP)

Partition spéciale qui contient **les fichiers de démarrage** :

- `BOOTX64.EFI` : Boot manager
- `shimx64.efi` : Chargeur avec Secure Boot
- `grubx64.efi` : **Chargeur de démarrage principal** ← TRÈS IMPORTANT
- `vmlinuz` : Noyau Linux
- `initramfs` : Système de fichiers initial

GUID ESP : `C12A7328-F81F-11D2-BA4B-00A0C93EC93B`

7.5 Séquence de démarrage complète

1. **UEFI POST** : Autotest matériel
2. **UEFI Boot** : Cherche ESP dans NVRAM (BootOrder)
3. **BOOTX64.EFI** : Lance le Boot Manager
4. **GRUBX64.EFI** : Chargeur de démarrage (ou shimx64.efi si Secure Boot)
5. **GRUB Menu** : Menu de sélection du noyau
6. **Kernel (vmlinuz)** : Noyau chargé en mémoire
7. **initramfs** : Système de fichiers initial temporaire
8. **Kernel init** : Lance le premier processus (systemd)

7.6 GRUB2 Configuration

Fichier principal : `/etc/default/grub`

```
GRUB_DEFAULT=0                      # Entrée par défaut (index)
GRUB_TIMEOUT_STYLE=menu             # ou "hidden"
GRUB_TIMEOUT=5                      # Secondes d'attente
GRUB_DISTRIBUTOR="Ubuntu"           # Nom affiche dans le menu
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash"  # Options noyau
```

Paramètres importants :

- `GRUB_DEFAULT=saved` : Redémarre sur le dernier noyau utilisé (fallback auto)
- `GRUB_TIMEOUT=0` : Aucune attente, boot automatique
- `GRUB_TIMEOUT_STYLE=hidden` : Menu masqué (appuyer Shift pour voir)

Ajouter entrées personnalisées : `/etc/grub.d/40_custom`

```
#!/bin/sh -e
echo "Adding custom menu entries"

menuentry "Arrêter le système" {
    halt
}

menuentry "Redémarrer le système" {
    reboot
}
```

Appliquer les modifications

```
sudo update-grub                    # OBLIGATOIRE après édition
# Met à jour /boot/grub/grub.cfg
```

Sécuriser GRUB par mot de passe

```
grub-mkpasswd-pbkdf2                # Générer password chiffré
# → Copier le hash généré

# Ajouter dans /etc/grub.d/40_custom ou nouveau fichier :
set superusers="root"
password_pbkdf2 root grub.pbkdf2.sha512.10000.LONG_HASH_HERE...

# Puis : sudo update-grub
```

CHAPITRE 8 : SYSTEMD

8.1 Unités Systemd

Fichiers de configuration :

- `/usr/lib/systemd/system/*.service` : Unités fournies par le système
- `/etc/systemd/system/*.service` : Unités personnalisées admin

8.2 Structure d'une unité `.service`

```
[Unit]
Description=Description du service
After=network.target database.service
Before=other.service
Requires=mandatory.service
Wants=optional.service
Conflicts=incompatible.service

[Service]
Type=simple
ExecStart=/usr/bin/monscript
ExecStop=/usr/bin/stop-script
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Sections :

[Unit] :

- `Description=` : Texte descriptif
- `After=target1 service2` : Démarre **après** ces unités (ordre, pas dépendance)
- `Before=target1` : Démarre **avant**
- `Requires=service` : Dépendance **forte** (si échoue, cette unité échoue)
- `Wants=service` : Dépendance **faible** (si échoue, continue quand même)
- `Conflicts=service` : Ne peut pas démarrer en même temps

[Service] :

- `ExecStart=/chemin/script` : Commande au démarrage
- `ExecStop=/chemin/stop` : Commande à l'arrêt
- `Type=simple | oneshot` : Type de service

[Install] :

- `WantedBy=multi-user.target` : Activé par cette cible
- `RequiredBy=target` : Requis par cette cible

8.3 Cibles (Targets)

Les **targets** remplacent les anciens runlevels.

Target	Ancien	Fonction
`multi-user.target`	Runlevel 3	Multiutilisateurs, sans GUI
`graphical.target`	Runlevel 5	Multiutilisateurs + GUI
`rescue.target`	Runlevel S	Mode de récupération
`poweroff.target`	Runlevel 0	Arrêt système
`reboot.target`	Runlevel 6	Redémarrage

8.4 Commandes Systemctl

```
# Gestion services
systemctl start nom.service          # Lancer
systemctl stop nom.service           # Arrêter
systemctl restart nom.service        # Relancer
systemctl status nom.service         # État
systemctl enable nom.service         # Activer au démarrage
systemctl disable nom.service        # Désactiver au démarrage
systemctl is-enabled nom.service     # Vérifier activation

# Recharger après modification
sudo systemctl daemon-reload

# Gestion targets
systemctl set-default graphical.target      # Cible par défaut
systemctl get-default                       # Afficher cible par défaut
```

```
systemctl isolate multi-user.target         # Basculer temporairement

# Arrêt système
systemctl poweroff                    # Éteindre
systemctl reboot                      # Redémarrer
systemctl halt                        # Arrêter (pas éteint)
```

8.5 Dépendances : After vs Requires vs Wants

Différence critique pour examen :

- **After/Before** : **Ordre seulement**
  - Service A `After=B` : A démarre après B, mais B peut avoir échoué
  - **Pas de dépendance fonctionnelle**
- **Requires** : **Dépendance forte**
  - Service A `Requires=B` : Si B échoue, A échoue aussi
  - A ne peut démarrer que si B a réussi
- **Wants** : **Dépendance faible**
  - Service A `Wants=B` : B est souhaité, mais A démarre même si B échoue

Exemple :

```
[Unit]
After=network.target                # Ordre de démarrage
Requires=database.service           # Dépendance fonctionnelle
Wants=backup.service                # Optionnel
```

8.6 Exemple complet : service montage-partition

```
[Unit]
Description=Service de montage de /dev/sda3
Requires=local-fs.target            # local-fs DOIT réussir
After=dev-sda3.device local-fs.target   # Ordre de démarrage
Before=multi-user.target            # Doit finir avant multi-user
Conflicts=umount.target             # Incompatible avec umount

[Service]
ExecStart=mount /dev/sda3 /pm2
Type=oneshot                         # Service "ponctuel" (pas daemon)
RemainAfterExit=yes                  # Reste actif après finish

[Install]
WantedBy=multi-user.target          # Activé par multi-user
```

CHAPITRE 9 : CRON (Planification de tâches)

9.1 Crontab

Planifier l'exécution périodique de commandes.

```
crontab -e                          # Éditer mes crontabs
crontab -l                          # Lister mes crontabs
crontab -r                          # Supprimer tous mes crontabs
```

9.2 Format crontab

```
mm hh jj MM JJ  user  commande  >> log 2>&1
|  |  |  |  |    |     |
|  |  |  |  |    |     └─ Commande à exécuter
|  |  |  |  |    └─────── Utilisateur (seulement dans /etc/crontab)
|  |  |  |  └─ Jour semaine (0=dim, 1=lun, ..., 7=dim)
|  |  |  └──── Mois (1-12)
|  |  └─────── Jour du mois (1-31)
|  └────────── Heure (0-23)
└──────────── Minute (0-59)
```

Exemple :

```
0 2 * * *   /usr/bin/backup.sh       # Chaque jour à 02:00
*/5 * * * *  /usr/bin/check.sh       # Tous les 5 minutes
0 10 * * 1-5 /usr/bin/work.sh        # Lun-Ven à 10:00
0 */6 * * *  /usr/bin/regular.sh     # Tous les 6 heures
30 23 31 12 * /usr/bin/newyear.sh    # 31 déc à 23:30
```

POINTS CRITIQUES POUR LUNDI

Obligatoire à maîtriser (90% de certitude à l'examen)

1. **MBR vs GPT** : Différences, limites, quand choisir
2. **Permissions octal** : 755, 644, 777, 4755, 2755, 1777
3. **SGID sur répertoire** : Fichiers héritent du groupe du dossier
4. **Sticky bit** : Seul le propriétaire peut supprimer son fichier
5. **/etc/passwd & /etc/shadow** : Format, champs, signification
6. **useradd/usermod/chage** : Créer, modifier, politique mots de passe

7. **chmod/chown/chgrp** : Changer droits, propriétaire, groupe
8. **find/grep/cut/sort** : Les 4 piliers des filtres
9. **LVM** : PV → VG → LV, commandes créer
10. **Montage /etc/fstab** : Format, options, permanence
11. **GRUB2** : `/etc/default/grub`, `40_custom`, `update-grub`
12. **Systemd** : Services, After/Requires/Wants, targets, `systemctl`
13. **UEFI/GPT** : Structure, ESP, efibootmgr
14. **Redirection E/S & pipes** : `>`, `>>`, `2>`, `|`, `&>`

Très probables (60-80% de certitude)

15. **umask** : Calcul permissions par défaut
16. **Quotas** : Soft/hard, limits inodes/blocs
17. **Scripts shell** : Variables, if, for, fonctions
18. **Cron** : Format, exemples simples

Possibles mais moins (30-50%)

19. **inode** : Concept, liens physiques vs symboliques
20. **Systèmes de fichiers** : Types, journalisation

CONSEIL FINAL

Lundi, l'examen testera surtout votre capacité à :

1. **Lire et interpréter des fichiers système** (`/etc/passwd`, `/etc/fstab`, etc.)
2. **Écrire des commandes correctes** (surtout avec options)
3. **Résoudre des cas pratiques** (créer user + groupe + permissions + quotas)
4. **Comprendre les dépendances systemd**
5. **Maîtriser les concepts** (pas juste les commandes)

Stratégies :

- Lisez bien les questions, elles donnent des indices
- Si c'est un sujet sur disques → pense à MBR/GPT/LVM/montage
- Si c'est sur utilisateurs → pense fichiers systèmes + permissions
- Si c'est sur systemd → pense After vs Requires
- Si c'est sur GRUB → pense `/etc/default/grub` + `update-grub`

À éviter :

- Confondre `chmod 755` et `chmod a+x`
- Oublier que SGID s'applique sur un **répertoire** pas un fichier

- Mélanger soft/hard limits de quotas
- Oublier de faire `update-grub` après édition GRUB
- Confondre After (ordre) et Requires (dépendance)

Bon courage ! Tu vas cartonner !