04 - Noyau et périphériques

Objectif

- Expliquer comment Linux représente le matériel, charge les bons composants noyau et permet à l'administrateur d'ajuster le comportement du système.

1. `/dev` : les périphériques comme fichiers

> Sous Linux, un périphérique est exposé sous forme de fichier spécial.

1. `/dev` : les périphériques comme fichiers

- Le répertoire `/dev` contient des fichiers spéciaux qui représentent des périphériques réels ou virtuels : disques, partitions, terminaux, générateurs aléatoires, périphériques loop, etc.
- Exemple :

```
ls -l /dev/sd*
```

- On observe des fichiers de type bloc, par exemple :

```
brw-rw---- 1 root disk 8, 0 /dev/sda
brw-rw---- 1 root disk 8, 1 /dev/sda1
```

- Le `b` indique un périphérique bloc.
- Les numéros majeur et mineur identifient le pilote et l'instance du périphérique.

1. `/dev` : les périphériques comme fichiers — Pseudo-périphériques utiles

```
/dev/null      trou noir à octets
/dev/zero      génère des zéros
/dev/random    génère de l'aléatoire
/dev/loop0     périphérique bloc basé sur un fichier
/dev/ttyS0     port série
/dev/lp0       port parallèle / imprimante
```

2. udev

> udev crée et gère dynamiquement les entrées de `/dev` en fonction du matériel détecté.

2. udev

- Avant udev, les systèmes pouvaient avoir une liste statique de nombreux périphériques. udev fonctionne en espace utilisateur et réagit aux événements envoyés par le noyau lorsqu'un périphérique apparaît ou disparaît.
- udev peut :
- créer un nom de périphérique ;
- créer des liens symboliques stables ;
- ajuster les permissions ;
- déclencher une action ;
- charger certains firmwares lorsque nécessaire.

2. udev — Commandes de démonstration

```
udevadm info --query=all --name=/dev/sda
udevadm info --attribute-walk --name=/dev/sda
udevadm monitor
```

3. Règles udev

> Une règle udev associe des attributs matériels à une action ou un nom stable.

3. Règles udev — Exemple générique

```
SUBSYSTEM=="block", ATTRS{serial}=="XXXX", SYMLINK+="disk/backup_usb"
```

3. Règles udev

- On utilise les attributs remontés par `udevadm info --attribute-walk`.
- Il faut choisir des attributs suffisamment stables : numéro de série, fabricant, modèle, chemin matériel selon le besoin.

3. Règles udev — Point d'attention

- Une règle trop générale peut s'appliquer à plusieurs périphériques.
- Une règle trop spécifique peut casser au changement de matériel.

4. Compiler un noyau personnalisé

> Compiler un noyau permet d'adapter le système, mais ce n'est plus un réflexe quotidien grâce au noyau modulaire.

4. Compiler un noyau personnalisé — Pourquoi recompiler ?

- activer ou désactiver une fonctionnalité ;
- intégrer un patch ;
- réduire la surface ou la taille du noyau ;
- supporter un matériel particulier ;
- expérimenter ou apprendre.

4. Compiler un noyau personnalisé — Grandes étapes

1. Installer un environnement de compilation.
2. Récupérer les sources du noyau.
3. Nettoyer l'arbre de compilation.
4. Configurer les options.
5. Compiler le noyau et les modules.
6. Installer les modules.
7. Générer l'initramfs.
8. Mettre à jour le chargeur de démarrage.

4. Compiler un noyau personnalisé — Commandes typiques à présenter

```
make mrproper
make menuconfig
make -j$(nproc)
make modules_install
make install
```

4. Compiler un noyau personnalisé — Point d'attention

- Toujours conserver une entrée de boot fonctionnelle vers l'ancien noyau.

5. Initramfs

> L'initramfs fournit au noyau ce dont il a besoin pour trouver et monter la vraie racine.

5. Initramfs

- Au début du démarrage, le noyau n'a pas forcément accès directement au système racine : il peut avoir besoin de modules pour le stockage, LVM, RAID, chiffrement ou certains drivers.
- L'initramfs contient un mini environnement temporaire pour charger ces éléments.
- Selon les distributions, les outils diffèrent : `mkinitcpio`, `dracut`, `update-initramfs`.

5. Initramfs — Démonstration possible

```
lsinitrd
lsinitramfs /boot/initrd.img-$(uname -r)
```

6. Créer une distribution personnalisée

> Créer une distribution, ce n'est pas seulement compiler un noyau : c'est définir une politique système complète.

6. Créer une distribution personnalisée

- Le cours évoque Linux From Scratch.
- LFS donne des instructions pour construire un système Linux minimal à partir des sources.
- C'est excellent pour comprendre les briques internes, mais fastidieux pour produire une distribution maintenable.
- NuTyX est cité comme exemple de distribution liée à cette logique de construction et de gestion de paquets.

6. Créer une distribution personnalisée — À faire retenir

- Créer une distribution implique des choix sur :
- la chaîne de compilation ;
- la libc ;
- le gestionnaire de paquets ;
- l'arborescence ;
- la politique de mise à jour ;
- la sécurité ;
- les dépôts ;
- l'intégration matérielle.

7. Inventaire matériel

> Avant d'installer un driver, il faut identifier précisément le matériel.

7. Inventaire matériel — Commandes utiles

```
lscpu
lspci
lsusb
lshw
hwinfo
cat /proc/cpuinfo
```

- Pour les composants PCI :

```
lspci -nn
lspci -k
```

- `lspci -k` permet de voir quel module noyau est utilisé.

8. `/sys`

> `/sys` expose une vue structurée des périphériques et du noyau.

8. `/sys`

- `/sys` est un système de fichiers virtuel.
- Il permet d'observer les objets du noyau : bus, périphériques, classes, modules, paramètres. udev s'appuie notamment sur les informations exposées par `/sys`.

9. Drivers et modules

> Linux charge beaucoup de fonctionnalités sous forme de modules noyau.

9. Drivers et modules — Commandes utiles

```
lsmod
modinfo <module>
modprobe <module>
modprobe -r <module>
```

- La configuration de chargement peut passer par :

```
/etc/modules-load.d/
/etc/modprobe.d/
```

9. Drivers et modules — Point d'attention

- Certains modules doivent être présents dans l'initramfs pour être disponibles assez tôt au démarrage.

10. Paramètres d'amorçage du noyau

> Les paramètres de boot permettent de donner au noyau des instructions avant même que le système soit démarré.

10. Paramètres d'amorçage du noyau

- On les utilise pour contourner une détection matérielle, activer un mode de secours, désactiver temporairement une fonctionnalité, choisir une console, passer une option réseau ou déboguer.
- Les paramètres sont généralement ajoutés dans GRUB, sur la ligne qui charge le noyau.

11. sysctl

> `sysctl` permet de lire et modifier des paramètres noyau à chaud.

11. sysctl

- `sysctl` manipule les valeurs exposées dans `/proc/sys`.
- Par exemple :

```
sysctl --all
sysctl kernel.sysrq
sysctl kernel.sysrq=1
```

- Équivalent par fichier :

```
echo 1 > /proc/sys/kernel/sysrq
```

- Pour rendre un réglage persistant :

```
/etc/sysctl.d/99-sysctl.conf
```

- Exemple :

```
kernel.sysrq = 1
```

11. sysctl — Point d'attention

- Les fichiers de `/usr/lib/sysctl.d/` peuvent être remplacés par ceux de `/etc/sysctl.d/`.
- L'ordre de chargement compte : le dernier réglage appliqué gagne.