# Système de fichiers – partie 2

# Système de fichiers – partie 2

## *Plan de la phase*

*Introduction*
*Fichier logique*
*Fichier physique*
*Correspondance fichier logique – fichier physique*
*Structure d'un fichier ext2*
*Structure d'un répertoire*
*Structure d'une partition*
*VFS : virtual file system*
*Créer un système de fichiers*
*Accéder aux systèmes de fichiers*
*Validation des acquis*

# Système de fichiers – partie 2

## *Introduction*

• Ce chapitre présente les notions relatives au système de gestion de fichiers, qui a pour but la conservation des données en dehors de la mémoire centrale volatile.

• Après avoir exposé les notions importantes liées au systèmes de gestion de fichiers, nous présentons le système de gestion de fichiers Linux ext2.

• Puis nous nous intéressons au système de gestion de fichiers virtuel supporté par Linux, qui permet à ce système de s'interfacer avec d'autres type de système de gestion de fichiers tels que ceux des systèmes DOS, Mac, IBM, etc.

# Système de fichiers – partie 2

## *Fichier logique*[1]

- Le fichier logique correspond à la vue que l'utilisateur de la machine a de la conservation de ses données. Plus précisément, le fichier logique est :

• Un  type de données standard.

• Un ensemble d'enregistrements.

- Les modes d'accès les plus courants sont[2] :

• Mode d'accès séquentiel.

• Mode d'accès indexé ou encore appelé aléatoire.

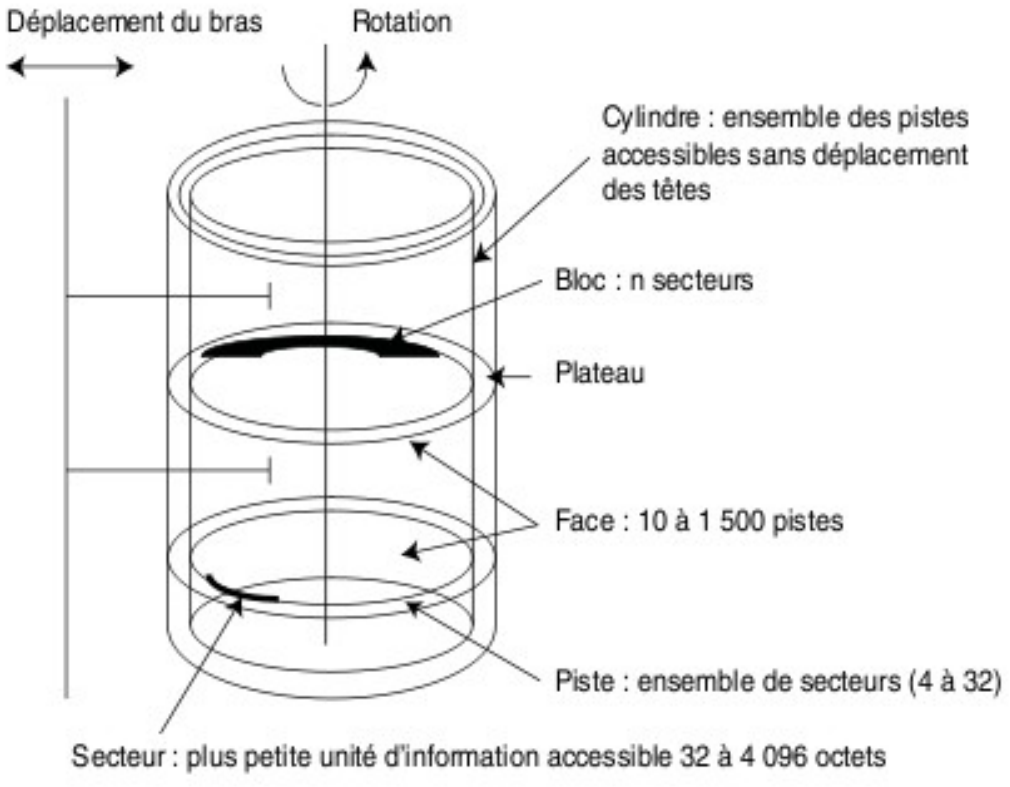
• Mode d'accès direct encore appelé accès relatif.

# Système de fichiers – partie 2

## *Fichier physique*

- Le fichier physique contient physiquement les enregistrements définis dans le fichier logique.

- Les secteurs sont regroupés en blocs.



# Système de fichiers – partie 2

## *Fichier physique*

***(b) Méthodes d'allocation de la mémoire secondaire***

• Les enregistrements sont écrits dans les secteurs des blocs du disques, pour former le fichier physique.

• Différentes méthodes d'allocation de la mémoire secondaire ont été définies, ce sont principalement :

- ➔ La méthode de l'allocation contiguë ;
- ➔ La méthode de l'allocation par zone ;
- ➔ La méthode de l'allocation par bloc chaînés ;
- ➔ La méthode de l'allocation indexée.

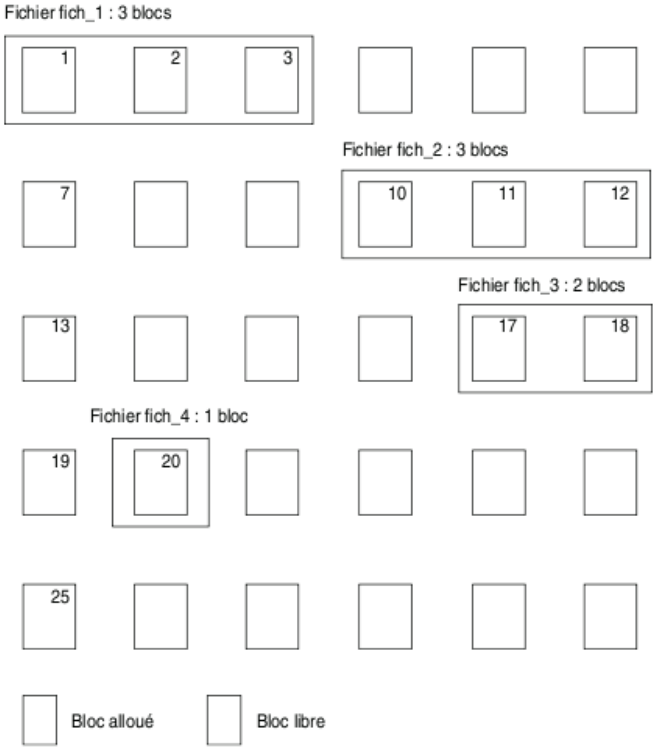
• Il faut connaître à tous moment l'ensemble des blocs libres et donc de gérer l'espace libre sur le disque.

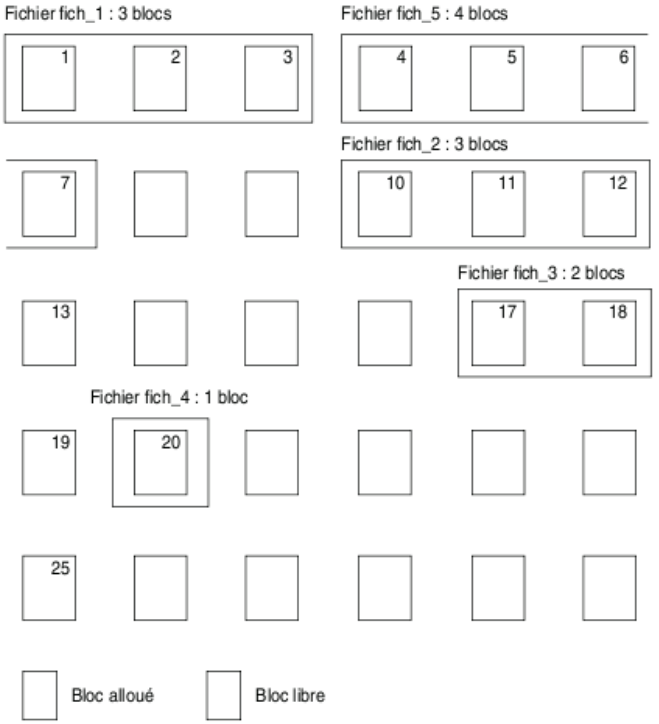
# Système de fichiers – partie 2

## *Fichier physique*

***(b) Méthodes d'allocation de la mémoire secondaire***

**La méthode de l'allocation contiguë**




- Les méthodes First Fit, Best Fit et Worst Fit sont appliquées ici pour choisir l'espace libre à allouer[1]

  ***=> problèmes de fragmentation***

- Une dernière difficulté est celle de l'extension d'un fichier si les blocs disques voisins ne sont pas libres[2].

# Système de fichiers – partie 2

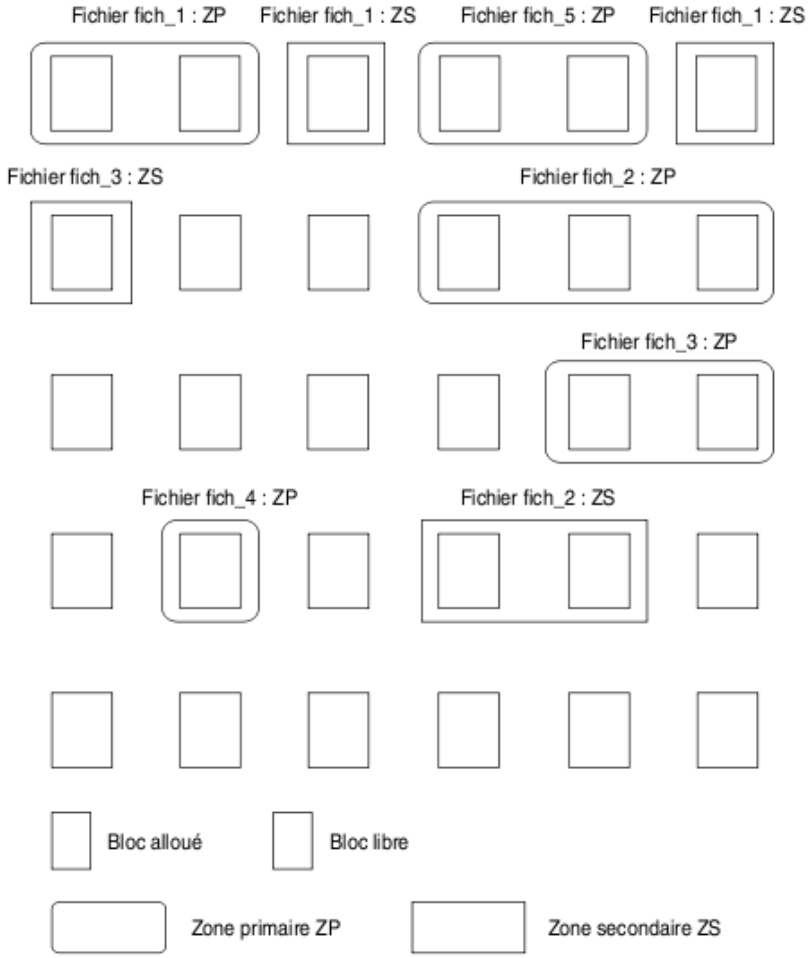
## *Fichier physique*

***(b) Méthodes d'allocation de la mémoire secondaire***

**La méthode de l'allocation par zone**

• Autorise un fichier à être constitué de plusieurs zones physiques distinctes[1].

***=> extension plus facile d'un fichier***



# Système de fichiers – partie 2

## *Fichier physique*

***(b) Méthodes d'allocation de la mémoire secondaire***

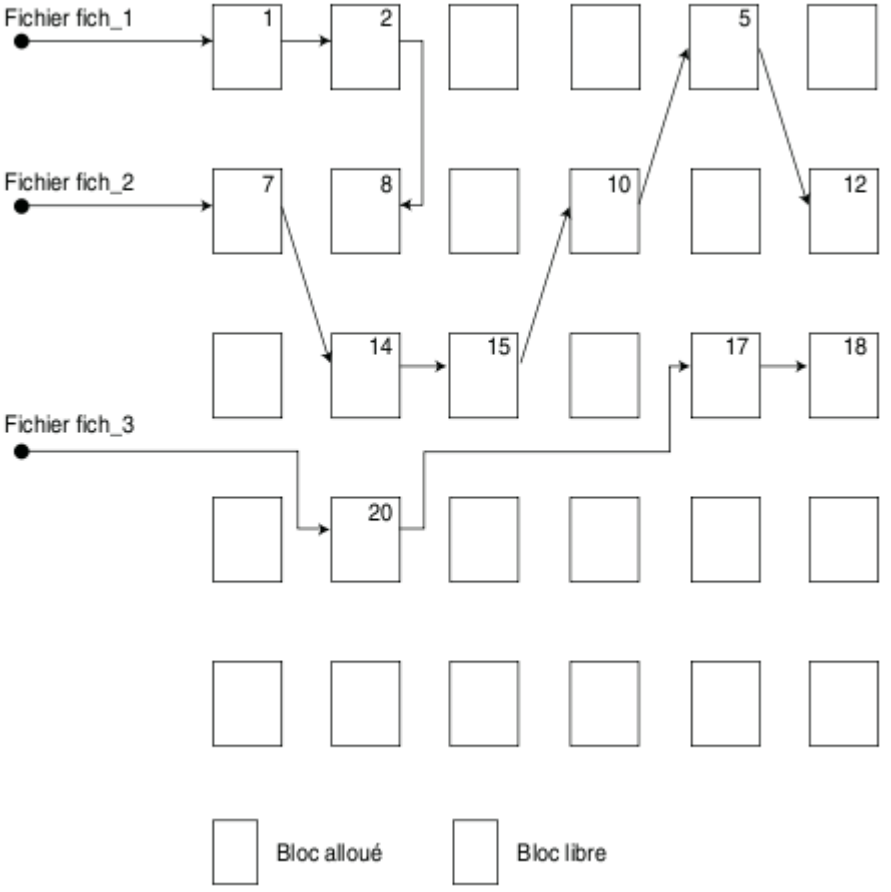
**<u>La méthode de l'allocation par bloc chainés</u>**

• Un fichier est constitué comme une liste chaînée de blocs physiques.

***=> extension plus facile d'un fichier***

• Plus nécessaire de connaître à la création du fichier, la taille maximale de celui-ci.
• Les problèmes de fragmentation externe disparaissent.

***Deux inconvénients :***
• mode d'accès séquentiel
• la place occupée dans chaque bloc par le chaînage de la liste.



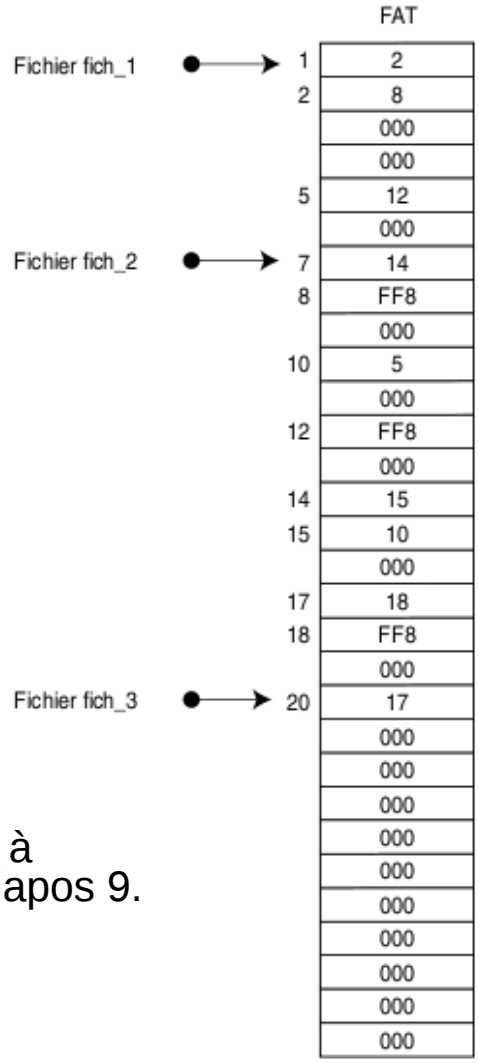
# Système de fichiers – partie 2

## *Fichier physique*

***(b) Méthodes d'allocation de la mémoire secondaire***

**<u>La méthode de l'allocation par bloc chainés</u>**

•Les systèmes de fichiers FAT (FAT16 et FAT32) est une variante de l'allocation par blocs chaînés.
• Blocs (Clusters) regroupé dans une table appelée FAT (*File Allocation Table*).
•Table : autant d'entrées qu'il y a de blocs de données sur le disque.




• La FAT 16 numéro de bloc est codé sur 16 bits autorisant ainsi 65 536 blocs sur le disque
• La FAT 32 numéro de bloc codé sur 32 bits autorisant ainsi 268 435 455 ($2^{28}$-1) Gigablocs sur le disque :
=> Pour une taille de cluster de 32 Ko, on a une taille de partition de 8 To (théorique) mais limitée à 2 To.

La figure donne la FAT correspondant à l'allocation chaînée de la figure de la diapos 9.

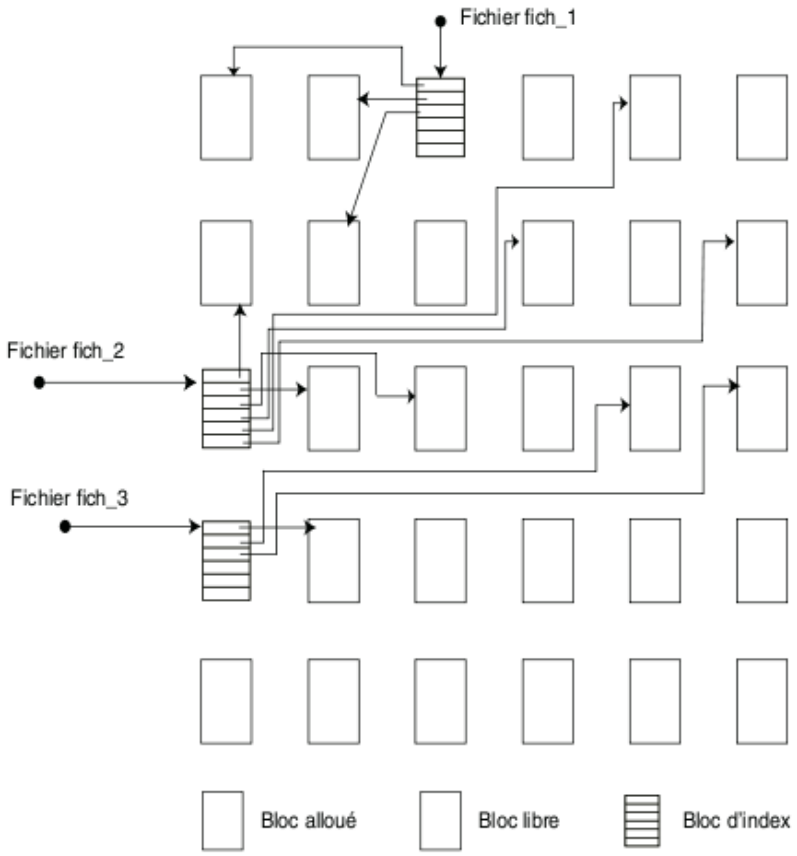
# Système de fichiers – partie 2

## *Fichier physique*

***(c) Allocation indexée***

• Table appelée *index.*

• Accès direct à chacun de ces blocs.




***Problèmes*** :

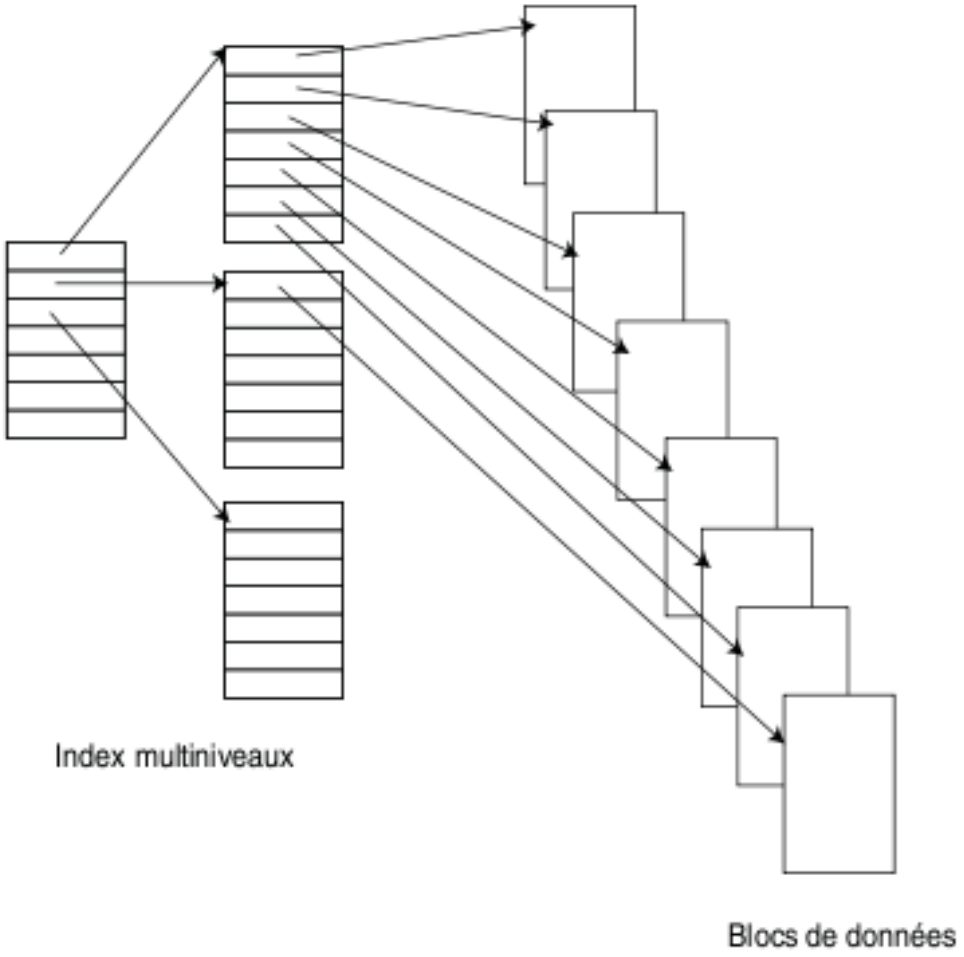
• La taille de la table d'index est conditionnée par celle d'un bloc physique et par le nombre de blocs existants sur le disque.

# Système de fichiers – partie 2

## *Fichier physique*

***(c) Allocation indexée***

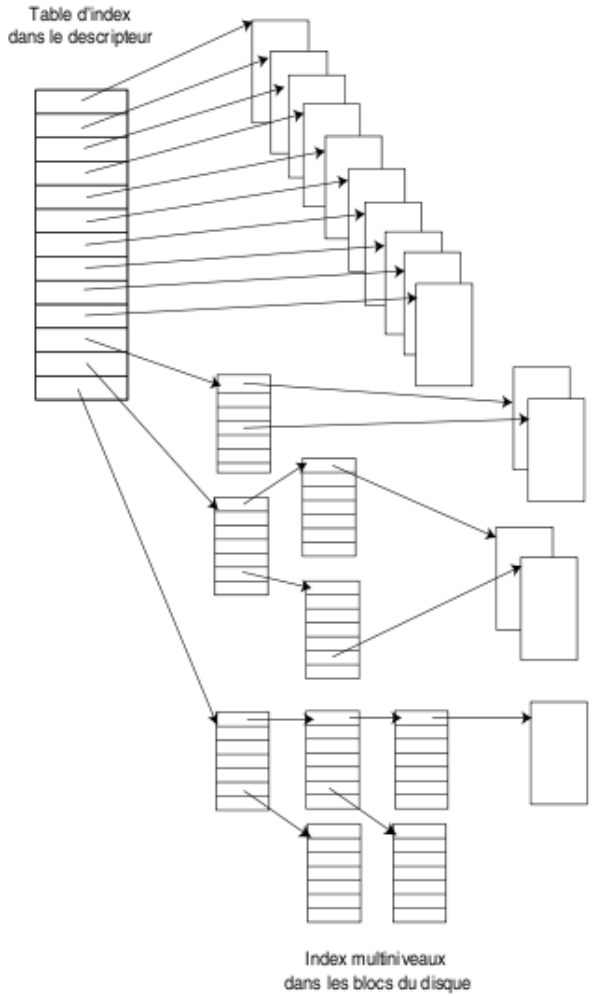
- Un index multiniveaux.



# Système de fichiers – partie 2

## *Fichier physique*

- ***(c) Allocation indexée***



# Système de fichiers – partie 2

## *Fichier physique*

***(d) Gestion de l'espace libre***

- Le système maintient une liste d'espace libre, qui mémorise tous les blocs disque libres.
- Il existe différentes représentation possibles de l'espace libre les principales sont :
  - ➔ représentation de l'espace libre sous forme d'un vecteur de bits.
  - ➔ représentation de l'espace libre sous forme d'une liste chaînée des blocs libres.

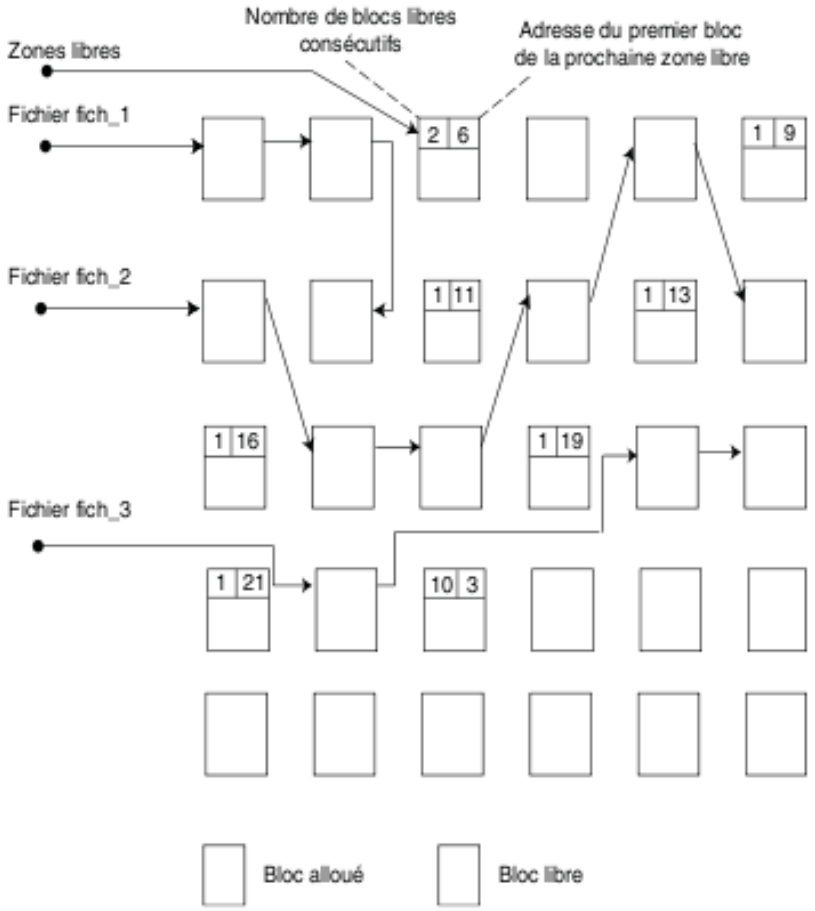
- *Gestion de l'espace libre par un vecteur de bits*
  - ➔ chaque bloc est figuré par un bit.
  - ➔ la longueur de la chaîne binaire est égale au nombre de blocs existant sur le disque (ou partition).
  - ➔ un bit à 0 = bloc libre, un bit à 1 = bloc alloué.

# Système de fichiers – partie 2

## *Fichier physique*

***(d) Gestion de l'espace libre***

- *Gestion de l'espace libre par liste chaînée*
  - ➔ L'espace libre sur le disque est représenté par une liste chaînée de l'ensemble des blocs libres du disque.



# Système de fichiers – partie 2

## *Correspondance fichier logique – fichier physique*

***(a) Notion de répertoire***

- Répertoire : correspondance entre fichiers logique et fichiers physiques.
- Une entrée du répertoire pour un fichier donné :
  - Le nom logique du fichier ;
  - Le type du fichier : fichier texte, objet, exécutable, fichier binaire.
  - L'adresse physique du fichier
  - La taille en octet ou en blocs du fichier ;
  - La date de création du fichier ;
  - Le nom du propriétaire du fichier ;
  - Les protections appliquées au fichier.

# Système de fichiers – partie 2

## *Correspondance fichier logique – fichier physique*

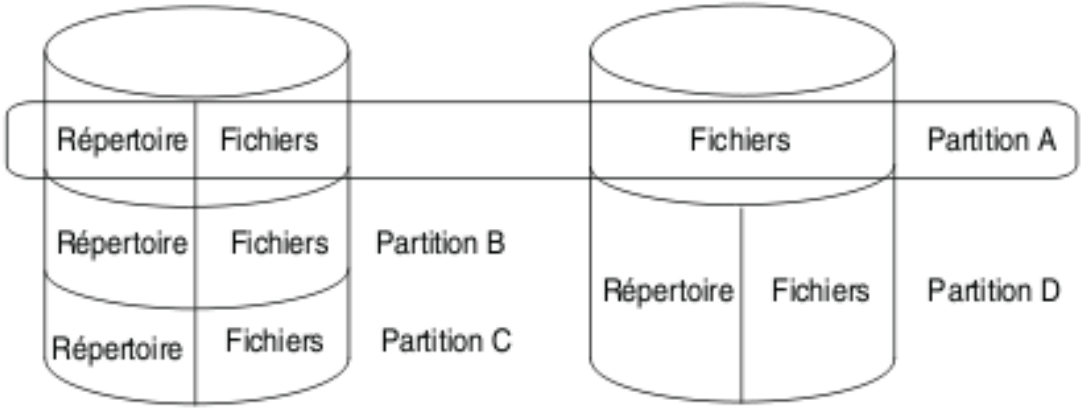
***(b) Notion de volumes ou partitions***

• Une solution couramment mise en œuvre est de diviser l'ensemble du système de gestion de fichier en morceaux indépendants appelés *volumes* ou *partitions*.

• Chaque partition  est associé un répertoire qui référence l'ensemble des fichiers présent sur la partition.

• Pour pouvoir être accessible, la partition doit être connecter à l'arborescence de fichier de la machine, en un point d'ancrage qui correspond à un répertoire.

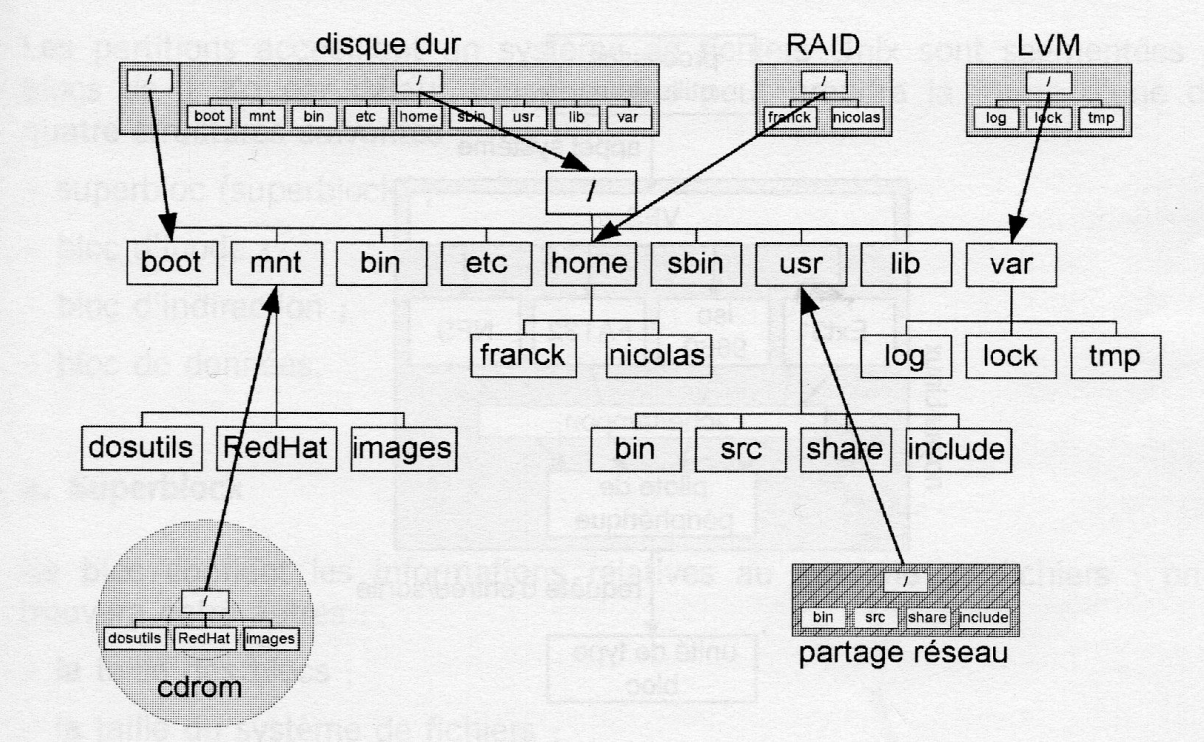
• *Opération de montage de la partition : r*endre accessible le contenu d'une partition en liant une partition à un répertoire.



# Système de fichiers – partie 2

## *Correspondance fichier logique – fichier physique*

- Document le FHS (*Filesystem Hierarchy Standard*) : http://www.pathname.com/fhs.
- Système de fichier.



# Système de fichiers – partie 2

## *Structure d'un fichier ext2*

- Le système d'exploitation Minix et système de gestion de fichier Minix.

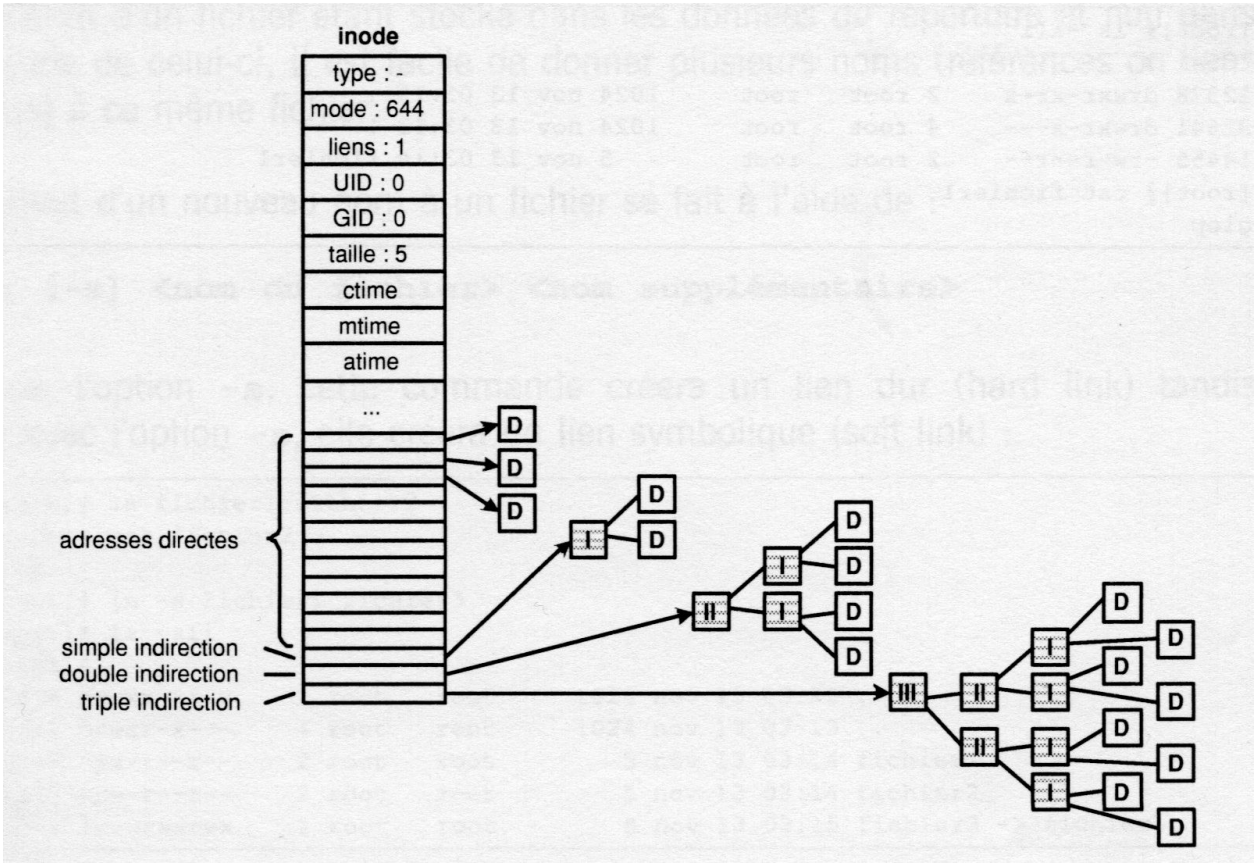
- Deux développements ont vu le jour pour répondre à ces problèmes :
  - Ext2.
  - *Virtual File System* (*VFS*) .

- *inode* ou i-nœud.
- Les blocs sont alloués au fichier selon une organisation indexée.
- La taille d'un bloc est un multiple de la taille d'un secteur : 512, 1024, 2048 et 4096.

# Système de fichiers – partie 2

## *Structure d'un fichier ext2*

***(a) L'inode***

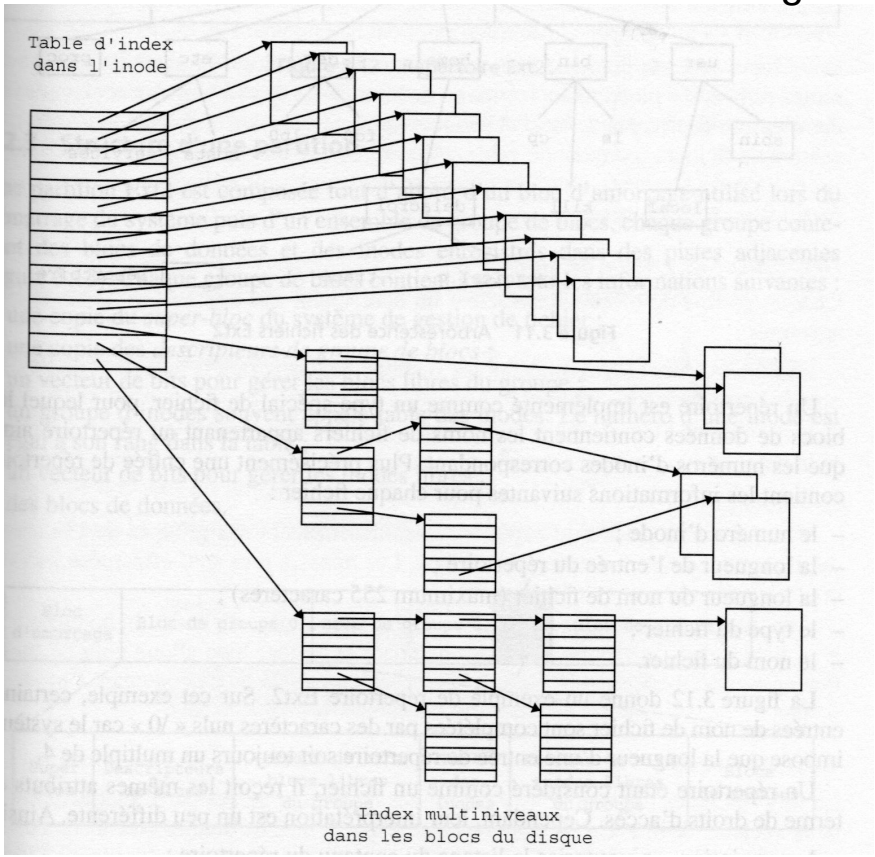


# Système de fichiers – partie 2
## *Structure d'un fichier ext2*

***(b) Allocation des blocs de données***

• Ainsi si T est la taille d'un bloc, alors la taille maximale d'un fichier en nombre de bloc est égale à : $12 + (T/4) + (T/4)^2 + (T/4)^3$ blocs.

*(rappel : Chaque bloc est identifié par un numéro logique codé sur 4 octets )*



# Système de fichiers – partie 2

## *Structure d'un fichier ext2*

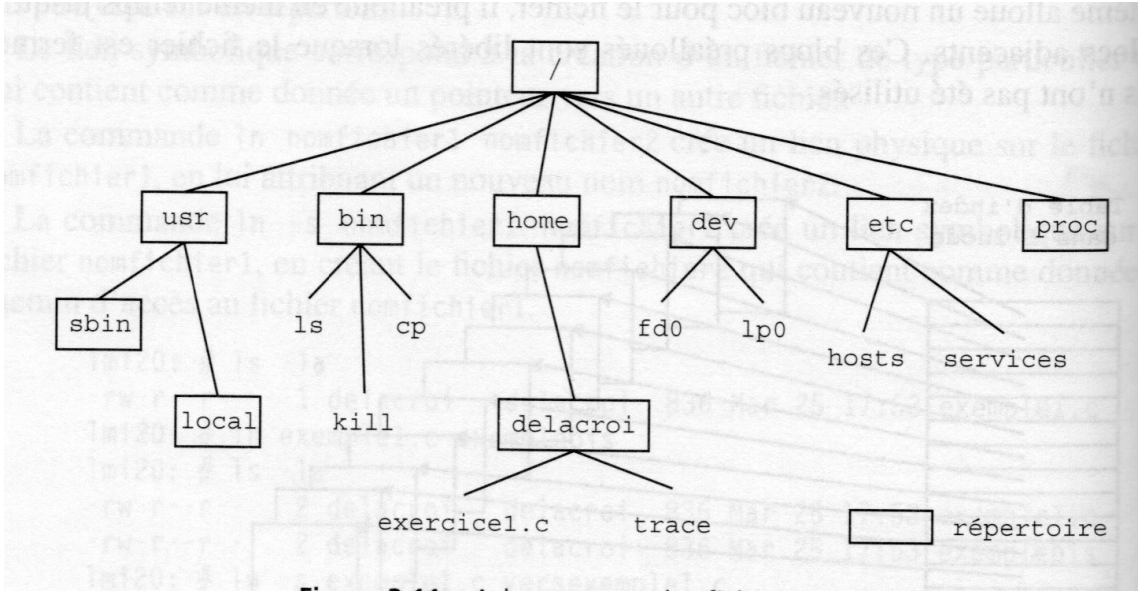
***(c) Type de fichiers Linux***

- \- fichier standard ;
- d : répertoire ;
- b : périphérique de type bloc ;
- c : Périphérique de type caractère ;
- l : lien symbolique ou logique ;
- p : tube nommé (named pipe) pour la communication entre processus ;
- s : socket, comme les tubes nommés mais généralement dans un contexte réseau.

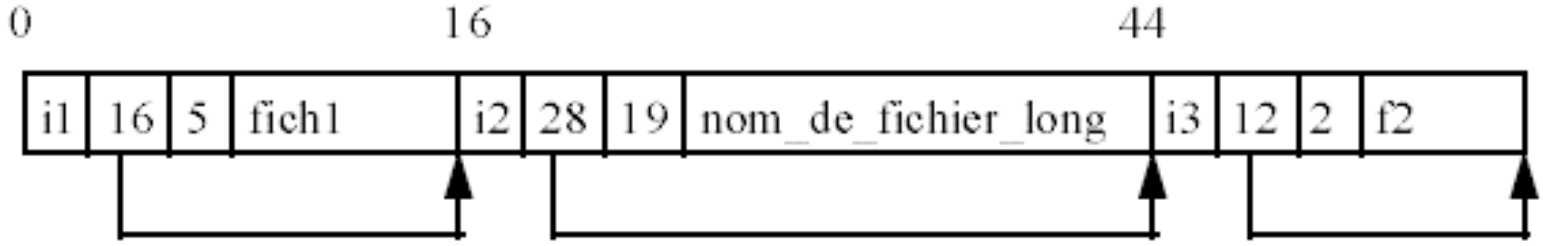
```
simoko@karmic:~$ ls -ld /etc/motd /lib /dev/sda /dev/null /etc/rc.local /dev/initctl /dev/log
prw-------  1 root root    0 2009-10-25 12:36 /dev/initctl
srw-rw-rw-  1 root root    0 2009-10-25 12:36 /dev/log
crw-rw-rw-  1 root root 1, 3 2009-05-18 05:40 /dev/null
brw-rw----  1 root disk 8, 0 2009-10-25 12:35 /dev/sda
lrwxrwxrwx  1 root root   13 2009-05-18 05:40 /etc/motd -> /var/run/motd
-rwxr-xr-x  1 root root  306 2009-05-18 05:40 /etc/rc.local
drwxr-xr-x 18 root root 4096 2009-10-11 01:08 /lib
simoko@karmic:~$
```

# Système de fichiers – partie 2

## *Structure d'un répertoire*

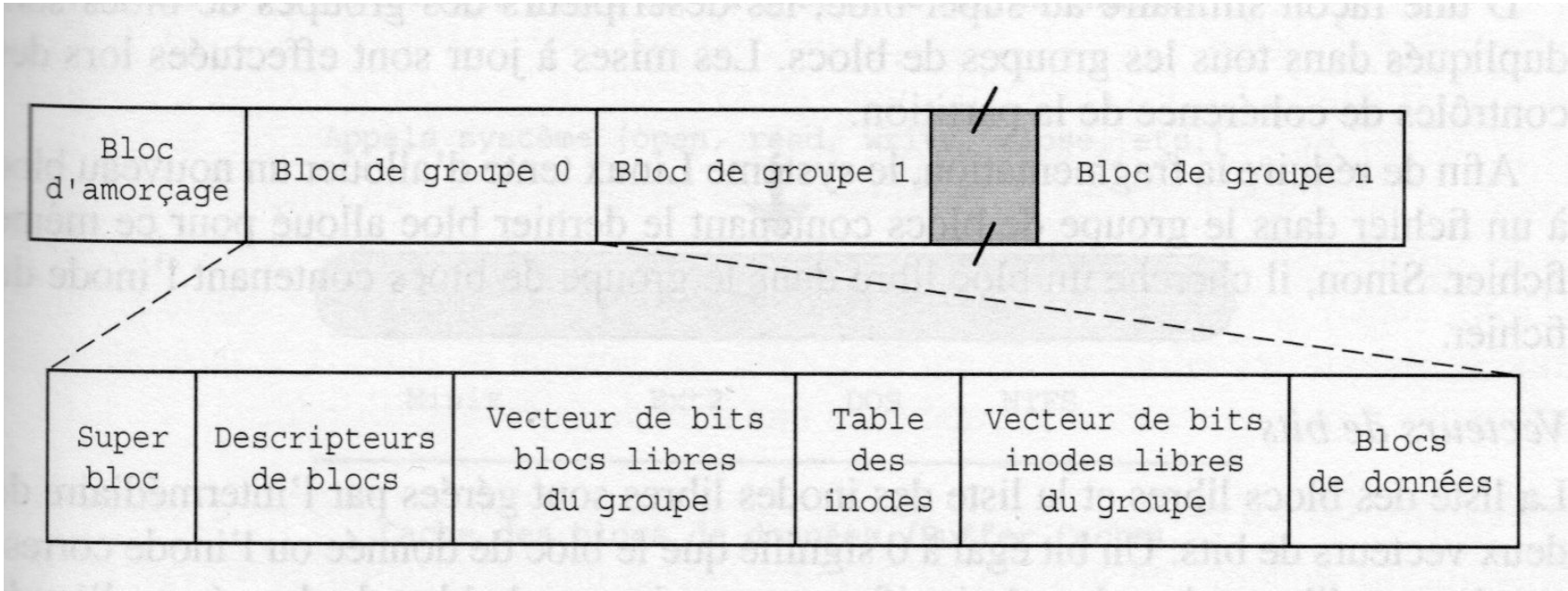



- Entrée de répertoire :
  - Le numéro d'inode ;
  - La longueur de l'entrée du répertoire ;
  - La longueur du nom de fichier (255 caractères max) ;
  - Le type de fichier ;
  - Le nom du fichier.



# Système de fichiers – partie 2

## *Structure d'une partition*



# Système de fichiers – partie 2

## *Structure d'une partition*

**Super-bloc du système de gestion de fichiers**

- Le nom de la partition ;
- L'heure de la dernière opération de montage, le nombre d'opération de montage réalisées sur la partition ;
- La  taille de la partition en blocs ;
- Le nombre total d'inodes dans la partitions ;
- La taille d'un bloc dans la partition ;
- Le nombre de blocs libres et d'inode libres dans la partition ;
- Le nombre de blocs et d'inode par groupe ;
- L'heure du dernier contrôle de cohérence et l'intervalle de temps entre chaque contrôle de cohérence.

# Système de fichiers – partie 2

## *Structure d'une partition*

**Descripteurs de groupes de blocs**

- Les numéros des blocs contenant les vecteurs de bits gérant la liste des blocs libres et la liste des inodes libres ;
- Le nombre de blocs libres et le nombre d'inodes libres dans le groupe ;
- Le nombre de répertoires dans le groupe ;
- Le numéro du premier bloc contenant les inodes libres.

**Vecteurs de bits**

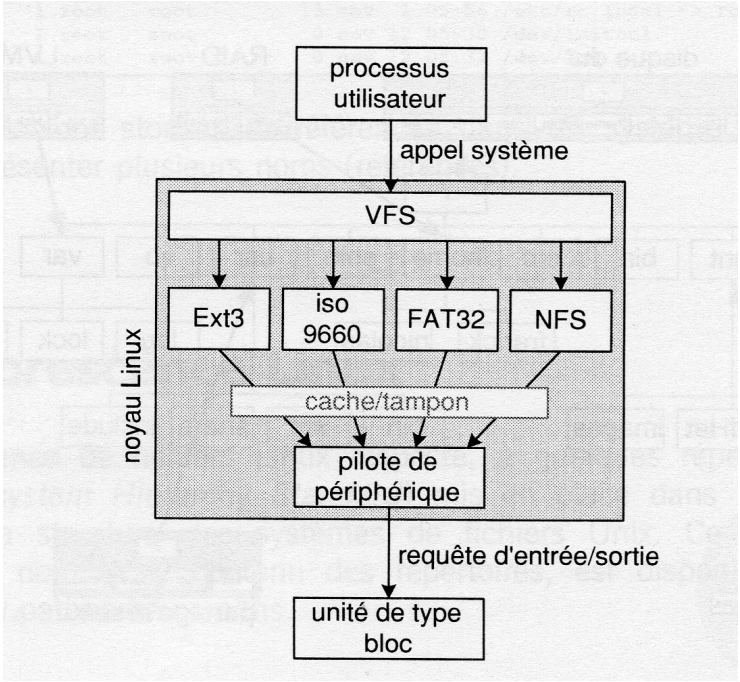
• La liste des blocs libres et la liste des inodes libres sont gérées par l'intermédiaire de deux vecteurs de bits (*bitmap blocs*).

# Système de fichiers – partie 2

## *VFS : virtual file system*

• Ce système de fichiers virtuel permet de faire le lien entre les applications et les différentes implémentation des systèmes de fichiers.

• Grossièrement pour un appel système :
  - Vérification des paramètres[1] ;
  - Conversion du nom de fichier en numéro de périphérique et numéro d'inode ;
  - Vérification des permissions ;
  - Appel à la fonction spécifique du système de gestion de fichier concerné.




**Exemples** : (1) Manipulation de fichier, (2) Manipulation d'un répertoire.

# Système de fichiers – partie 2
## *Créer un système de fichiers*

FIN PRESENTATION

### a. mkfs, syntaxe générale

• La commande pour créer un système de fichiers est ***mkfs***. ***mkfs*** appelle d'autres programmes en fonction du type de système de fichiers sélectionné.

```
mkfs -t typefs options périphérique
```

• C'est typefs qui détermine le type de système de fichiers et donc le programme appelé. Il existe un programme par type de système de fichiers :

- ***ext2*** : mkfs.ext2
- ***ext3*** : mkfs.ext3
- ***reiserfs*** : mkfs.reiserfs
- ***vfat*** : mkfs.vfat (pour tous les formats FAT, mais mkfs.msdos existe)
- ***ntfs*** : mkfs.ntfs

• Plutôt que d'utiliser mkfs, vous pouvez utiliser directement les programmes correspondant au type de système de fichiers à écrire.

# Système de fichiers – partie 2
## *Créer un système de fichiers*

### b. Un premier exemple en ext2

- Vous allez créer un système de fichiers de type ext2 sur la partition sdb1.
- Voici la commande de base :

  - Il est possible de donner une étiquette (un nom) au système de fichiers.
  - Chaque bloc fait 4096 octets.
  - Il y a 261048 blocs.
  - Les i-nœuds (inodes) représentent le nombre maximal de fichiers : 65280.
  - 5% de l'espace disque est réservé à root, ce qui signifie qu'un utilisateur lambda ne pourra pas remplir le disque à plus de 95%.
  - Les tables d'inodes sont réparties par groupes.
  - Il y a un superbloc principal et quatre superblocs de secours (un par groupe).
  - Il est possible de modifier certains paramètres du système de fichiers avec la commande tune2fs.

```
debian:/home/komo# mkfs -t ext2 /dev/sdb1
mke2fs 1.41.3 (12-Oct-2008)
Étiquette de système de fichiers=
Type de système d'exploitation : Linux
Taille de bloc=4096 (log=2)
Taille de fragment=4096 (log=2)
65280 i-noeuds, 261048 blocs
13052 blocs (5.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=268435456
8 groupes de blocs
32768 blocs par groupe, 32768 fragments par groupe
8160 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        32768, 98304, 163840, 229376

Écriture des tables d'i-noeuds : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété

Le système de fichiers sera automatiquement vérifié tous les 21 montages ou
après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou -i
pour écraser la valeur.
```

# Système de fichiers – partie 2
## *Créer un système de fichiers*

### c. ext2 et ext3

Les systèmes des fichiers ext2 et ext3 étant compatibles, ils partagent les mêmes paramètres, dont voici les plus courants :

| Paramètre | Signification |
|---|---|
| -b | Taille des blocs en octet, multiple de 512. Si la taille n'est pas précisée, elle sera déterminée par la taille de la partition. Tout fichier créé sur le disque occupe au moins un bloc et donc si on manipule un grand nombre de petits fichiers il faut mettre une valeur basse (ex : 1024). |
| -c | Vérifie les mauvais blocs avant de créer le système de fichiers. On peut aussi utiliser la commande **badblocks**. |
| -i | Ratio octets/inode. La taille de la table des inodes est calculée en fonction de la taille totale du système de fichiers. Un inode occupe 128 octets. En mettre moins limite le nombre de fichiers possibles mais permet de gagner de la place. -i 4096 : un inode pour chaque 4 ko. |
| -m | Pourcentage réservé au super-utilisateur, par défaut 5%. Le mettre à zéro permet de gagner de la place et root pourra tout de même y travailler. |
| -L | Label, étiquette (nom) du système de fichiers, utile pour le montage. |
| -j | Crée un journal ext3, donc crée un système de fichiers ext3. |

# Système de fichiers – partie 2
## *Créer un système de fichiers*

### c. ext2 et ext3

• L'exemple suivant crée un système de fichiers ***journalisé ext3*** (option -j) avec une taille de ***blocs*** de ***2048*** octets, et ***un inode*** pour chaque ***16 Ko***. La totalité du système est utilisable par les utilisateurs (***aucun espace*** n'est réservé pour ***root***). L'***étiquette*** est ***DATA***.

```
debian:/home/komo# mkfs -t ext2 -j -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
mke2fs 1.41.3 (12-Oct-2008)
Étiquette de système de fichiers=DATA
Type de système d'exploitation : Linux
Taille de bloc=2048 (log=1)
Taille de fragment=2048 (log=1)
65280 i-noeuds, 522096 blocs
0 blocs (0.00%) réservés pour le super utilisateur
Premier bloc de données=0
Nombre maximum de blocs du système de fichiers=534773760
32 groupes de blocs
16384 blocs par groupe, 16384 fragments par groupe
2040 i-noeuds par groupe
Superblocs de secours stockés sur les blocs :
        16384, 49152, 81920, 114688, 147456, 409600, 442368

Écriture des tables d'i-noeuds : complété
Création du journal (8192 blocs) : complété
Écriture des superblocs et de l'information de comptabilité du système de
fichiers : complété

Le système de fichiers sera automatiquement vérifié tous les 35 montages ou
après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou -i
pour écraser la valeur.
```

• Notez que la ligne de commande suivante a exactement le même effet car le système de fichiers ext3 induit le paramètre -j :

```
# mkfs -t ext3 -b 2048 -i 16384 -m 0 -L "DATA" /dev/sdb1
```

# Système de fichiers – partie 2
## *Créer un système de fichiers*

**c. ext2 et ext3**

***ext2 vers ext3***

• Ext3 est un système de fichiers ext2 auquel on a rajouté un journal. Vous pouvez convertir un système de fichiers ext2 en ext3 (ou de ext2 vers ext4 ou de ext3 vers ext4) en utilisant ***tune2fs***.

```
debian:/home/komo# tune2fs -j /dev/sdb1
tune2fs 1.41.3 (12-Oct-2008)
Création de l'i-noeud du journal : complété
Le système de fichiers sera automatiquement vérifié tous les 35 montages ou
après 180 jours, selon la première éventualité. Utiliser tune2fs -c ou -i
pour écraser la valeur.
```

***ext3 vers ext2***

• Pour revenir en ext2, il faut supprimer le journal encore une fois avec tune2fs et le paramètre -O (grand O) :

```
debian:/home/komo# tune2fs -O ^has_journal /dev/sdb1
tune2fs 1.41.3 (12-Oct-2008)
```

• Vérifiez l'éventuelle présence d'un fichier ***.journal*** et supprimez-le. Enfin, effectuez une vérification avec ***fsck***.

# Système de fichiers – partie 2
## *Créer un système de fichiers*

**c. ext2 et ext3**

***Label***

- Vous pouvez afficher et changer le label du système de fichiers en tapant e2label.

```
debian:/# e2label /dev/sdb1
DATA
debian:/# e2label /dev/sdb1 OLDDATA
debian:/# e2label /dev/sdb1
OLDDATA
```

- Il ne faudra pas oublier de modifier les options de montage en conséquence.

- Un label ne doit pas dépasser 16 caractères ou il sera tronqué.

# Système de fichiers – partie 2
## *Créer un système de fichiers*

### d. vfat

- La création d'un système de fichiers VFAT se fait de la même manière.
- Cette fois vous allez le créer sur la partition sdc1 prévue pour ça.
- La commande va sélectionner automatiquement en fonction de la taille de la partition le type de FAT à créer (12, 16 ou 32).
- Le paramètre -v a été rajouté pour voir les traces de création.

```
debian:/# mkfs -t vfat -v /dev/sdc1
mkfs.vfat 3.0.1 (23 Nov 2008)
Auto-selecting FAT32 for large filesystem
/dev/sdc1 has 255 heads and 63 sectors per track,
logical sector size is 512,
using 0xf8 media descriptor, with 2088386 sectors;
file system has 2 32-bit FATs and 8 sectors per cluster.
FAT size is 2036 sectors, and provides 260535 clusters.
Volume ID is 7fe60700, no volume label.
```

- Vous pouvez spécifier plusieurs paramètres, notamment si vous souhaitez forcer la création d'un type de FAT donné :

| Paramètre | Signification |
|---|---|
| -c | Vérifie le périphérique avant la création. |
| -F | Taille de la FAT (12, 16, 32). |
| -I | Permet d'utiliser un disque complet et non une partition (pratique pour certains lecteurs MP3). |
| -n | Nom du volume (étiquette, label). |
| -v | Affichage des détails lors de la création. |

# Système de fichiers – partie 2
## *Créer un système de fichiers*

**d. vfat**

**Les mtools**

• Les mtools sont des outils permettant de travailler sur des systèmes de fichiers FAT et VFAT comme si vous étiez sous MSDOS ou la console Windows.
• Ils reprennent la syntaxe des commandes d'origine en ajoutant un m devant : `mdir`, `mformat`, `mlabel`, `mdeltree`, etc.

• Les disques et partitions sont représentés par des lettres de lecteurs *c :*, *d :*, *e* :.
• Ils peuvent représenter un disque, une partition ou un répertoire. Vous devez cependant modifier un fichier de configuration */etc/mtools.con*f.
• Par exemple pour déclarer */dev/sdc1* comme *d:* rajoutez ou modifiez la ligne suivante :

```
drive d: file="/dev/sdc1"
```

• C'est utile pour modifier après coup certaines informations comme le nom du volume du système de fichiers vfat :

```
debian:/# mlabel -s d:
 Volume has no label
debian:/# mlabel d:
 Volume has no label
Enter the new volume label : DATAFAT
debian:/# mlabel -s d:
 Volume label is DATAFAT
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### 1. mount

- La commande ***mount*** permet d'accéder aux périphériques de type blocs (les partitions) sur lesquels un système de fichiers existe.
- La commande ***mount*** attache le répertoire racine du système de fichiers à un répertoire pré-existant appelé point de montage (*mountpoint*).

```
mount -t typefs -o options périphérique point_de_montage
```

#### *a. Montage par périphérique*

- La partition sdb1 disposant de nouveau d'un système de fichiers ext3, la commande suivante rattache la racine du système de fichiers contenu dans sdb1 au répertoire */mnt/DATA*.

```
debian:/# mkdir /mnt/DATA;mount -t ext3 /dev/sdb1 /mnt/DATA
```

- La commande mount utilisée seule donne tous les détails sur les systèmes de fichiers actuellement montés (périphériques, système de fichiers, point de montage, options) :

```
debian:/# mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sdb1 on /mnt/DATA type ext3 (rw)
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### Montage par label

On peut souligner l'intérêt pratique d'utiliser des labels pour ses systèmes de fichiers. En cas de réorganisation des disques (déplacement dans une chaîne SCSI par exemple), l'ordonnancement des périphériques est modifié. L'utilisation des noms de périphériques oblige dans ce cas à modifier le fichier /etc/fstab à chaque modification. Ce n'est pas le cas avec les labels. Utilisez le paramètre -L de mount, suivi du nom du volume.

```
debian:/# umount /mnt/DATA/
debian:/# e2label /dev/sdb1
OLDDATA
debian:/# e2label /dev/sdb1 DATA
debian:/# e2label /dev/sdb1
DATA
debian:/# mount -t ext3 -L DATA /mnt/DATA
```

```
debian:/# mount
/dev/sda1 on / type ext3 (rw,errors=remount-ro)
tmpfs on /lib/init/rw type tmpfs (rw,nosuid,mode=0755)
proc on /proc type proc (rw,noexec,nosuid,nodev)
sysfs on /sys type sysfs (rw,noexec,nosuid,nodev)
procbususb on /proc/bus/usb type usbfs (rw)
udev on /dev type tmpfs (rw,mode=0755)
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev)
devpts on /dev/pts type devpts (rw,noexec,nosuid,gid=5,mode=620)
/dev/sdb1 on /mnt/DATA type ext3 (rw)
```

La liste des labels actuellement connus de Linux peut être obtenue en listant le répertoire /dev/disk/by-label. Notez que le label est un lien symbolique vers le fichier périphérique correspondant :

```
debian:/home/komo# ls -l /dev/disk/by-label/
total 0
lrwxrwxrwx 1 root root 10 nov 19 02:59 DATA -> ../../sdb1
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### Montage par UUID

• Chaque système de fichiers dispose d'un **identifiant unique** appelé ***UUID*** : *Universal Unique Identifier*, généralement un nombre aléatoire codé sur assez de bits pour que sur un ou plusieurs systèmes donnés, ils soient tous différents.
• Ainsi si le disque change de position logique, l'*UUID* ne varie pas et `mount` retrouve le système de fichiers, alors qu'il est théoriquement bien plus possible que deux systèmes de fichiers portent le même label.

• Il existe plusieurs moyens de connaître l'*UUID* d'une partition. Si *udev* est utilisé sur votre Linux, alors la commande ***vol_id*** est probablement disponible.

```
debian:/# vol_id -u /dev/sdb1
aabc49ac-4729-4d74-8a9c-c45720b4e842
```

• Si votre système de fichiers est en ext2 ou ext3 la commande dumpe2fs retourne énormément d'informations dont l'UUID :

```
debian:/# dumpe2fs -h /dev/sdb1 | grep UUID
dumpe2fs 1.41.3 (12-Oct-2008)
Filesystem UUID:          aabc49ac-4729-4d74-8a9c-c45720b4e842
```

• Enfin, le contenu de */dev/disk/by-uuid* contient les liens symboliques des UUID pointant sur le fichier périphérique correspondant :

```
debian:/home/komo# ls -l /dev/disk/by-uuid/
total 0
lrwxrwxrwx 1 root root 10 nov 19 02:59 8c4b3394-97ca-485c-b21b-3c0bb653584f -> ../../sda1
lrwxrwxrwx 1 root root 10 nov 19 02:59 aabc49ac-4729-4d74-8a9c-c45720b4e842 -> ../../sdb1
lrwxrwxrwx 1 root root  9 nov 19 02:59 bf7bfb79-b36a-4fc4-8d69-3d5c37431cd4 -> ../../md0
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### Montage par UUID

Pour monter un système de fichiers par UUID, utilisez le paramètre *-U* de *mount* :

```
debian:/home/komo# mount -U aabc49ac-4729-4d74-8a9c-c45720b4e842 /mnt/DATA
```

### Remonter un système de fichiers

- Vous n'êtes pas obligé de démonter puis de remonter un système de fichiers si vous modifiez une option.
- Si vous modifiez une option de montage du système de fichiers (via le paramètre -o)
- Vous pouvez passer l'option ***remount*** pour que la modification soit prise tout de suite en compte.
- Ne retapez pas la ligne de commande complète, mais seulement le périphérique ou le point de montage.
- Dans l'exemple suivant le système de fichiers est remonté en lecture seule :

```
debian:/home/komo# mount -o ro,remount /mnt/DATA/
debian:/home/komo# mount
...
/dev/sdb1 on /mnt/DATA type ext3 (ro)
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### *b. Options de montage*

Chaque système de fichiers accepte un certain nombre d'options de montage qui peuvent être spécifiées après le paramètre -o de mount. Les options sont séparées par des virgules. Sauf indication contraire les options suivantes fonctionnent avec ext2 et ext3.

| Option | Signification |
|---|---|
| defaults | Souvent présente, l'option **defaults** reprend les options rw, suid, dev, exec, auto, nouser, et async. |
| sync/async | Active ou désactive les écritures synchrones. Avec async les écritures passent par un tampon qui diffère les écritures (plus performant) rendant la main plus vite. Il est préférable d'activer les écritures synchrones sur des supports externes (clés USB, disques USB/Firewire/eSATA, etc.). |
| exec/noexex | Permet l'exécution/ou non des fichiers binaires sur le support. |
| noatime | Évite la mise à jour de l'horodatage à chaque accès à un fichier (pertinent sur les supports externes, disques SSD, pages web, newsgroups, etc.). |
| auto/noauto | Le système de fichiers est automatiquement monté/ne peut être monté que explicitement (voir fstab). |
| user/nouser | N'importe quel utilisateur peut monter le système de fichiers (implique noexec, nosuid, et nodev)/seul root a le droit de monter le système de fichiers (voir fstab). |
| remount | Remontage du système de fichiers pour la prise en compte de nouvelles options. |
| ro/rw | Montage en lecture seule ou lecture et écriture. |
| dev/nodev | Interpréter/Ne pas interpréter les fichiers spéciaux. |

| Option | Signification |
|---|---|
| noload | Pour ext3, ne charge pas le journal. |
| usrquota/grpquota | Ignoré par le système de fichiers lui-même mais utilisé par le sous-système de quotas. |
| acl | Permet l'utilisation des Access Control Lists. |
| user_xattr | Pour ext3 et xfs, accepte les attributs étendus sur les fichiers, par exemple pour y coller des informations additionnelles (l'encodage du texte, etc.), des champs d'indexation (utilisés par Beagle par exemple), etc. |
| umask | Pour FAT/NTFS, applique un autre masque global que celui par défaut (ex 133). |
| dmask=/fmask= | FAT/NTFS, différencie les masques pour les répertoires et les fichiers. |
| uid=/gid= | FAT/NTFS, comme les droits et propriétaires ne sont pas gérés, applique un utilisateur ou un groupe par défaut sur les fichiers (ex gid=users). |

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### *c. umount*

La commande ***umount*** détache le système de fichiers du point de montage.

```
debian:/home/komo# umount /mnt/DATA/
```

```
komo@debian:~$ cd /mnt/DATA/
komo@debian:/mnt/DATA$
```

```
debian:/home/komo# umount /mnt/DATA/
umount: /mnt/DATA: device is busy
umount: /mnt/DATA: device is busy
```

- Si un ou plusieurs fichiers du système de fichiers à démonter sont encore en cours d'utilisation, alors *umount* ne marchera pas.
- Vous devez vous assurer qu'aucun processus n'accède au système de fichiers.
- La commande lsof vous aide à déterminer quel processus est actuellement en train d'utiliser un fichier du point de montage.
- Ici c'est le shell bash lancé par l'utilisateur komo qui y est présent (probablement que le répertoire courant est /mnt/DATA).

```
debian:/home/komo# lsof /mnt/DATA/
COMMAND  PID USER   FD   TYPE DEVICE SIZE NODE NAME
bash    3883 komo  cwd    DIR   8,17 2048    2 /mnt/DATA/
```

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### *c. umount*

De manière très violente vous pouvez forcer l'arrêt des processus accédant au point de montage avec fuser. Il est fort probable que l'utilisateur concerné n'apprécie pas du tout (dans le cas présenté ici son shell sera arrêté et il sera déconnecté).

```
debian:/home/komo# w
 04:00:41 up  1:01,  3 users,  load average: 0,25, 0,10, 0,03
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
komo     tty7     :0               03:00    0.00s 45.12s  0.80s x-session-manager
root     pts/1    :0.0             03:01    0.00s  0.58s  0.00s w
komo     pts/2    :0.0             04:00    7.00s  0.32s  0.32s bash


debian:/home/komo# fuser -km /mnt/DATA/
/mnt/DATA/:           3917c
debian:/home/komo# w
 03:57:29 up 58 min,  2 users,  load average: 0,00, 0,02, 0,01
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
komo     tty7     :0               03:00    0.00s 39.48s  0.80s x-session-manager
root     pts/1    :0.0             03:01    0.00s  0.60s  0.02s w
```

# Système de fichiers – partie 2
## *Accéder aux systèmes de fichiers*

***d. /etc/fstab***

• Le fichier ***/etc/fstab*** contient une configuration statique des différents montages des systèmes de fichiers.
•Il est **appelé à chaque démarrage du système** car c'est ici qu'on indique les périphériques et leurs points de montage. Il contient six champs.

```
périphérique point_de_montage typefs options dump fsck
```

• Les champs sont séparés par des espaces ou des tabulations.

| Champ | Description |
|---|---|
| périphérique | Le périphérique à monter. Il peut être spécifié en tant que chemin de périphérique (/dev/hda1 par exemple), que label de système de fichiers s'il existe (LABEL=/home), ou encore en tant que UUID (UUID=xxxx). |
| point de montage | Le répertoire d'accès au système de fichiers monté. |
| typefs | Le type (ext2, etx3, reiser, vfat, etc.) du système de fichiers. |
| options | Les options, séparées par des virgules, vues précédemment. |
| dump | Fréquence de dump pour les outils de dump ou de sauvegarde. |
| fsck | Fréquence de vérification du système de fichiers.0=ignorer. 1=en premier, 2 en second, etc. Les systèmes ayant le même numéro sont vérifiés en parallèle. |

# Système de fichiers – partie 2
## *Accéder aux systèmes de fichiers*

### *d. /etc/fstab*

• Voici un exemple tronqué (les systèmes de fichiers virtuels n'apparaissent pas) de fichier /etc/fstab :

```
/dev/sda3  /         ext3   acl,user_xattr        1 1
/dev/sda2  /boot     ext3   acl,user_xattr        1 2
/dev/sdb1  /home     ext3   acl,user_xattr        1 2
/dev/sda6  /public   ext3   acl,user_xattr        1 2
/dev/sda1  /windows  ntfs   noauto,users,gid=users,umask=0002,utf8=true 0 0
/dev/sda5  swap      swap     defaults              0 0
```

• Plutôt que de préciser des noms de périphériques statiques, il peut être préférable de préciser une étiquette (label, volume) ou un UUID (*il est où le point de montage dans la seconde ligne??*).

```
LABEL=BOOT   /boot             ext3       acl,user_xattr        1 2
UUID=f0bed37c-9ddc-4764-ae7f-133205c36b5d ext3    acl,user_xattr     1 2
```

**Montage au boot**

• Le contenu de */etc/fstab* peut être utilisé après l'initialisation du système pour monter et démonter ponctuellement les systèmes de fichiers qui n'ont pas par exemple l'option noauto, ou les supports de masse comme les lecteurs CD/DVD.
• Dans ce cas vous utilisez simplement les labels, les points de montage ou le périphérique sans avoir à réécrire toute la ligne de commande.
• *mount* va chercher ses renseignements dans */etc/fstab*.

# Système de fichiers – partie 2

## *Accéder aux systèmes de fichiers*

### *d. /etc/fstab*

**Montage au boot**

Lors de la séquence de **démarrage** le fichier */etc/fstab* est balayé par l'un des scripts, presque au tout début du boot, entre le chargement du noyau et le démarrage des services. Tous **les systèmes de fichiers** ne possédant **pas noauto** comme option sont automatiquement **montés** (le auto est implicite). Le premier à l'être est le système de fichiers racine /. Puis viennent ensuite le swap et les autres systèmes de fichiers s'ils sont spécifiés (ex : /home, /usr, etc.) ainsi que les systèmes de fichiers virtuels /proc, /sys , /dev/pts, etc.

**Montage manuel**

Le contenu de */etc/fstab* peut être utilisé après l'initialisation du système pour monter et démonter ponctuellement les systèmes de fichiers qui n'ont pas par exemple l'option noauto, ou les supports de masse comme les lecteurs CD/DVD. Dans ce cas vous utilisez simplement les labels, les points de montage ou le périphérique sans avoir à réécrire toute la ligne de commande. ***mount* va chercher ses renseignements dans */etc/fstab*.**

```
mount /home
mount -L /u01
mount LABEL=/boot
mount /dev/hda5
```

**Tout monter**

Si vous avez effectué des modifications importantes dans la fstab comme le rajout de plusieurs nouveaux points de montage, vous pouvez, au lieu de monter chaque système de fichiers un par un, tous les monter d'un coup avec le paramètre -a de mount : `# mount -a`

# Système de fichiers – partie 2
## *Accéder aux systèmes de fichiers*

### *e. Cas des CD et images ISO*

Les CD-Rom, DVD-Roms et autres supports de ce type se montent comme n'importe quel disque. Les CD-Roms et certains DVD-Roms utilisent le système de fichiers iso9660.

```
# mount -t iso9660 /dev/cdrom /media/cdrom
```

La plupart des DVD-Roms utilisent plutôt le format UDF (Universal Disk Format).

```
# mount -t udf /dev/cdrom /media/dvd
```

Les distributions Linux récentes s'affranchissent du montage manuel des supports externes qu'il s'agisse d'un CD, DVD, clé USB ou disque externe. Les services *udev* se chargent au branchement ou à l'insertion du support de créer les fichiers spéciaux associés, et de monter et de démonter les supports automatiquement.

Une image ISO est une image du contenu d'un CD ou d'un DVD. C'est un système de fichiers iso9660 ou udf dans un fichier. Il est possible d'utiliser cette image comme un périphérique, à l'aide des périphériques de loopback. L'image est rattachée à un périphérique de loopback, et les outils passent par ce périphérique comme s'il s'agissait d'un disque.

```
# mount -o loop -t iso9660 image.iso /mnt/iso
```

# Système de fichiers – partie 2

## *Validation des acquis*

**Disques et systèmes de fichiers**

**1.** Que représente le périphérique /dev/hde2 ?
A - Impossible, il ne peut y avoir que 4 disques IDE dans un PC.
B - Le deuxième disque du cinquième port IDE.
C - Le disque esclave connecté sur la troisième prise IDE.
D - La seconde partition du disque maître du contrôleur IDE2.

**2.** Comment est vu un disque connecté en SATA sur une machine dont le SATA natif est supporté par Linux ?

**3.** La commande lsscsi permet-elle de différencier le « vrai »SCSI des disques SATA ou USB ?

**4.** Comment obtenir les informations détaillées du disque /dev/hda ?
A - cat /dev/hda
B - hdparm -I /dev/hda
C - getparm /dev/hda
D - sdparm /dev/hda

**5.** Où sont stockées les méta-données d'un fichier ?

# Système de fichiers – partie 2

## *Validation des acquis*

**Disques et systèmes de fichiers**

**6.** Pourquoi un système de fichiers journalisé est-il préférable aux autres ?
A - Parce qu'il est plus récent.
B - Parce qu'il est plus rapide.
C - Parce qu'il est plus fiable.
D - Parce qu'il n'y a rien d'autre.

**7.** Est-il possible d'utiliser un système de fichiers FAT ou VFAT comme système de la partition principale (racine) d'un système Linux ?

**8.** Est-il pertinent d'utiliser xfs sur un poste de travail ?

**9.** Où se situe le MBR ?
A - Au tout début du disque.
B - Au début de la première partition primaire.
C - Au début de la partition étendue.
D - Au début de la première partition logique.

# Système de fichiers – partie 2

## *Validation des acquis*

**Partitions**

**10.** Comment créer plus de quatre partitions sur son disque ?
A - Impossible : il ne peut y avoir que 4 partitions.
B - En créant une partition étendue qui contiendra d'autres partitions logiques.
C - En rajoutant un disque et en créant un LVM.
D - En supprimant le swap, on peut rajouter une partition.

**11.** Quel est le type hexadécimal de partition à créer pour y placer un système de fichiers ext3 ?
A - fd
B - 8e
C - 88
D - 82

**12.** Quel paramètre faut-il passer à fdisk pour lister les partitions du disque /dev/sdb ?

**13.** Sur un disque /dev/hdb vide, quelle est la séquence complète sous fdisk pour créer une partition primaire de type Linux qui utilise tout l'espace ?

**14.** La partition que vous avez créée n'est pas encore vue par Linux. Comment mettre à jour la table des partitions du noyau ?
A - partprobe /dev/hdb
B - cat /proc/partitions
C - touch /dev/hdb1
D - Il faut obligatoirement rebooter.

# Système de fichiers – partie 2

## *Validation des acquis*

**Création du système de fichiers**

**15.** Si un système de fichiers a une taille de bloc de 4096 octets, quelle place occupe un fichier de 1024 octets ?
A - Un quart de bloc.
B - 1 ko.
C - Un bloc.
D - Le fichier sera complété par le système pour occuper 4096 octets.

**16.** Combien y a-t-il de superblocs dans un système de fichiers ext3 ?
A - Au moins 2.
B - Au moins 4.
C - Impossible de le savoir.
D - Ça dépend de la taille du système de fichiers.

**17.** Quelle est la seule information manquante sur un fichier dans l'inode ?

**18.** Vous supprimez un fichier dont le compteur de liens était à 2. A-t-il totalement disparu ?

**19.** Est-il simple de récupérer un fichier supprimé ?

**20.** Quelle ligne de commande utiliser pour créer une partition ext3 dotée du label « PUBLIC » dans la partition /dev/hdb1.

# Système de fichiers – partie 2

## *Validation des acquis*

**Montage**

**21.** Quelle commande devez-vous saisir pour accéder à la partition /dev/hdb1 depuis /media/PUBLIC ?

**22.** Quelle ligne devez-vous rajouter dans /etc/fstab pour que la partition précédente soit automatiquement sur /media/PUBLIC au boot, via son nom de volume, avec les ACL ?

**23.** En tant que root tentez de démonter le système de fichiers racine. Pourquoi obtenez-vous ce résultat ?
Gestion du système de fichiers