Système

Disques

Gilles Roussel

Gilles.Roussel@univ-mlv.fr

http://igm.univ-mlv.fr/ens/Licence/L2/2012-2013/System/

Licence 2

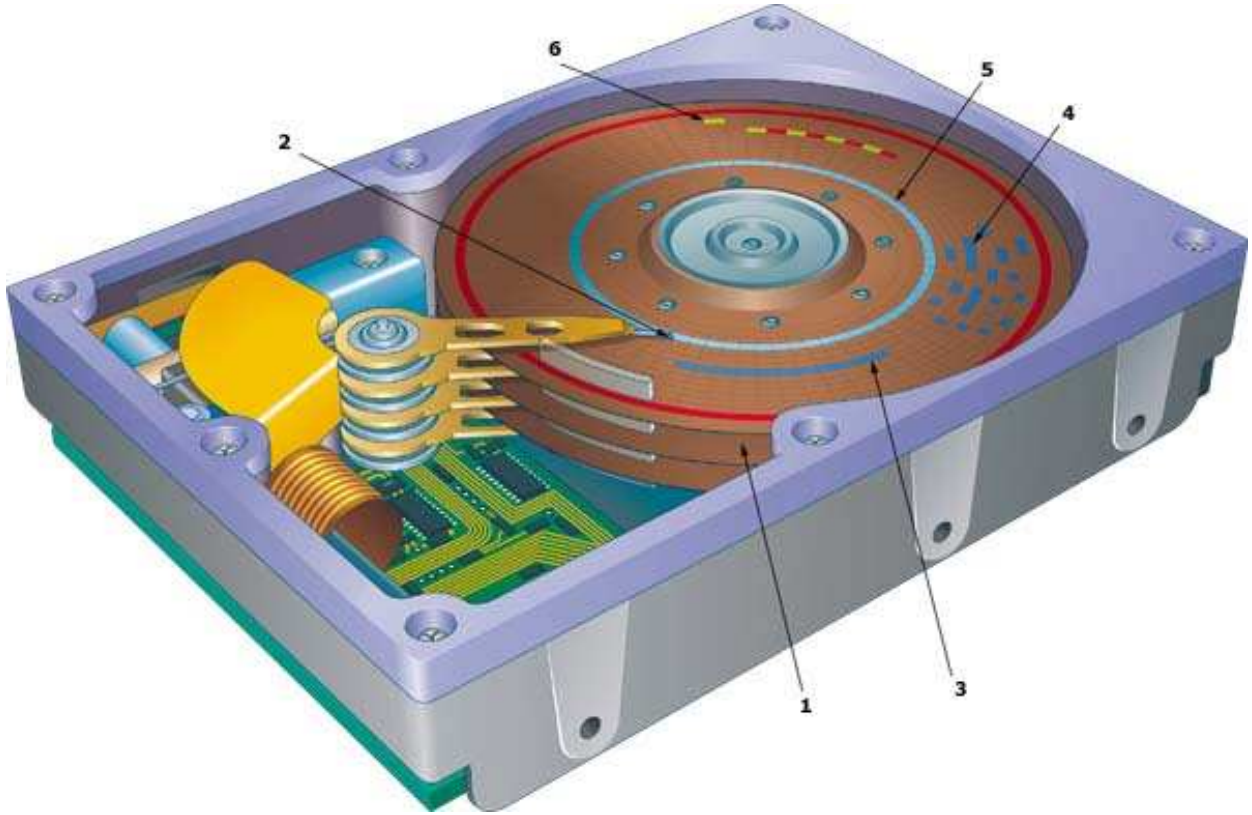
7 avril 2015

C'est quoi un disque ?

C'est quoi un disque ?

- Différents types de disques
  - disques magnétiques (disques durs, disquettes, etc.)
  - disques optiques (CDROM, CD-R, DVD, etc.)
  - mémoire flash (clefs USB, Solid-State Disk)
- Disque dur décomposé en :
  - plateaux (entre 1 et 10)
  - ayant deux faces liées à une tête ou *head*
  - contenant des pistes ou *tracks* (plusieurs milliers)
  - regroupées en cylindres par l'alignement des têtes (même si une seule est active à la fois)
  - décomposées en secteurs (entre 60 et 120) de taille « fixe » (en général 512 octets de données)
- Nombre de secteurs par piste variable sur les nouveaux disques (notion de zones)
  - Les zones externes du disque ont plus de pistes que les zones internes

C'est quoi un disque ?



© http://www.vnunet.fr/vnuimg/dossiers/svmmac/disquedur/disque.htm

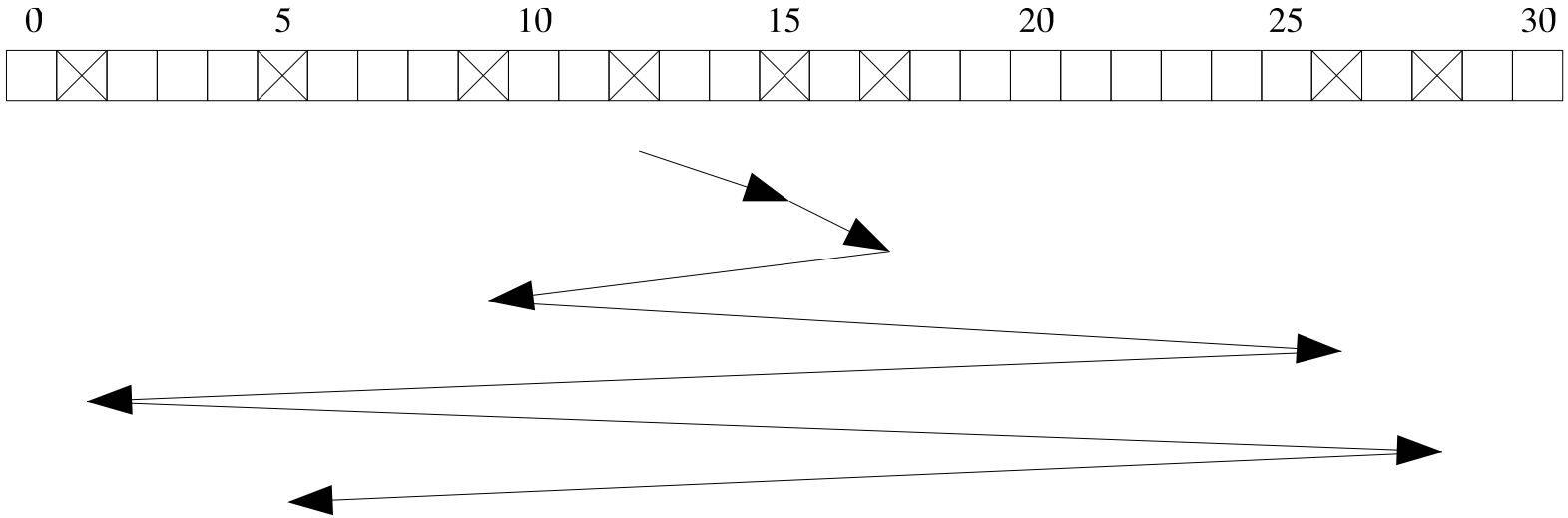
Organisation du disque

- Les secteurs sont ordonnés dans l'ordre des pistes, des faces et enfin des secteurs
  - Adressage CHS (*Cylinder Head Sector*)
- Minimiser le mouvement des têtes et la latence entre deux lectures de secteurs qui se suivent
  - Numérotation entrelacée et obliquité (décalage) de cylindre et de tête
- Décomposition en secteurs réalisée par formatage bas niveau

Ordonnancement des requêtes disque

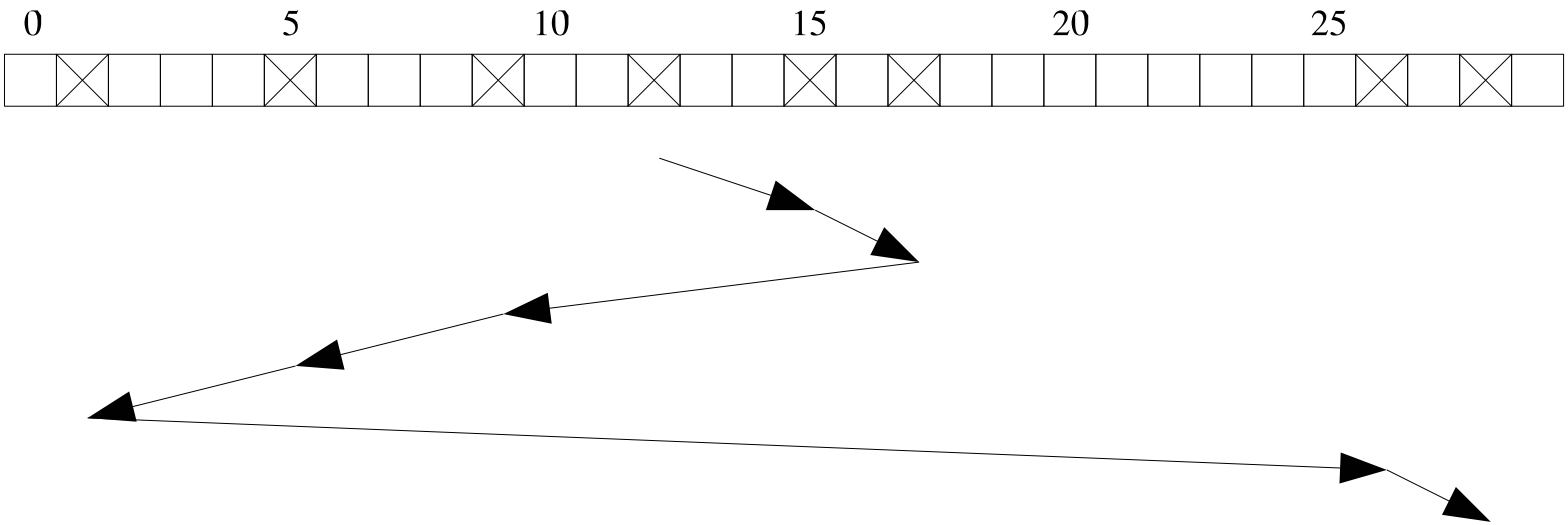
- Minimiser le temps d'accès :
  - positionnement du bras ;
  - temps de rotation du plateau ;
  - temps de transfert.
- Système ordonne les requêtes dans la file d'attente du disque
- Requête :
  - lecture ou écriture ;
  - adresse disque ;
  - adresse mémoire ;
  - nombre d'octets.
- Différents algorithmes d'ordonnancement
  - Performances dépendent du nombre et du type des requêtes
  - Requêtes et leur ordonnancement dépendent du système de fichiers

FCFS : *First Come First Served*



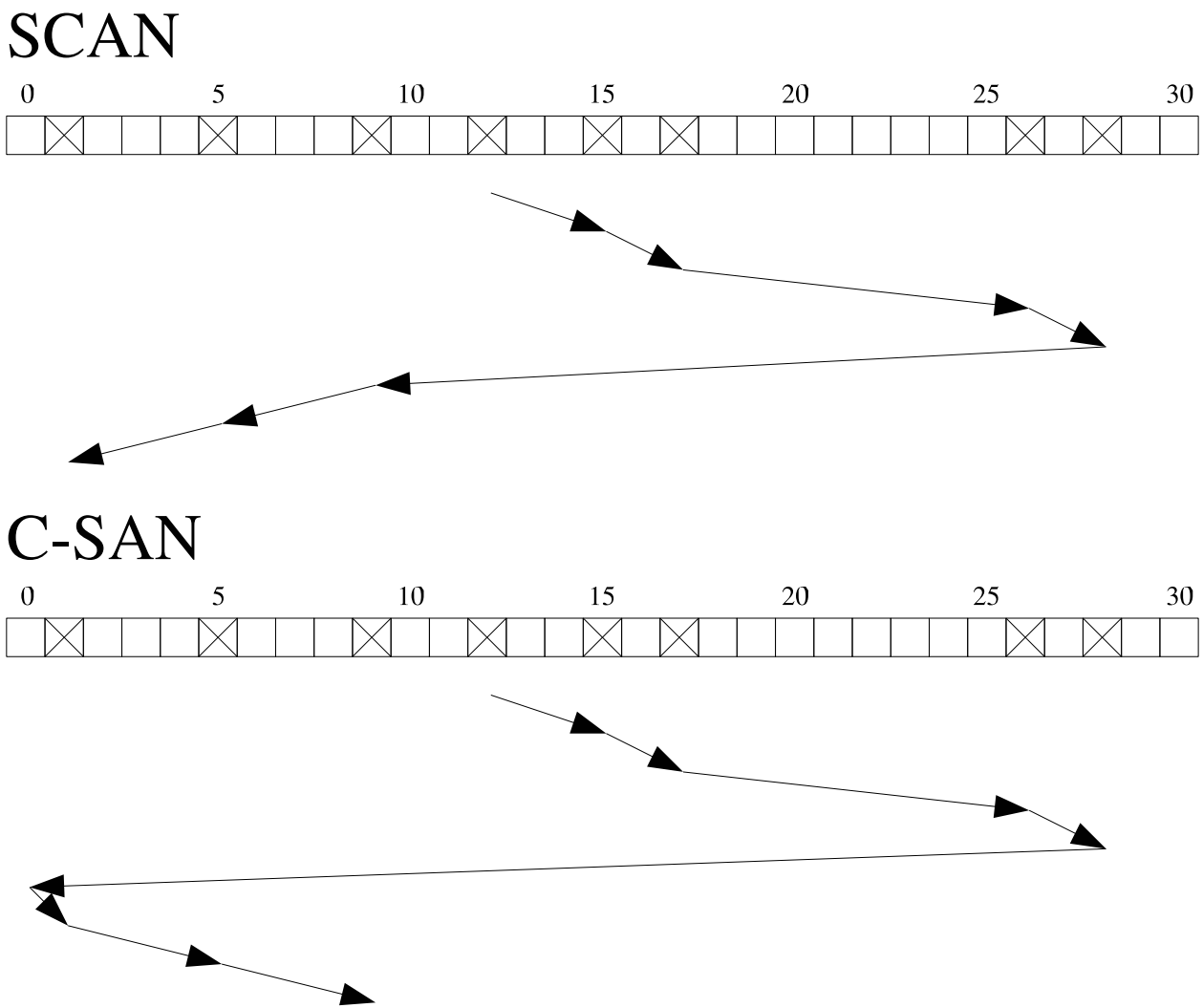
Pas d'optimisation

SSTF : *Shortest Seek Time First*



Risque de « famine »

SCAN ou algorithme de l'ascenseur



Optimiser l'accès au disque

- Éviter l'accès au disque à chaque lecture écriture
- Données gardées en mémoire aussi longtemps que possible
- *Buffer cache* chargé de la gestion des blocs en mémoire
- Problème de cohérence pour l'écriture
- Utilisation du *read-ahead* pour améliorer les performances en lecture

Formatage bas niveau

- Formatage bas niveau réalisé en usine
- Formatage bas niveau indépendant des systèmes d'exploitation
- Formatage bas niveau utilise de la place car il ajoute :
  - un espace inter-secteurs
  - un en-tête par secteur contenant en général son numéro (permet de positionner la tête) et d'autres informations sur le secteur
  - un code correcteur d'erreur (ECC) des données du secteur mis à jour à chaque écriture
  - des secteurs de rechanges pour les secteurs défectueux
- Explique (avec les problèmes d'unité) la différence entre la taille du disque vendue et la taille perçue

L'accès au secteurs

- Secteurs accédés *via* un contrôleur plus ou moins intelligent :
  - SCSI (*Small Computer System Interface*),
  - ATA (*AT Attachement*) ou IDE (*Integrated Drive Electronics*)
  - SATA (*Serial ATA*)
- Optimisation des déplacements pour plusieurs requêtes réalisée par le contrôleur et/ou le système
- Sous Unix disque vu comme un fichier accessible par bloc `/dev/hda`, `/dev/sdb`, etc.

Les partitions

- Possibilité de découper le disque en morceaux (partitions)
- Partition : ensemble de cylindres contigus
- Repérée par son cylindre de début et son cylindre de fin
  - Géométrie virtuelle CHS pour compatibilité
  - En fait, adressage logique LBA (*Logical Block Addressing*)
- Sur PC :
  - Premier cylindre réservé
  - Secteur `0` ou MBR (*Master Boot Record*) contient une table de quatre partitions primaires (`/dev/hda1`, ... ,`/dev/hda4`)
  - Une des partitions peut être étendue pour créer une liste chaînée de tables partitions
  - Partitionnement réalisé *via* `fdisk`, `sfdisk`, etc.

Les partitions

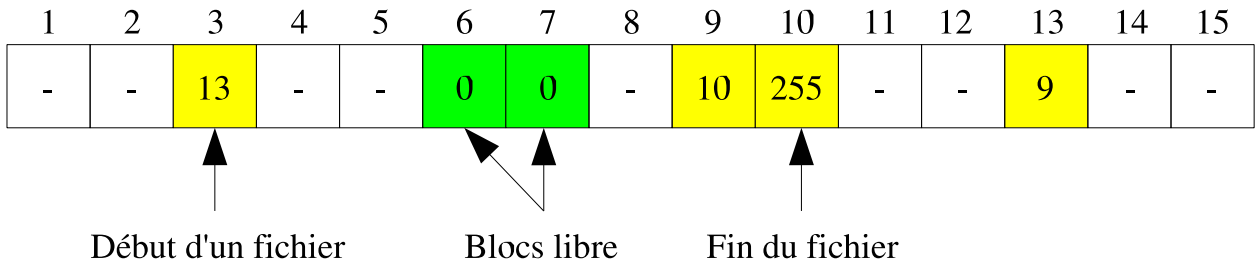

```
# sfdisk -uS -x -l /dev/hda
Disque /dev/hda: 4864 cylindres, 255 têtes, 63 secteurs/piste
Unités= secteurs de 512 octets, décompte à partir de 0

   Périph Amorce  Début       Fin   #secteurs Id  Système
/dev/hda1            63    112454     112392  de  Dell Utility
/dev/hda2   *    112455  24691904   24579450   7  HPFS/NTFS
/dev/hda3      24691905  78140159   53448255   f  W95 Ext'd (LBA)
/dev/hda4             0         -          0   0  Vide

/dev/hda5      24691968  45174779   20482812   b  W95 FAT32
    -          45174780  47231099    2056320   5  Extended
    -          24691905  24691904          0   0  Vide
    -          24691905  24691904          0   0  Vide

/dev/hda6      45174843  47231099    2056257  82  Linux swap
    -          47231100  63617399   16386300   5  Extended
    -          45174780  45174779          0   0  Vide
    -          45174780  45174779          0   0  Vide

/dev/hda7      47231163  63617399   16386237  83  Linux
    -          63617400  67713974    4096575   5  Extended
    -          47231100  47231099          0   0  Vide
    -          47231100  47231099          0   0  Vide
....
```

Les système de fichiers

- Chaque partition est vue comme disque
- Organisation des secteurs pour créer les fichiers liée à un système de fichiers particulier
  - FAT-16, FAT-32 et VFAT (*File Allocation Table*)
  - NTFS (*New Technology File System*)
  - Ext2, Ext3, Ext4 (*Extended File System*)
  - ReiserFS
  - HFS (*Hierarchical File System*)
  - UFS (*Unix File System*)
- Type précisé dans la table des partitions
- Formatage haut niveau réalisé par `mkfs`, `format`, ...
- Fichier = ensemble de blocs (ou clusters)
- Un bloc correspond à plusieurs secteurs $\Rightarrow$ taille minimale d'un fichier et perte de place

L'exemple de FAT

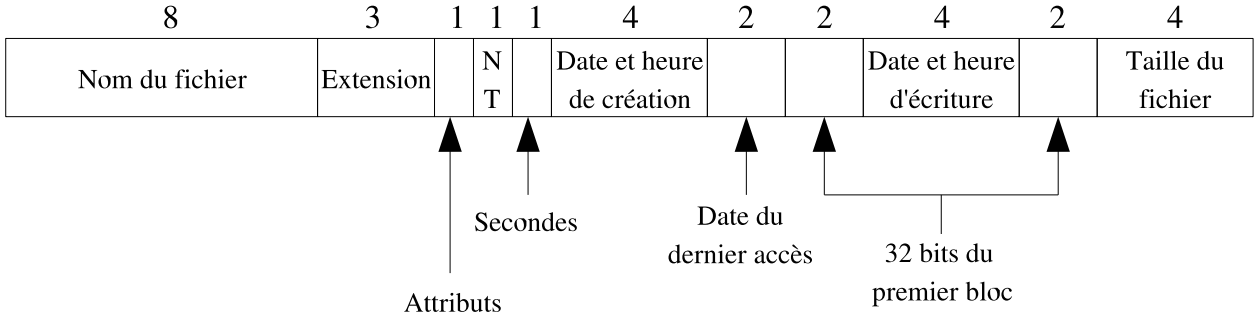
- Ensemble des blocs du fichiers chaînés dans la table d'allocation (FAT) qui occupe les premiers secteurs de la partition (64 Ko)
- FAT manipulée en mémoire (deux sauvegardes sur disque)
- Sert également à repérer les blocs libres



- Défragmentation = permettre au fichier de chaîner des blocs contigus pour éviter la latence d'accès

L'exemple de FAT

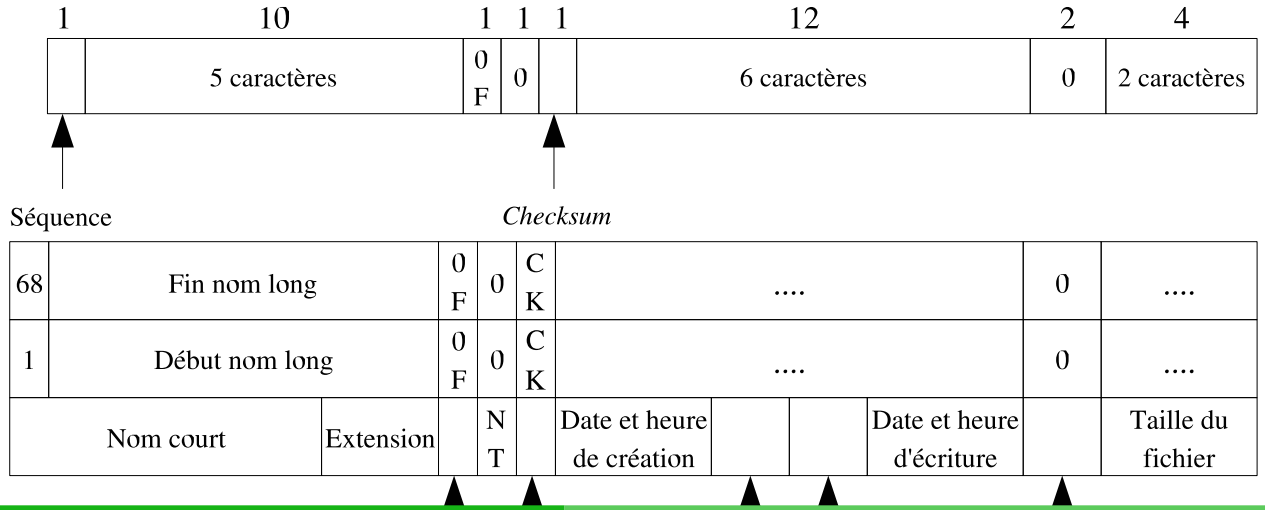
- Répertoire = ficher ordinaire d'un type particulier
- Premier bloc contient le répertoire racine
- Contient la liste des entrées du répertoire
- Un entrée permet de récupérer à partir d'un nom :
  - le contenu du fichier
  - des méta-informations (dates, droits, etc.) sur le fichier
  - FAT ne permet pas une gestion multi-utilisateurs des droits



Attributs : répertoire, lecture seule, cachée, etc.

L'exemple de VFAT

- Noms longs nécessitent des entrées particulières
- Pour compatibilité ascendante :/dev/hdb4 /zip vfat defaults,noauto,user 0 0
  - deux noms pour un fichier ; un nom FAT (8+3) et éventuellement un nom long
  - nom FAT = nom long tronqué + ~1, ~2, ...
  - nom long en unicode stocké dans plusieurs entrées précédant nom court
  - attribut indique que c'est une partie de nom long



Points de montage

- Un système de fichiers par partition
- Points de montage regroupent plusieurs systèmes de fichiers en un seul
- Le fichier `/ets/fstab` définit les points de montage par défaut

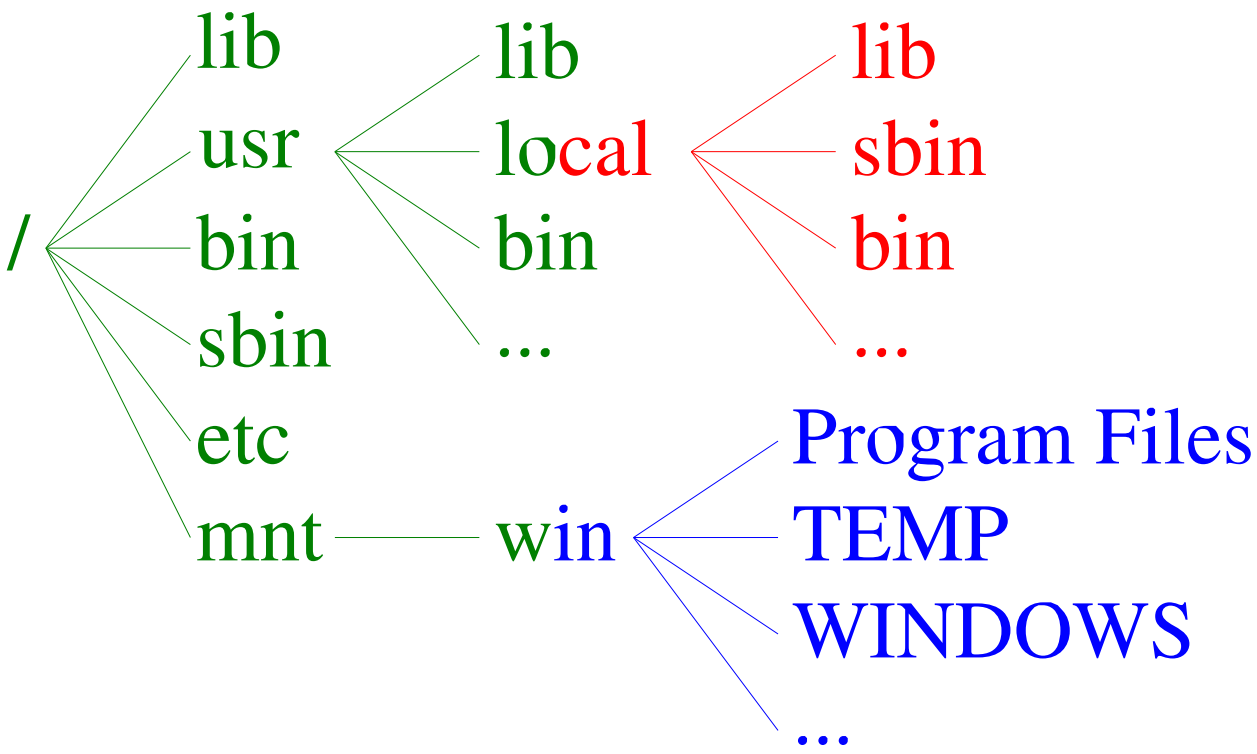
```
/dev/hda8 /usr/local ext3 defaults 0 2
/dev/hda9 /          ext3 defaults 0 2
/dev/hda2 /mnt/win   vfat defaults 0 2
```

- Commande `mount` utilisée monter une partie du système de fichiers

```
# mkdir /mnt/c
# mount -t ntfs -o ro /dev/hda2 /mnt/c
```

- Permet également de visualiser les partions montées

Points de montage



Vérification d'intégrité

- Opérations sur fichier non atomique
- Plusieurs blocs mis à jour en mémoire puis sur le disque
- Suite d'opérations non atomiques
- Panne au milieu des opérations disques = incohérences
- Vérificateur tente de récupérer ce qu'il est possible de récupérer
- Nécessité de parcourir tout le disque : temps de vérification proportionnel à sa taille
- `fsck`, `chdsk`, vérifie que :
  - les blocs libres sont libres en parcourant l'arborescence
  - les blocs utilisés sont attachés à des fichiers
  - les blocs n'appartiennent qu'à un seul fichier
- Un bloc est modifié au démarrage et à l'arrêt normal pour dire qu'il n'y a pas eu d'arrêt brutal

Journalisation

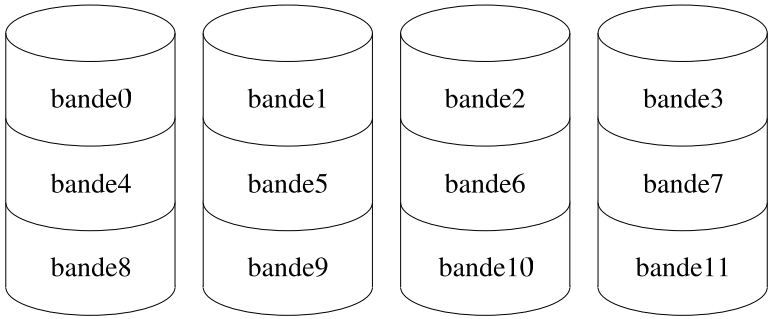
- Idée des transactions en base de données
- Problème d'atomicité
- Commencer par écrire ce qui doit être modifé (journal)
- Faire les modifications
- Effacer ce qui doit être modifié
- Si on arrête avant d'avoir fini d'écrire ce qui doit être modifié : système cohérent, rien à faire
- Si on arrête avant d'avoir effacer ce qui doit être modifié : on rejoue ce qui doit être modifié
- Sur-coût à l'exécution
- Vérifications au redémarrage non liées à la taille du disque

RAID

- *Redundant Array of Inexpensive Disks* (RAID) par opposition à *Single Large Expensive Disk* (SLED)
- Utiliser plusieurs disques pour en simuler un disque unique
- Améliore les performances
- Améliore la fiabilité
- Diminue le coût
- Réalisé de façon matérielle ou logicielle

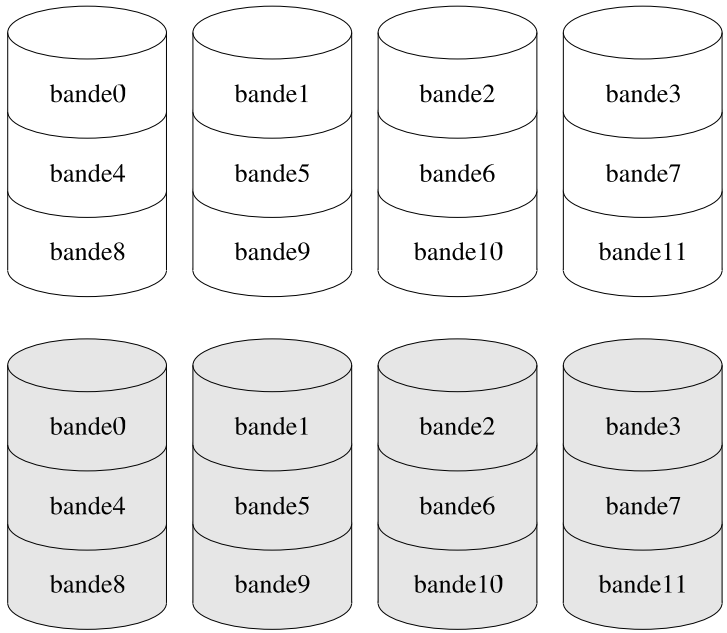
RAID niveau 0

- *Stripe* mode (agrégat de bande)
- Paralléliser les requêtes de lecture-écriture
- Utile si requêtes de grande taille
- Pas de fiabilité



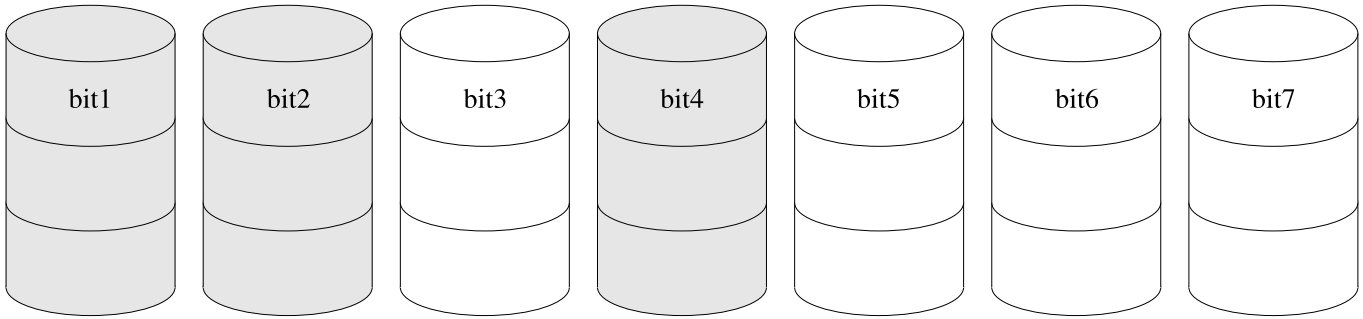
RAID niveau 1

- Miroir
- Une écriture en parallèle sur deux disques
- Deux lectures en parallèle sur les disques



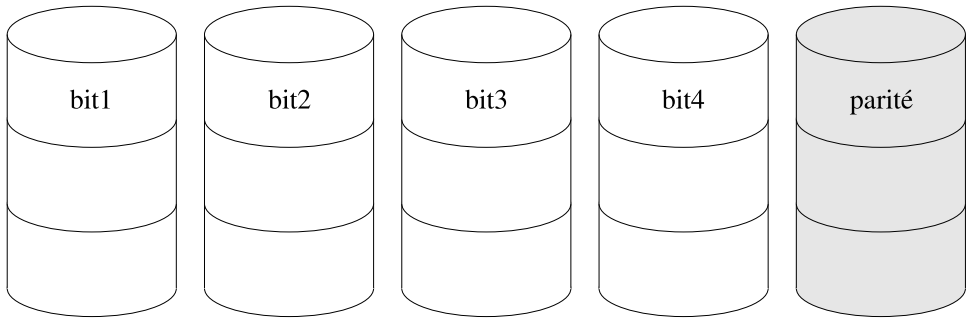
RAID niveau 2

- Travail au niveau octet
- Utilisation du code de Hamming (4/7)
- Disques synchrones = matériel



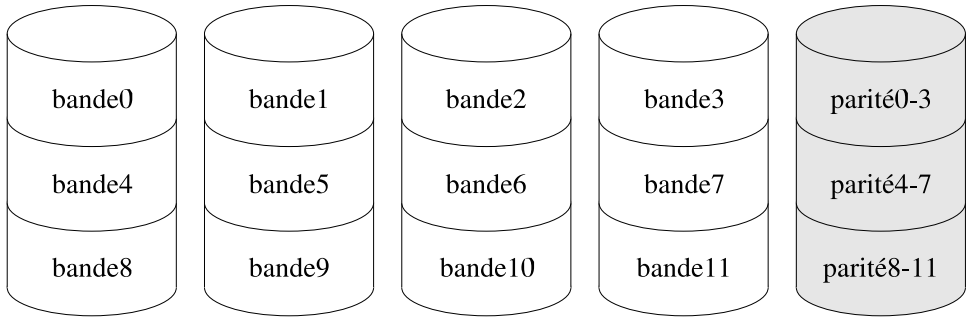
RAID niveau 3

- Travail au niveau octet
- Utilisation d'un bit de parité
- Disques synchrones = matériel



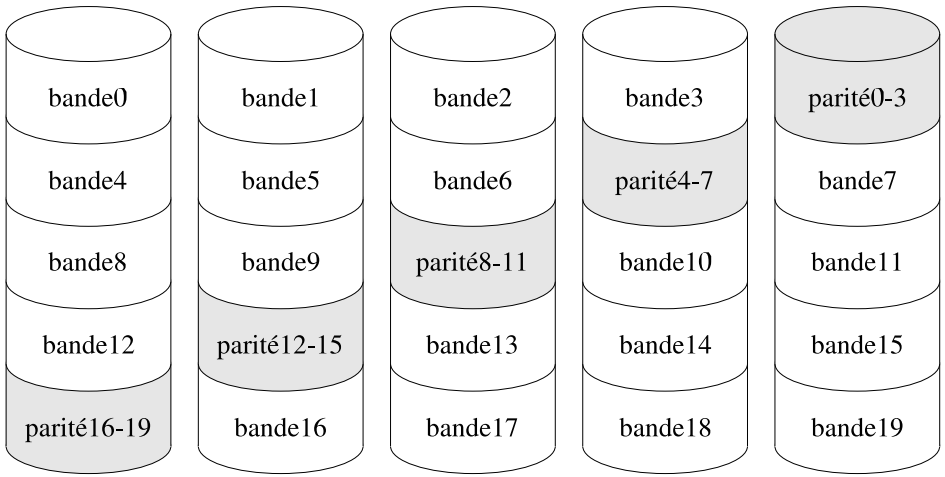
RAID niveau 4

- Travail au niveau bloc
- Bloc de parité
- Si on modifie un secteur soit on relit :
  - tous les autres disques sauf celui de parité
  - les anciennes données du bloc et l'ancien bloc de parité
- Puis on écrit le bloc de parité



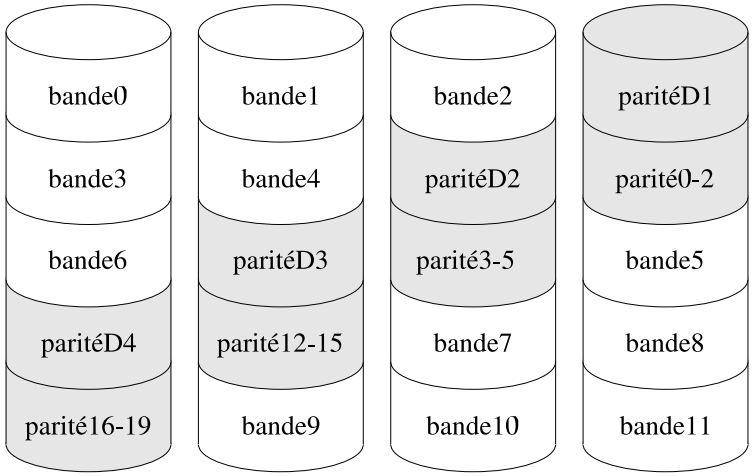
RAID niveau 5

- Distribution du bloc de parité pour améliorer les performances
- Reconstitution du disque après panne complexe



RAID niveau 6

- RAID 5
- Avec bloc de parité par disque



RAID niveau 10, 0+1 et 50

- 10 disques de RAID0 constitués de miroirs RAID1
- 0+1 miroir RAID1 du disque RAID0 (perte d'un disque implique passage en RAID0)
- 50 (3+0) disques de RAID0 constitués de disques RAID3