IUT - Département Informatique

ASR3 Système 2013-2014

Gestion des fichiers

Michel Billaud - septembre 2013

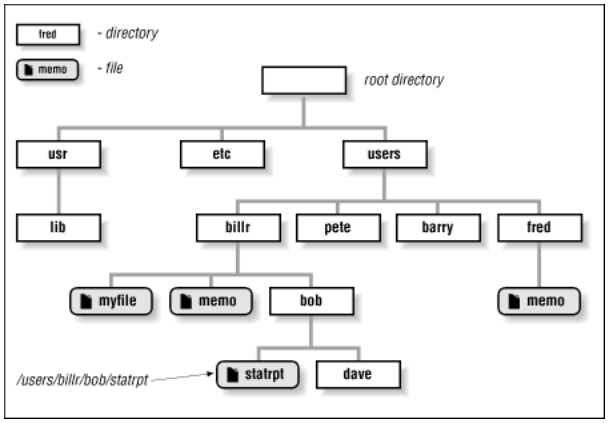
SYS FICHIERS

Table des matières

1 Les fichiers

Qu'est-ce qu'un fichier ?

- Un fichier est une suite de données structurées codées sur un support.
- Les fichiers sont la plupart du temps conservés sur des *mémoires de masse* tels que les disques durs.
- Les fichiers sont classés dans des *répertoires*, chaque répertoire peut contenir d'autres répertoires, formant ainsi une *organisation arborescente* appelée *système de fichiers*.



un système de fichiers

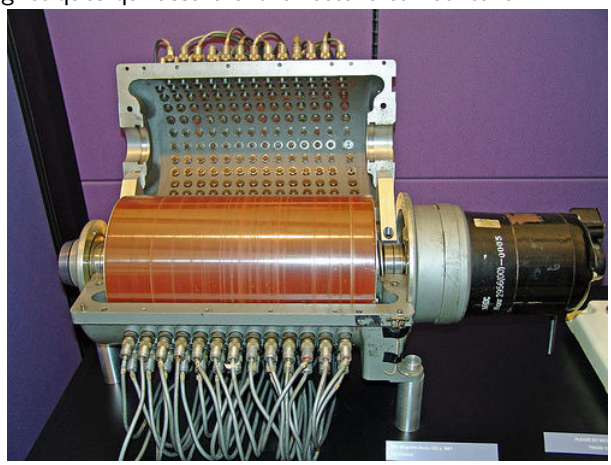
Autrefois les systèmes d'exploitation prenaient en charge différents types de fichiers :

- les fichiers de texte, composés de lignes de taille variable ;
- les fichiers séquentiels, composés d'enregistrements
- les fichiers relatifs, contenant des enregistrements accessible directement par leur numéros (le système d'exploitation fournit des fonctions d'accès du type : "lire l'enregistrement numéro 25")
- les fichiers indexés, dont les enregistrements comportent une *clé* (par exemple la plaque d'immatriculation pour un fichier de cartes grises). On peut alors demander à lire l'enregistrement concernant la plaque "1234AB56".
- ...

De nos jours, la plupart des systèmes connaissent un seul type : les fichiers sont une séquence d'octets, avec des opérations (`read, write`) pour lire et écrire à partir d'une position courante, et se positionner (`seek`) à un endroit déterminé (numéro d'octet). Les fonctionnalités des fichiers indexés, par exemple, sont fournies par des bibliothèques spécialisées (voir par exemple la page de manuel `dbopen`).

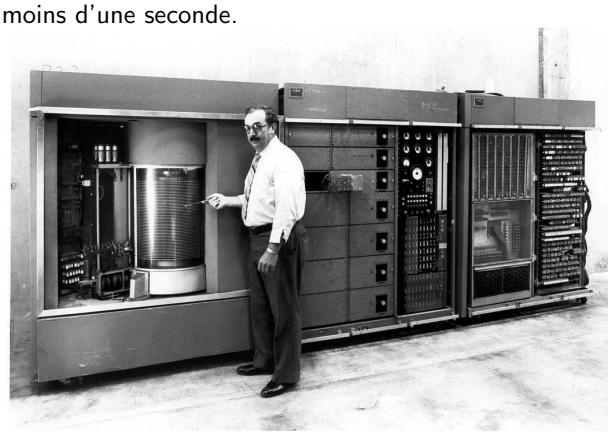
2 Les disques

Les disques magnétiques que nous connaissons dérivent des mémoires à tambours magnétiques, des cylindres en rotation couverts d'un enduit ferro-magnétiques, avec des têtes magnétiques qui assuraient la lecture et l'écriture.



ancêtre : le tambour magnétique

En 1953, un ingénieur d'IBM a eu l'idée de superposer des plateaux sur un axe, et d'ajouter une tête de lecture-écriture mobile, portée par un bras. Le bras pouvait changer de disque en moins d'une seconde.



Prototype d'IBM

La production commercale du RAMAC 305 (Random Access Method of Accounting and Control) a commencé en juin 1957.

Caractéristiques :

- 59 disques de 24 pouces
- 2 têtes pouvant se déplacer d'un plateau à l'autr
- capacité totale de 5 megaoctets.
- prix de 10000 dollars par megaoctet.

De nombreuses améliorations ont été apportées :

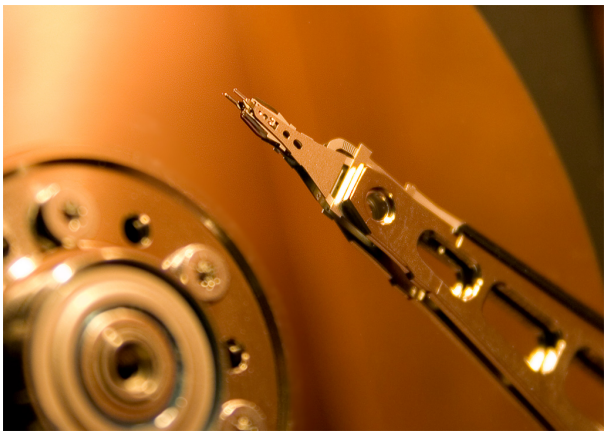
- l'utilisation de plusieurs têtes de lectures portées par un même bras (une par face de disque) dispense de déplacer la tête d'un disque à l'autre.
- un coussin d'air permet de "faire flotter" les têtes en évitant qu'elles rentrent en contact avec le disque, incident appelé "atterrissage" qui endommage irrémédiablement la surface magnétique.

L'augmentation de la précision mécanique du mouvement des têtes et de la vitesse de rotation du disque (entre 3600 et 15000 tours/mn) a permis d'augmenter la capacité : 5 Mo en 1956, 1 Go en 1982, 25 Go en 1998, 500 Go en 2005, 2 To en 2009, 4 To en 2011.

Une unité de disque récente (80GB), début du siècle

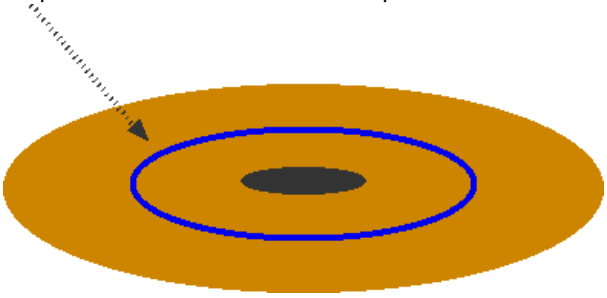
De nos jours, on trouve des disques de capacité 4 To en format 3 pouces et demi (Hitachi 7K4000).

2.1 Disque, tête



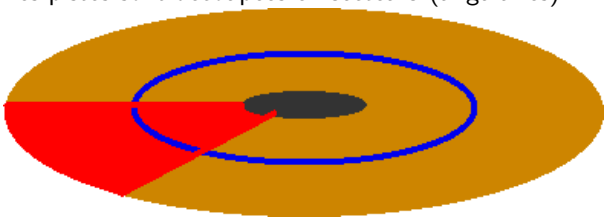
Disque, tête de lecture, bras

La position d'une tête définit une *piste*



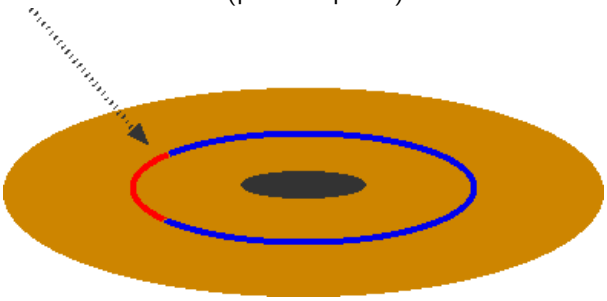
Une piste

Les pistes sont découpées en *secteurs* (angulaires)



Un secteur

Un secteur contient (pour simplifier) un *bloc de données*.



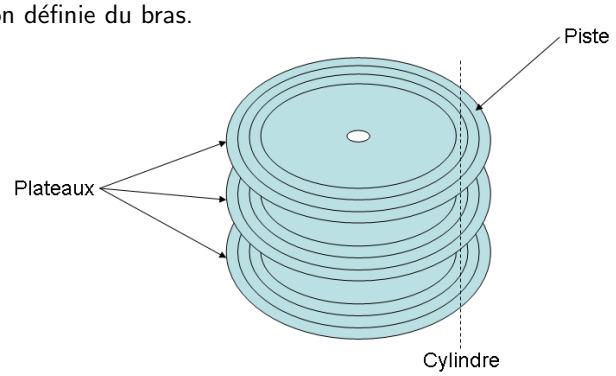
Un bloc

Changer de piste implique un *mouvement* de la tête, changer de secteur une rotation.

2.2 Avec plusieurs plateaux

Un axe, un bras, plusieurs plateaux

On appelle *cylindre* l'ensemble des pistes pour une position définie du bras.



Cylindre = pistes
pour une position du bras

- Accéder à un autre cylindre implique un mouvement du bras, qui est lent (à l'échelle de l'ordinateur)
- Changer de tête se fait par commutation électroinique d'une tête à l'autre.

À retenir : le disque magnétique est un médium à *temps d'accès non uniforme* : le délai pour accéder à une information dépend de la position de cette information, et de la position courant des têtes de lecture.

Dans ce délai il faut intégrer

- le déplacement du bras (sélection du cylindre)
- la rotation (sélection du bloc)

Le temps de commutation (sélection de la piste) est négligeable.

Le délai le plus long vient du mouvement de la tête. Il en résulte un principe de localité : pour minimiser les déplacements de tête, et donc les temps d'accès, il vaut mieux grouper les données sur des cylindres proches.

2.3 Quelques chiffres

couche magnétique	10-20 nm
vitesse de rotation	5400-15000 tours/min
capacité	120 GB -2 TB
temps d'accès	2-15 ms
vitesse de transfert	70 MB/sec (7200 t/min)

3 Gestion des entrées/sorties sur disque

3.1 Sur un système monotâche

Sur un système monotâche : il n'y a pas (trop) de problème, on déplace la tête à la demande là où on en a besoin.

Bien évidemment les performances sont meilleures si les blocs d'un fichier sont regroupés. Il est donc prudent, parfois, de *défragmenter* les systèmes de fichier pour regrouper les blocs.

Autrefois on profitait de la sauvegarde pour copier le disque original, fichier par fichier, sur un disque vierge, ce qui regroupait les blocs. Le disque original était alors démonté et conservé comme sauvegarde.

Bien entendu, une telle sauvegarde se faisait dans le cadre d'une maintenance planifiée : pendant quelques heures l'ordinateur était inaccessible aux utilisateurs.

3.2 Sur un système multitâche

Sur un système multitâches il est possible que plusieurs processus soient bloqués pour avoir demandé une opération sur le même disque. Il y a donc plusieurs demandes en attente : laquelle servir en premier ? C'est le problème d'*ordonnancement des mouvements du bras du disque*.

3.3 E/S : Premier arrivé, premier servi

Une première politique peut être facilement imaginée : exécuter les demandes dans l'ordre où elles ont été effectuées.

Exemple : trois processus qui lisent et traitent des données situées dans des blocs consécutifs : P1 commence au bloc 100, puis 101, 102, etc. Idem pour P2 et P3 qui commencent respectivement à 200 et 500.

Déroulement :

- P1 demande la lecture du bloc 100, qui commence à s'exécuter ; P2 demande le bloc 200, P3 le bloc 400
- la lecture se termine, la lecture du bloc 200 s'exécute, P1 demande le bloc 101.
- la lecture se termine, la lecture du bloc 500 s'exécute, P2 demande le bloc 201.
- le lecture du bloc 101 commence, P3 demande le bloc 501, ...
- etc.

Successivement, on lit donc les blocs 100, 200, 500, 101, 201, 501, 102, ...

Cette politique "premier arrivé - premier servi" est

- *simple* à mettre en oeuvre
- *équitable* ...
- mais *pas efficace*

3.4 E/S : plus court déplacement

Cette politique consiste à exécuter la requpête qui demandera le moindre déplacement.

Sur le même exemple, après avoir lu les blocs 100 et 200 on a en attente des demandes pour 101 et 500. La position 101 est la plus proche de la position courante (200), on retourne donc à P1.

Il en résulte l'ordre suivant : 100, 200, 101, 201, 102, 202 ... et on constate que les lectures de P3 ne sont pas faites en raison de la coalition entre P1 et P2.

En résumé, la politique "déplacement le plus court" est

- *simple* à mettre en oeuvre
- *efficace* ...
- mais *pas équitable*

3.5 Politique de l'ascenseur

La politique dite "de l'ascenseur" est un compromis intéressant.

L'algorithme de gestion est à deux phases

- dans la phase montante, on traite la requête qui concerne le plus petit de bloc strictement supérieur à la position courante. SI il n'y en a pas, on passe en phase descendante.
- dans la phase descendante, on traite la requête pour le plus grand numéro de bloc strictement inférieur à la position courante. Si il n'y en a pas on passe en phase montante.

Déroulement :

- (⇑) 100, 200, 500,
- (⇓) 201, 101,
- (⇑) 202, 501,
- (⇓) 203, 102,
- (⇑) 204, 502
- (⇓) ...

La politique de l'"ascenseur" est

- *assez équitable* ...
- *assez efficace*

3.6 E/S : Deadline driven disk scheduler

Une idée plus moderne : on fixe un délai maximum, après lequel l'exécution d'une requête devient urgente.

Algorithme :

- si il y existe des requêtes urgentes, on traite la plus ancienne (qui est aussi la plus urgente)
- si il n'y en a pas, on traite la requête qui demande le moins de mouvement.

La politique "deadline driven scheduling" est

- *efficace*
- et *équitable*
- ... et brevetée

Exercice 1. Traduire en français correct le titre du brevet *United States Patent 5787482* de 1998 :

```
Subject: Deadline driven disk scheduler
method and apparatus with thresholded most
urgent request queue scan window
```

3.7 Conclusion

- Une problématique spécifique, par rapport à la gestion des données en mémoire, est de tenir compte du temps d'accès aux données, qui n'est pas uniforme.
- Il y a un compromis à trouver entre *efficacité* et *équité*
- quelques heuristiques simples pour l'ordonnancement :

1. premier arrivé, premier servi
2. plus court déplacement
3. ascenseur
4. *deadline scheduling*

4 Systèmes RAID

4.1 Principe et objectifs

Les systèmes RAID (*Redundant array of inexpensive disks*) sont composés de plusieurs disques groupés dans un boîtier ou un *rack*[1].

1. Boîtier que l'on boulonne dans une armoire informatique de taille normalisée

un boitier RAID

un rack RAID

L'objectif initial était de réaliser des systèmes de stockage de grande capacité en combinant des "petits" disques bon marchés, plutôt qu'en achetant de gros disques, qui n'étaient disponible que sur marché du matériel professionnel, et donc beaucoup plus coûteux.

On verra que grouper des disques permet non seulement d'étendre la *capacité*, mais aussi d'améliorer la *fiabilité* (par l'introduction d'une redondance) et les *performances*, en faisant travailler les disques en parallèle.

Actuellement on traduit RAID par Redundant Array of *Independent* Disks, les systèmes n'étant pas forcément composés de disques bon marché.

Connexion des disques

- les disques peuvent être connectés de la manière habituelle, et le RAID géré par des modules spécifiques du système d'exploitation
- les disques peuvent être raccordés à une carte d'extension qui prend en charge les fonctions du RAID
- la carte mère peut intégrer la fonction RAID,
- les disques peuvent également être intégrés à un boitier (ou rack) extérieur, qui se comporte comme un disque unique.

Il existe un grand nombre de manières de combiner plusieurs disques physiques pour réaliser un système de stockage.

4.2 NRAID (JBOD)

Le degré zéro du RAID consiste à "concaténer" de petits disques pour un faire un gros. Par exemple, avec 4 disques de 500 Go on obtient un système de 2 To. Les premiers secteurs du système sont sur le disque 1, les suivants sur le disque 2, etc.

Ceci permet de répartir les données, mais n'assure aucune redondance, d'où les noms usuels NRAID et JBOD

- NRAID *is Not RAID*
- JBOD *just a bunch of disks*

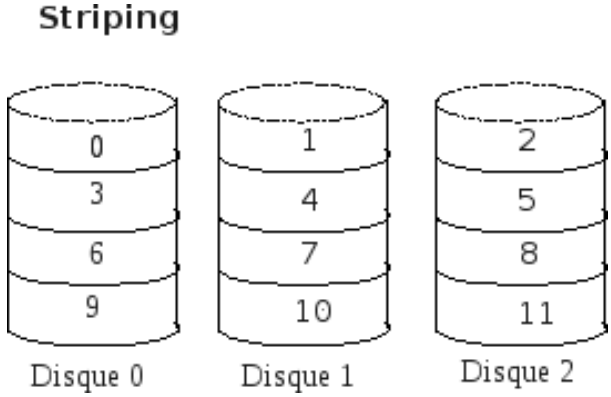
Exercice 2. Les disques sont-ils nécessairement de même taille ?

On notera qu'il est possible de mener plusieurs requêtes d'entrées-sorties simultanément sur le système, si par chance elles concernent des disques différents. Ceci a un impact positif sur les performances. [2].

En cas de défaillance d'un disque, les données sont compromises. Les fichiers qui ont des blocs sur le disque défaillant sont irrécupérables.

4.3 RAID 0 : agrégation par bandes

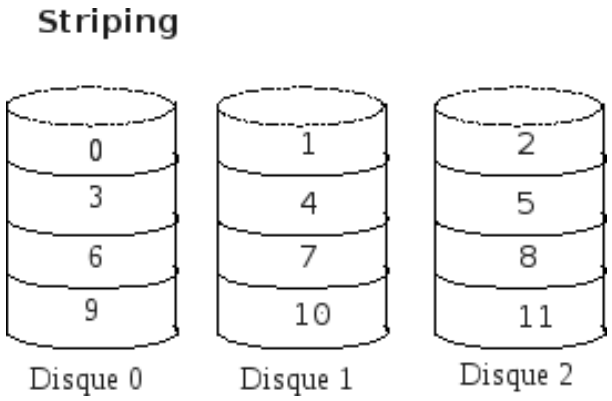
Le RAID 0, ou striping, combine des disques de même taille. A la différence du NRAID, les données sont réparties par *bandes* : le prermier bloc du système est sur le premier disque, le second sur le second disque etc.



RAID 0 : agrégation par bandes

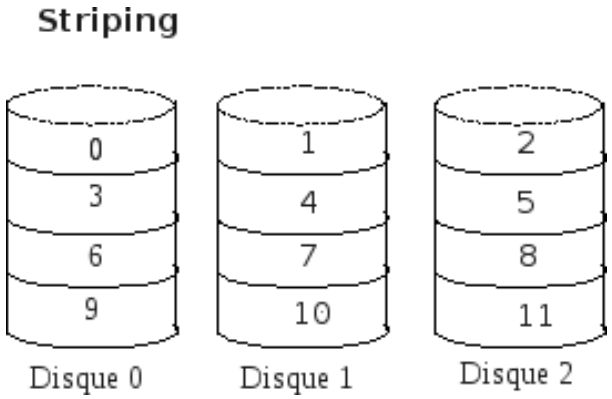
Ceci permet une meilleure répartition de la charge de travail, en tenant compte également du principe de localité : après avoir lu un bloc on a très probablement besoin du suivant ; il est donc possible, en parallèle, de le lire de façon anticipée.

2. par rapport à l'utilisation d'un disque unique de 2 To



Demande de lecture 7 :
possibilité de *lecture anticipée* de 8 et 9
dans le même temps

On ne peut pas anticiper les écritures, mais plusieurs écritures en même temps, à condition qu'elles se déroulent sur des disques différents.

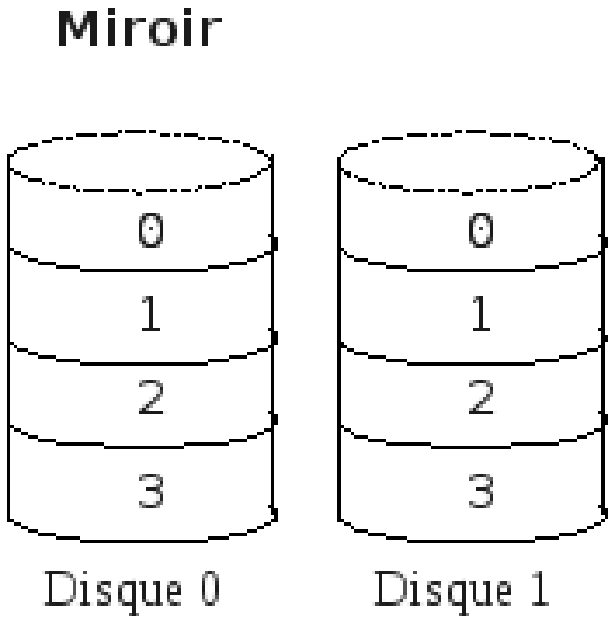


exemple
lecture/écriture de 2, 6 et 11

En cas de défaillance d'un disque les données sont irrécupérables.

4.4 RAID 1 : miroir

Dans une configuration en miroir, les données sont stockées en double (ou plus).



RAID 1 : miroir

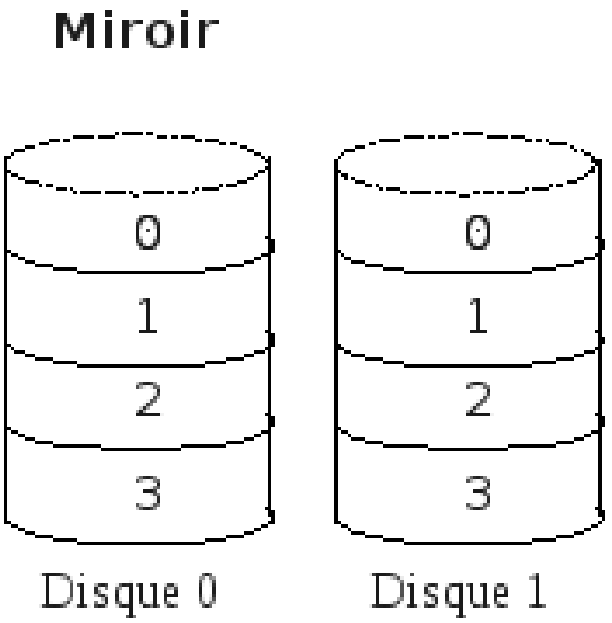
Avec 2 disques de 500 Go chacun, on fait donc un sytème RAID 1 de 500 Go.

L'objectif est bien sûr d'améliorer la redondance : si on des disques tombe en panne, on peut retrouver les données sur l'autre disque.

Exercice 3. Si les disques ont une probabilité de panne de 5/100 par an, quelle est la probabilité de perdre les données dans un système RAID de 2 disques ?

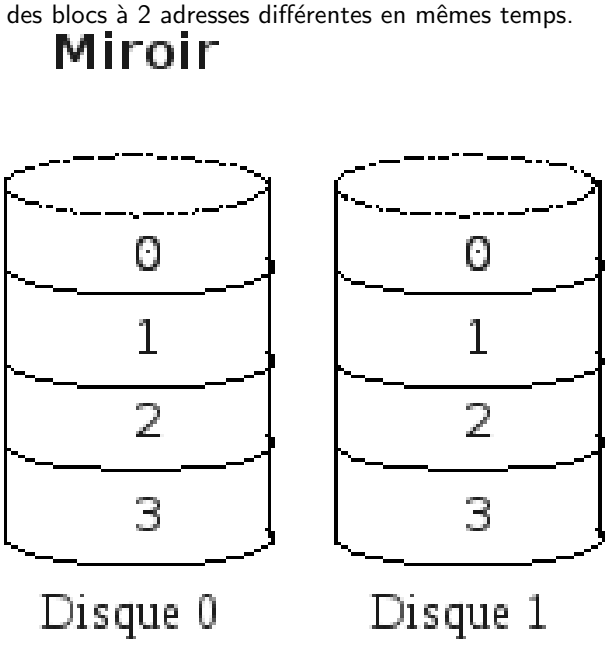
Exercice 4. Peut on faire du RAID 1 avec 3 disques ?

Toute opération d'écriture sur le système se traduit obligatoirement par des écritures tous les disques, menées simultanément, ce qui ne prend pas plus de temps qu'une seule écriture.



Écriture : doit se faire en parallèle

Par contre, puisque les données sont en double, on peut lire des blocs à 2 adresses différentes en mêmes temps.



Lecture : deux à la fois

Globalement, on peut donc mener deux fois plus de lectures qu'avec un seul disque : il y a donc un gain de performances en lecture.

4.5 Disques de rechange, reconstruction

Un boitier RAID possède en général des emplacements supplémentaires pour des disques de rechange (*spare disks*).

Par exemple un boitier en RAID 1 à deux disques peut avoir 3 emplacements. Normalement 2 disques (D1, D2) sont actifs et fonctionnent en miroir, le dernier D3 est arrêté.

Si un problème est détecté sur un disque actif - par exemple D2 - le système de disques peut encore fonctionner en mode dégradé (le fonction miroir n'est plus assurée) sur D1. Le boitier procède alors à la *reconstruction* automatique du miroir : il met alors automatiquement en route le disque D3, et y fait une copie du disque D1 qui est encore valide. Quand cette copie est terminée, le système fonctionne à nouveau en miroir sur D1 et D3.

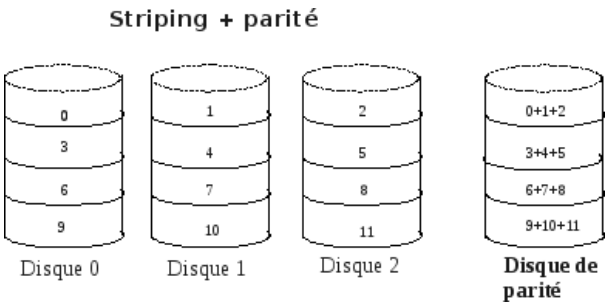
En général le boitier possède aussi une interface réseau, et fait remonter une alerte à l'administrateur pour qu'il remplace rapidement le disque défectueux par un nouveau disque de rechange.

4.6 RAID 4 : agrégation par bandes avec parité

Le principe du RAID 4 est une amélioration du principe du miroir. Au lieu de dupliquer les données, on utilise un système de "parité" qui permettra de reconstituer le disque manquant.

Partons d'un système RAID 0 à trois disques A, B, C : les trois premiers blocs du système sont sur les premiers blocs des disques, les trois blocs suivants sur les seconds blocs etc. Ajoutons un quatrième disque D, sur lequel nous stockerons, dans le Nième bloc, un ou-exclusif des Nièmes blocs des 3 disques de données.

$$D_n = A_n \oplus B_n \oplus C_n$$



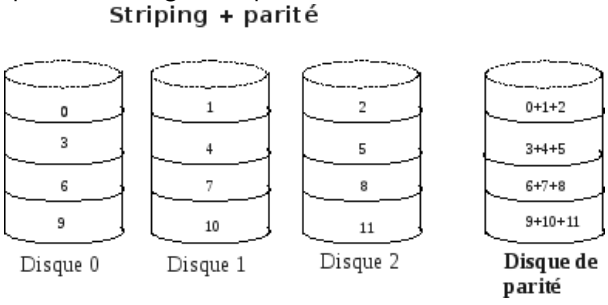
RAID 4 : agrégation par *bandes* avec*parité*

Si le disque A tombe en panne, il sera possible de reconstituer son contenu. En effet, les propriétés de l'algèbre de Boole font que le bloc A_n est le ou-exclusif des autres blocs de la même "bande" :

$$A_n = B_n \oplus C_n \oplus D_n$$

Le RAID 4 permet donc d'avoir de la redondance sans doubler la taille totale des disques. Les systèmes usuels ont 4 ou 5 disques, la place disque nécessaire à la redondance est donc de l'ordre de 20 ou 25%.

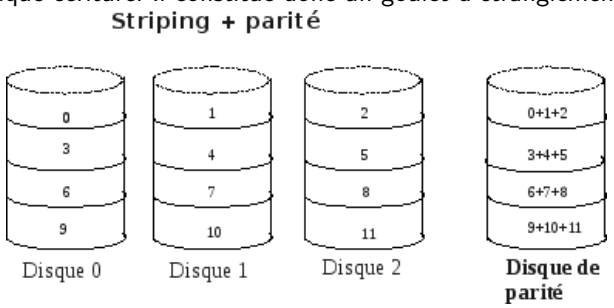
Il est possible de mener de front plusieurs lectures, d'où une possibilité de gain de performances.



Lectures en parallèle

Par contre, pour modifier un bloc, il faut modifier aussi le bloc "de parité", et pour le calculer, il faut avoir lu les blocs de la bande. Une écriture peut donc prendre le temps de deux accès disque.

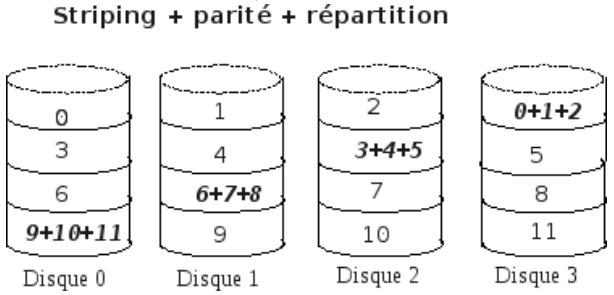
Remarquez aussi que le disque de parité est sollicité à chaque écriture. Il constitue donc un goulet d'étranglement.



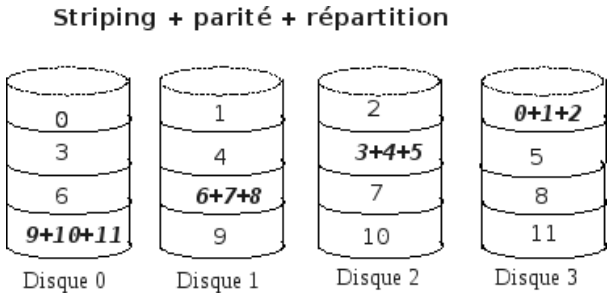
goulet d'étranglement en écriture
(accès au disque de parité)

4.7 RAID 5 : agrégation par bandes avec parité répartie

Le RAID 5 est une amélioration du RAID 4 pour mieux répartir la charge : selon les bandes, le bloc de parité n'est pas situé sur le même disque.



RAID 5 : agrégation par bandes avec *parité répartie*



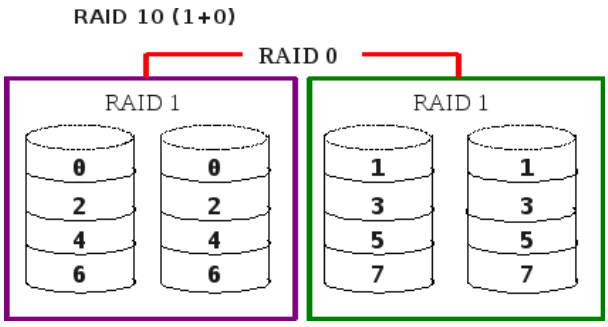
la charge en écriture est répartie

Ceci conduit à de meilleurs performances.

4.8 Combinaisons : RAID 10

Les techniques vues précédemment sont combinées entre elles pour combiner, selon les besoins, redondance et performances.

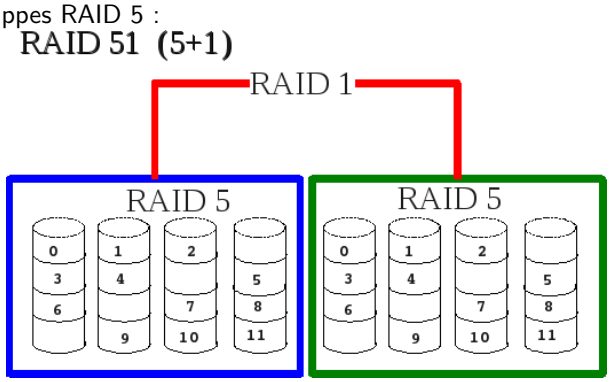
Par exemple, le RAID 10 (ou RAID 1+0) est une "grappe" RAID 0 (striping) de paires de disques en miroir :



RAID 1 sur disques en miroir

4.9 RAID 51

De la même façon, le RAID 51 est un miroir de deux grappes RAID 5 :



miroir entre 2 *grappes RAID 5*

4.10 RAID 50

Laissé en exercice :

Exercice 5. Dessinez un système de stockage RAID 50. Comparez performance et fiabilité avec le RAID 51

4.11 Choix d'un système RAID

Les systèmes RAID servent à réaliser des systèmes de stockages de données, avec des objectifs de capacité, de fiabilité et de performances qui sont supérieurs à ceux des disques individuels. C'est une technologie qui est de nos jours indispensable en environnement professionnel.

Les objectifs sont atteints en combinant diverses techniques : la duplication, la répartition, la parité.

De multiples combinaisons sont possibles : pour faire le bon choix dans un contexte donné, il faut les comprendre.

5 Systèmes de fichiers

Les *systèmes de fichiers* sont des structures de données, stockées sur disque, qui représentent des fichiers, des répertoires etc.

Vu par l'utilisateur du système (c'est à dire le programmeur d'applications), un fichier possède un contenu, et des méta-données :

– sa taille
– son propriétaire
– les droits d'accès
– la date de création
– date de dernier accès
– ...

5.1 Fonctions du SGF

Le Système de Gestion de Fichiers (SGF) contient des fonctions pour

– Manipulation des fichiers : créer/détruire des fichiers, ...
– Allouer de la place sur mémoires secondaires
– Localiser des fichiers : accès au contenu
– Sécurité et contrôle des fichiers
– Fiabilité en cas de panne
– ...

On s'intéresse ici à la représentation des fichiers et des répertoires sur les disques.

5.2 Catalogue de fichiers

Les premiers disques avaient une capacité très faible (quelques méga octets). Selon un principe constant en histoire de l'informatique,

faire du vieux avec du neuf

ils ont été d'abord gérés en utilisant les méthodes issues des technologies précédentes, à savoir les bandes magnétiques.

Les bandes magnétiques sont un support à accès essentiellement séquentiel (il existe quand même un moyen de "bobiner" plus vite, en avant ou en arrière) sur lequel les fichiers sont stockés les uns après les autres. En début de bande, on met un premier petit fichier, qui contient le nom de la bande, et la liste des fichiers présents.

Les premiers disques étaient gérés de la même façon (VTOC = Volume Table of Contents, d'IBM) : au début du disque se trouve une table, qui contient les noms, positions de départ (numéro de bloc) et taille de chaque fichier. On trouve aussi la liste des espaces inutilisés, pour pouvoir allouer de nouveaux fichiers.

Cette manière de faire possède tous les inconvénients que l'on connait pour la gestion de la mémoire par blocs contigus : il se produit fatalement une certaine fragmentation, et il est difficile d'allonger un fichier existant.

Par contre, pour une utilisation en mono-tâche les performances sont très bonnes : les données d'un même fichier sont proches les unes des autres, ce qu minimise le temps d'accès ?

A noter : avec des disques de taille aussi réduite, il n'apparait pas nécessaire d'avoir une organisation hiérarchisées sous forme de répertoires.

5.3 FAT : file allocation table

Une autre solution est d'utiliser deux tables

– une table d'allocation qui indique les méta-données de chaque fichier, et la position (index) de son premier bloc ;
– une table d'index donnant, pour chaque bloc, la position de son successeur.

La table d'index est chargée en mémoire au montage du disque.

Ceci permet de gérer les fichiers de manière similaire à la mémoire paginée : les blocs d'un fichier ne sont pas contigus, ce qui fait qu'on ne rencontre pas le problème de "gruyérisation".

Par contre, la dispersion sur le disque des blocs d'un même fichier peut avoir un impact important sur les performances, parce que le disque est un support à temps d'accès non-uniforme.

Il y a donc diverses stratégies au niveau du système pour mimimiser la dispersion. Par exemple le disque sera géré par zones, le système prenant soin d'essayer garder un certain pourcentage de place libre dans chaque zone, pour pouvoir y loger de préférence les blocs supplémentaires des fichiers qui font déjà partie de la zone.

Exercice 6. Soit un disque de 10 Mo, géré par blocs de 2 Ko. Quelle serait la taille de la table d'index ?

5.4 Tables des blocs Unix

Unix est une vaste famille de système d'exploitations, qui supporte une multitude de types de systèmes de fichiers.

Exercice 7. Sur les machines Linux regardez dans `/lib/modules/*/kernel/fs` la liste des pilotes qui prennent en charge les systèmes de fichiers. Recherchez à quoi correspondent `aufs`, `jfs`, `ntfs` ?

Dans ce qui suit nous décrivons les principes utilisés dans les premiers systèmes Unix : l'espace disque comporte deux zones

– les méta-données
– les blocs de données (contenu des fichiers)

À chaque fichier est associé un *i-node* (noeud d'information)

– qui contient des attributs (taille, propriétaire, droits...)
– qui permet de retrouver les adresses de ses blocs de données

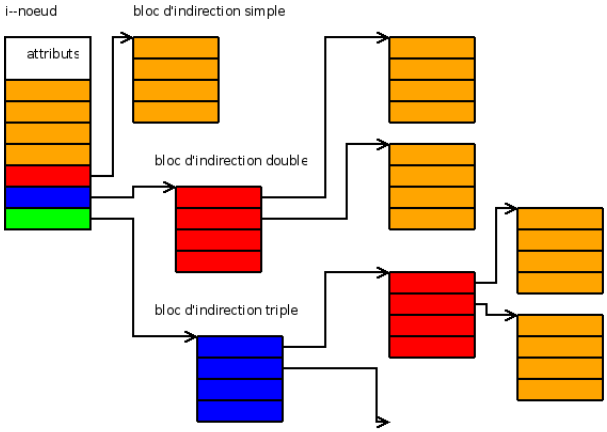
Un fichier n'a pas de nom intrinsèque, il est repéré par son numéro d'i-node. Nous verrons plus loin que ce sont les répertoires qui servent à faire la correspondance entre des noms et des numéros d'i-nodes.

Pour pouvoir retrouver les données, l'i-node contient

– l'adresse des premiers blocs du fichier, ce qui suffit pour les petits fichiers
– l'adresse d'un *bloc d'indirection simple* qui contient d'autres adresses de blocs de données.

3. qui contiennent d'autres adresses de blocs de données.

– l'adresse d'un *bloc d'indirection double* qui contient des adresses de blocs d'indirection simple [3]
– l'adresse d'un *bloc d'indirection triple* qui contient des adresses de blocs d'indirection double [4]



I-nodes et blocs d'indirection

Un exemple Cette structure de données, qui peut paraitre un peu baroque, permet d'accéder efficacement au début d'un fichier indépendamment de sa longueur.

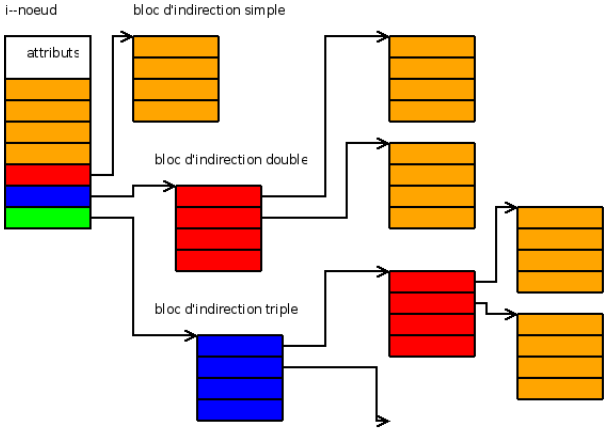
Supposons le cas d'une machine avec

– un disque avec des blocs de 4 Ko (2^{12})
– des entiers sur 32 bits

En théorie, un système de fichiers, avec des numéros de blocs sur 32 bits, peut contenir jusqu'à 4 milliards (2^{32}) de blocs, ce qui représente un espace de $2^{32} \times 2^{12} = 2^{44}$ octets, soit 16 To.

Le système de fichiers a donc une capacité maximale de 16 To. Voyons maintenant la taille maximale d'un fichier :

– les numéros de blocs sont sur 32 bits, donc 4 octets
– un bloc d'indirection simple est de la taille d'un bloc du disque soit 4 Ko. Il peut donc contenir les adresses de 1024 blocs de données (2^{10}), soit 4 Mo de données.
– un bloc d'indirection secondaire permettra de références les 2^{20} blocs suivants (4 Go), et le bloc tertiaire 2^{30} blocs (4 To).



Pour la plupart des accès, une indirection suffit

5.5 Représentation des répertoires

Plusieurs approches sont possibles pour la représentation des répertoires. On peut les considérer comme une extension de la notion de table des fichiers, ou bien les regarder comme des fichiers spéciaux.

5.5.1 Catalogues de fichiers

Avec cette approche (CP/M, MS/DOS, Windows...), les "catalogues" sont matérialisés par des entrées spéciales de la table des fichiers qui renvoient vers d'autres parties de la table.

Ceci interdit les *liens* comme on les connait sous Unix, c'est à dire qu'un même fichier soit visible à des endroits différents de l'arborescence. En effet, chaque entrée du catalogue correspond à des méta-données distinctes ; dupliquer des méta-données conduirait rapidement à des incohérences lors de la modification d'un fichier.

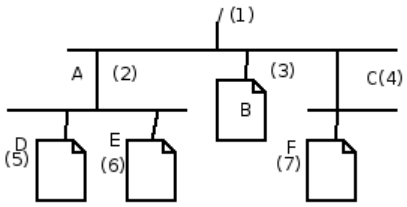
5.5.2 Comme des fichiers de données

C'est la représentation utilisée par UNIX : chaque i-node contient un indicateur qui précise le type de l'objet, qui peut être

– un fichier normal. Les blocs de données correspondent au contenu du fichier.
– un répertoire. Dans ce cas les blocs contiennent une table de correspondance entre les noms des objets présents dans ce répertoire, et leurs et numéros d'i-node
– un lien symbolique. L'i-node contient le chemin d'accès.
– un périphérique,
– ...

Exemple Table des i-nodes

4. qui contiennent des blocs d'indirection simple qui contiennent d'autres adresses de blocs de données.



N°	type	CR	contenu des blocs
1	d	4	..→1, .→1, A→2, B→3, C→4
2	d	2	..→1,.→2, D→ 5, E→6
3	f	1	"coucou"
4	d	2	$..rightarrow$1, .→4, F→7
		...	

Le compteur de références (CR) indique combien de fois un objet est cité dans les répertoires. Quand il n'est plus cité, il est inaccessible et donc on peut récupérer la place qu'il occupe.

Exercice 8. Sur ce schéma, étudiez l'effet succcessif des commandes
- `ln /B /A/G`
- `rm /B`
- `rm /C/F`
- `mkdir /C/X`

Gestion des blocs libres Le système possède
- une liste des blocs libres
- un tableau de marquage des blocs occupés

Vérification du système de fichiers L'utilitaire traditionnellement appelée `fsck` (*File System CheckK*) sous UNIX effectue un parcours de l'arborescence, et de la liste des blocs libres, pour en vérifier la cohérence.

1. vérification des i-noeuds, des blocs et des tailles
2. vérification de la structure des répertoires
3. vérification de la connectivité des répertoires
4. vérification des compteurs de référence
5. vérification de l'information du sommaire de groupe

Certaines corrections sont effectuées automatiquement :
- Dans le cas où des répertoires contiennent des références à des i-nodes détruits, les références sont supprimées.
- Dans le cas inverse où des i-nodes "vivants" ne sont plus référencés, ils sont rattachés arbitrairement au répertoire "`lost+found`".

D'autres nécessitent l'intervention de l'administrateur, par exemple quand des fichiers font usage de blocs qui sont marqués comme libres. Il faut alors choisir entre supprimer le fichier, ou rattacher le bloc au fichier.

5.6 Autres caractéristiques des SGF

Les systèmes de fichiers modernes ont d'autres fonctionnalités importantes.

5.6.1 Journalisation

Journal :
- garde une trace des opérations d'écriture non terminées
- fournit un *point de reprise* au cas où l'opération serait interrompue (plantage, coupure de courant) pendant une écriture sur disque.

Avantages
- pas de pertes d'informations
- reprise plus rapide (évite le *fsck*)

5.6.2 Snapshots (instantané)

Les "instantanés" sont des copies (virtuelles) de l'état du système de fichiers à un moment donné.

Besoin Ceci est utile en particulier pour les sauvegardes. Quand on fait une sauvegarde avec `tar` par exemple, ce programme établit la liste des fichiers à sauvegarder avec leurs noms, leurs tailles etc, puis construit une archive contenant cette liste et le contenu des fichiers.

Or il se peut que les données à sauvegarder "bougent", parce que les utilisateurs (ou des programmes) ajoutent, suppriment ou modifient les fichiers entre la constitution de la liste et la construction de l'archive ; ce qui peut conduire à une archive inexploitable.

Fonctionnement Quand on demande un "snapshot", le SGF prend note de toutes les modifications apportées au système de fichier "actif" à partir d'un moment précis. Le snapshot est constitué d'une table des sauvegardes des blocs qui ont été modifiés sur le système de fichiers "actif". Lorsqu'on demande le bloc n du snapshot, le SGF renvoie la sauvegarde du bloc n si il y en a eu une, et sinon le bloc n du système de fichiers. La commande de sauvegarde peut donc, à travers le "cliché", accèder aux données dans l'état où elles étaient à un moment précis.

En quelque sorte c'est une duplication partielle du système de fichiers, au fur et à mesure des modifications ; le snapshop est effectué instantanément (au moment de sa création, la table des blocs modifiés est vide), et prend peu de place quand le système de fichiers est peu actif.