Chapitre III: GESTION DES FICHIERS

III. 1 Fichier

Un **fichier** (**File**) désigne un ensemble d'informations stockées sur un support de stockage permanant, mais indépendant du support de stockage, en vue de leur conservation et de leur utilisation dans un système informatique. Il représente, du point de vue utilisateur, la plus petite entité logique de stockage sur une mémoire de masse.
Chaque fichier est identifié par un nom auquel on associe un **emplacement** sur le disque (une référence) et possède un ensemble de propriétés : ses **attributs**. Appelés aussi les **métadonnées.**
Les Métadonnées les plus connues sont :

- Un propriétaire (**Owner**).
- Un groupe (**Group**).
- Une série de droits d'accès (**Access Permissions**) : lecture (**R, Read**), écriture (**W, Write**), exécution (**X, Execute**). Ces droits sont définis pour l'utilisateur, le groupe, et Le reste du monde (**Other**).
- Type de fichier : fichier régulier, répertoire, périphérique caractère, périphérique bloc, tube, etc.
- Dates de création/modification/dernier accès.
  Dans linux, L'option **–l** de la commande **ls** (**l**i**s**t) permet de lister les attributs des fichiers et répertoires. Ces méta-données sont traditionnellement stockées dans une structure de données appelée **i-nœud** (**inode ; i**ndex **node**). La taille d'un inode est de 16 à 1000 octets Chaque inode possède un **numéro** (**unique**). La commande **stat** permet d'afficher l'intégralité du contenu de l'inode et l'option **–i** de la commande **ls** permet d'afficher le numéro d'inode.

Exemple
[root@localhost ls –l

- rw- r-- r-- 1 root root 1255493 fév 19 2008 02whole.pdf

Type : **-** Fichier régulier, **d** Répertoire, **l** Lien symbolique, **b** Périphérique bloc ; **c** Périphérique caractère
s Socket, **p** Tube nommé

Droits d'accès propriétaire : rw-
Droits d'accès groupe : r--
Droits d'accès autres : r--
Nombre de liens physiques : 1
Propriétaire : root
Groupe propriétaire root
Taille en octets : 1255493
Date et heure de modification : fév 19 2008
Nom : 02whole.pdf

Notons que les motivations pour l'utilisation des fichiers Sont :

1. la nécessité de manipuler une très grande quantité d'informations
2. Les informations doivent être conservées après la fin d'un processus qui les utilise
3. Permettre à plusieurs processus de pouvoir accéder simultanément à une information

III. 2 Type de fichiers

On distingue différents types de fichiers :

1. **Fichiers ordinaires (Regular Files)**

Les fichiers ordinaires (appelés aussi fichiers **réguliers**) peuvent contenir du texte, du code exécutable ou tout type d'information (binaire, c'est à dire non texte). Unix n'impose Aucune structure interne au contenu des fichiers ; ils sont considérés comme une suite d'octets.

2. **Fichiers répertoires (Directory Files)**

Un répertoire (**directory)** peut être vu comme un fichier spécial contenant des liens vers d'autres répertoires ou fichiers. Cela permet de regrouper dans un même répertoire, les fichiers ayant des caractéristiques communes (même propriétaire, les programmes, . . .) et de hiérarchiser le système de fichier.

3. **Fichiers périphériques (fichiers spéciaux) (Special Files)**

Les fichiers spéciaux correspondent à des ressources ; ils sont associés à des dispositifs d'entrées/sorties physiques. Ils n'ont pas de taille. Ils sont traités par le système comme des fichiers ordinaires, mais les opérations de lecture/écriture sur ces fichiers activent les dispositifs physiques associés. On distingue :

- les **fichiers spéciaux mode caractère** (**Character Devices**) qui permettent de modéliser les périphériques d'E/S série tels que les terminaux, les imprimantes et les modems.
- les **fichiers spéciaux mode bloc** (**Block Devices**) qui permettent de modéliser les disques.
-

4. **Les Pipes**

Correspondent aux canaux de communications FIFO (Tube) permettant de faire communiquer des processus au sein d'une même machine selon un modèle producteur/ consommateur

5. **Les Sockets**

Ils permettent de faire communiquer des processus dans un réseau

6. **Les liens symboliques (Symbolic Links)**

III. 3 Les liens de fichiers

Chaque fichier (ordinaire ou répertoire) est référencé par son nom. Un lien est une entrée d'un répertoire qui référence un fichier qui se trouve dans un autre répertoire. Il existe deux sortes de liens : physiques et symboliques.

1 Un lien physique (Hard Link)

Un lien physique, appelé aussi lien dur, correspond à l'ajout d'un nouveau nom pour le même fichier. Les liens physiques ne sont autorisés que pour les fichiers de données (et donc interdits pour les répertoires ou les fichiers spéciaux) et ne peuvent exister que dans le même système de fichiers que le fichier lié.. Un lien physique n'est donc pas un fichier mais juste un autre nom au fichier. Le nombre de lien physique crée est mentionné dans un attribut appelé nombre de lien physique. La suppression d'un fichier n'a lieu que si ce nombre est égale à zéro (= 0)

2 Un lien symbolique (Soft Link)

Un lien symbolique est implémenté comme un fichier spécial contenant le chemin du fichier ou du répertoire lié. Il peut se trouver n'importe où dans la hiérarchie. Il est considéré comme un chemin raccourci au fichier source. La suppression du lien ne supprime pas le fichier et la suppression du fichier ne supprime pas le lien. Ce type de lien est lié au nom du fichier et nom au fichier lui-même. Le changement de nom ou la

suppression du fichier casse ce lien. Et la création d'un nouveau fichier avec le nom du lien précédant réalise un lien direct entre eux. De plus, le lien logique peut se trouver dans n'importe quel autre endroit. .

Soit F un fichier et FP et FS respectivement un lien physique et symbolique vers F
F et FP ont les mêmes caractéristiques et partage le même inode par contre FS n'a pas les mêmes caractéristique et sa taille est presque nulle

Dans linux , La création des liens physiques ou symboliques se fait à l'aide de la commande **ln**. La commande :
$ **ln** nomfich nomlien
crée un lien **physique** appelé **nomlien** sur le fichier **nomfich**. Par contre la commande :
$ **ln -s** nomfich nomlien
crée un lien **symbolique** appelé **nomlien** sur le fichier **nomfich**

III. 4 Système de gestion de fichier (SGF)

Le système de gestion fichiers est la partie du système d'exploitation qui se charge de l'organisation et gestion des fichiers. Il représente une interface entre l'utilisateur et les couches basses du système. Il assure également la correspondance entre les noms de fichiers visibles à l'utilisateur et l'emplacement physique de ces fichiers sue la mémoire de masse.

Les fonctions d'un SGF se résument à ce qui suit :

- Offrir aux utilisateurs la possibilité d'effectuer des opérations abstraites sur les objets du système de fichier telle que la création, la suppression, l'ouverture, la fermeture, la lecture et l'écriture (manipulation)
- Garder trace des informations sur tous les fichiers stockés par le SGF
- Définir une politique ou méthode d'accès aux fichiers (localisation)
- Définir une politique d'allocation de l'espace de mémoire secondaire utilisé
- Définir une politique de restitution, ou libération de l'espace disque occupé
- Réaliser une sécurité et contrôle sur les fichiers (% confidentialité, intégrité…)

III. 5 Manipulation des fichiers

La manipulation d'un fichier est réalisée à travers un ensemble d'opérations qui se traduisent par des appels système. Les opérations de base sont :

- **Création de fichier :** le système alloue un espace destiné au nouveau fichier puis crée une nouvelle entrée dans le répertoire destination. Cette entrée contient le nom du fichier et son emplacement

- **Ouverture et fermeture :** l'ouverture d'un fichier consiste à informer le système que le fichier est actif, ceci permet d'éviter de refaire, à chaque demande d'accès , la recherche du fichier dans le système du fichier/ Pour cela, le système utilise une table des fichiers ouverts où chaque entré correspond à un fichier ouvert indiquant les informations du fichier (emplacement et autres). L'accès au fichier se fait se fait alors à travers son point d'entrée dans la table.
  La fermeture du fichier permet donc de supprimer son entrée dans la table
  Un descripteur de fichier représente un pointeur vers la **table des fichiers ouverts** (**Open Files Table**) dans le système.
- **Lecture et écriture :** Le système recherche l'entré correspondante au fichier et utilise un pointeur de position indiquant l'emplacement dans le fichier
- **Positionnement dans un fichier :** Dans ce cas, le pointeur de positionnement est initialisée à une valeur donnée

- **Suppression :** Le système recherche l'entrée correspondante dans le répertoire puis est supprimée une fois trouvée

D'autres opérations sont possibles telles que la troncature, renommage ou copie de fichiers,…etc.

III. 6 Les structures des fichiers

Les fichiers peuvent être structurés selon trois manières différentes :

- Un fichier est une séquence d'octets non structurés comme le cas de Windows et Unix
- Un fichier est une séquence d'enregistrements de taille fixe
- Un fichier est un arbre d'enregistrements qui ne sont pas nécessairement de même taille

III. 7 Méthodes d'accès à un fichier

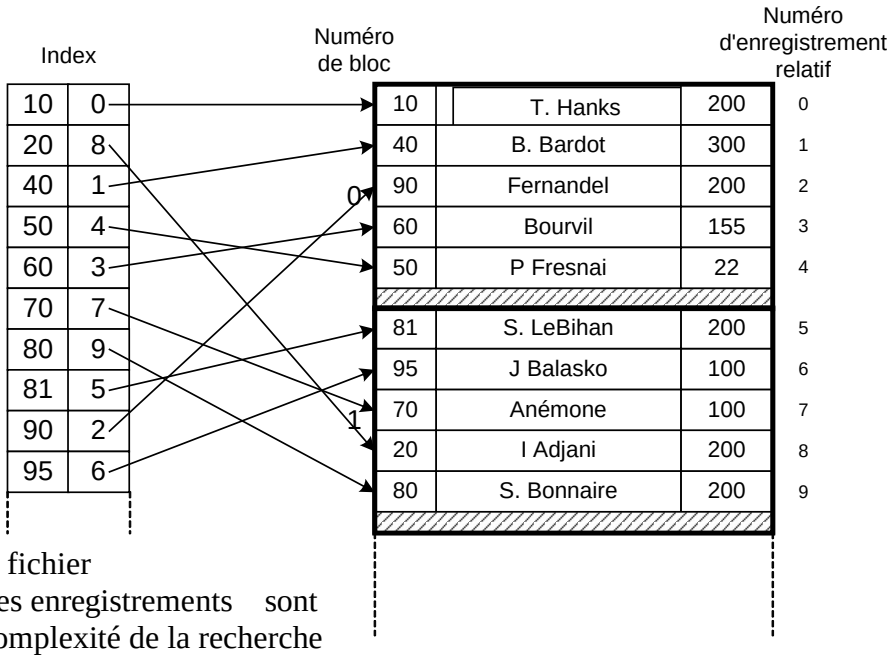
L'accès à l'information dans un fichier se fait selon une méthode défini, parmi ces méthodes nous citons
:

a. Accès séquentiel : Dans ce mode, le traitement des informations d'un fichier se fait dans l'ordre de leur stockage, d'une manière séquentielle (comme le cas d'une bande ou cassette magnétique). Dans ce code, L'opération de rajout d'information très simple et consiste à transférer le dernier bloc dans le buffer. Dans l'emplacement indiqué par le pointeur de position. En revanche les recherches sont coûteuses. La suppression nécessite des indicateurs de suppression et provoque des trous ce qui engendre des réorganisations. La complexité des algorithmes de recherche est de O(N)

On peut avoir une variante L'organisation séquentielle triée
Elle repose sur une notion de champ clé. Elle permet d'accélérer les recherches.
L'insertion est très coûteuse : on peut laisser de la place libre dans chaque bloc ou réserver une zone non triée pour les insertions.

b. Accès direct : l'accès à l'information se fait directement à n'importe quel bloc du fichier. Dans ce cas le fichier doit être organisé en enregistrements logiques de taille fixe. L'utilisateur fourni au système d'exploitation un numéro de bloc relatif (index relatif) (0, 1, 2…)

c. Accès indexé : Ce mode se base sur l'utilisation d'un index de fichier qui contient des pointeurs vers les blocs du fichier l'accès consiste à rechercher d'abord séquentiellement dans l'index pour trouver le pointeur associé au bloc demandé, puis accéder directement au bloc en utilisant son pointeur. Cette méthode est appelée séquentielle indexée.



Un fichier séquentiel indexé est un fichier à accès séquentiel où l'ensemble des enregistrements sont ordonnés selon une clé de tri. La complexité de la recherche est de ***O***(log(N))

Un fichier est découpé en paquets de faible taille qui peuvent être composés d'une ou plusieurs pages. Les enregistrements sont répartis dans ces paquets en utilisant une fonction de hachage : Fonction H : Clé ----> Paquet.

L'indexation a pour but d'accélérer les traitements en ajoutant des structures de servitude. Il existe différentes techniques d'indexation selon le contexte. L'évaluation d'une technique se fait selon des critères de temps d'accès, délai d'insertion, délai d'effacement, occupation mémoire. Pour une même relation, il est possible d'avoir plusieurs index.

Remarque : On peut avoir une variante de l'accès indexé, il s'agit de l'a**ccès** indexé basé sur le hachage **(accès haché)** : EN effet, pour éviter de parcourir un index pour accéder à une donnée, une méthode de hachage est préconisée. Etant donnée une valeur clé d'un enregistrement, l'application d'une fonction de hachage doit permettre de trouver la position de cet enregistrement dans la structure contenant tous les enregistrements (tableau, fichier).

Lors d'une insertion, l'application de la fonction de hachage permet de déterminer la place où l'enregistrement doit être inséré. Il peut se poser des problèmes de collision, différentes techniques sont alors utilisées pour choisir la bonne fonction de hachage. La complexité de la recherche , dans ce cas, est de ~O(1)

III. 8 Concept de répertoire

Les ordinateurs actuels permettent des support de stockage à capacité considérable (de méga octet =2 20 à giga =2 30 à Téra =2 40 octets). Le nombre de fichiers peut être ainsi augmenté et la localisation d'un fichier devient complexe. Ainsi le concept de répertoire a été introduit afin de hiérarchiser l'ensemble des fichiers sous forme d'arbre de dossiers

III. 8.1 Répertoire

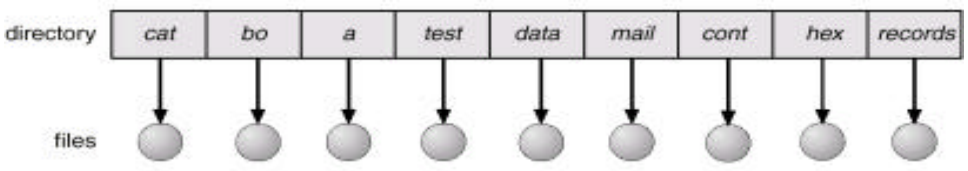
Un répertoire est un objet du système de fichier, il peut être vu comme un tableau contenant les informations d'autres objets de ce système telles que le nom, emplacement (adr du bloc) et l'ensemble des attributs.

III. 8.2 Structures de répertoire

Un répertoire peut avoir plusieurs structures

1. **Répertoire à un seul niveau :**

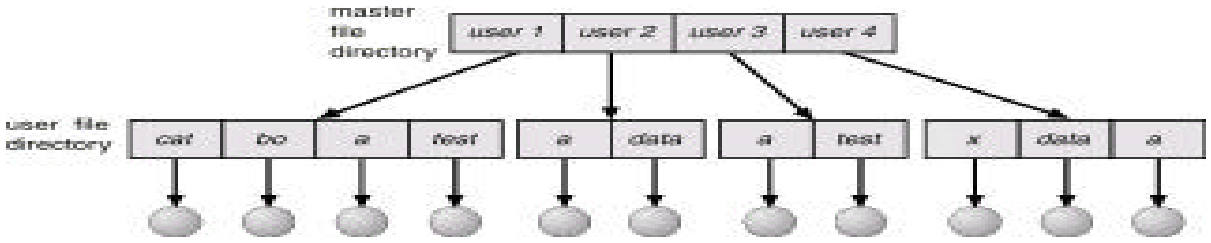
C'est la structure la plus simple, elle est composée d'un seul répertoire contenant uniquement des feuilles fichiers (pas de sous répertoires)



Inconvénient : difficile à gérer quand le nombre de fichier est grand car le nom doit être unique.

2. Répertoire à deux niveaux :

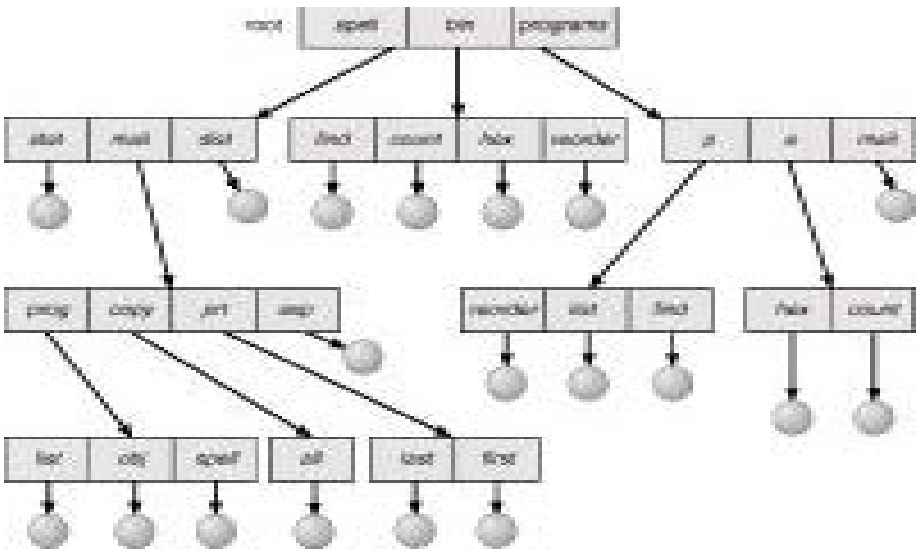
Dans cette structure, un répertoire distinct est crée pour chaque utilisateur. Le système dispose d'un répertoire de fichiers maitre MFD (Master File Directory) qui contient les répertoires de fichiers utilisateurs UFD (User File Directory). Ainsi des utilisateurs différents peuvent avoir des fichiers de mêmes noms. La localisation se fait par un nom complet unique à partir de la racine
Les répertoires utilisateurs sont créés et supprimés uniquement par l'administrateur (super utilisateur)



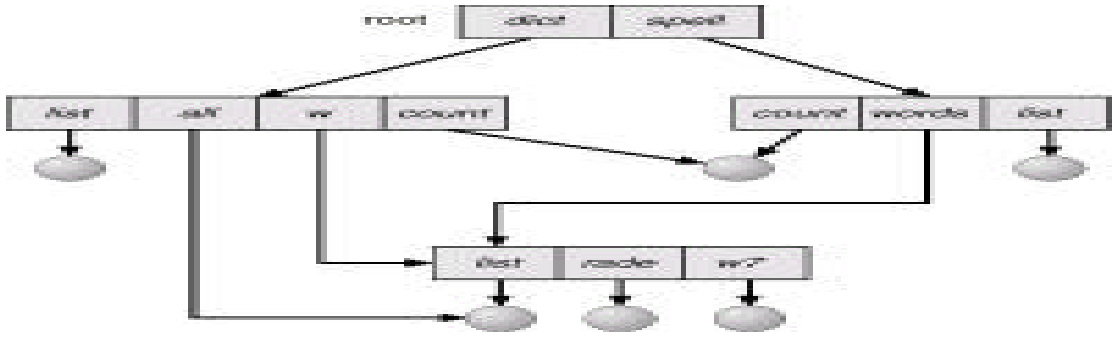
Inconvénient : Ce n'est pas possible de partager les fichiers entre utilisateurs à cause de l'isolation dans l'UFD

3. Répertoire structuré en arbre :

C'est une extension à la structure à deux niveaux. Qui permet à plusieurs utilisateurs de se partager un répertoire. La localisation peut se faire de deux manières soit par un chemin absolu à partir de la racine ou par un chemin relatif (c'est la notion de path).



4. Répertoire à graphe acyclique :

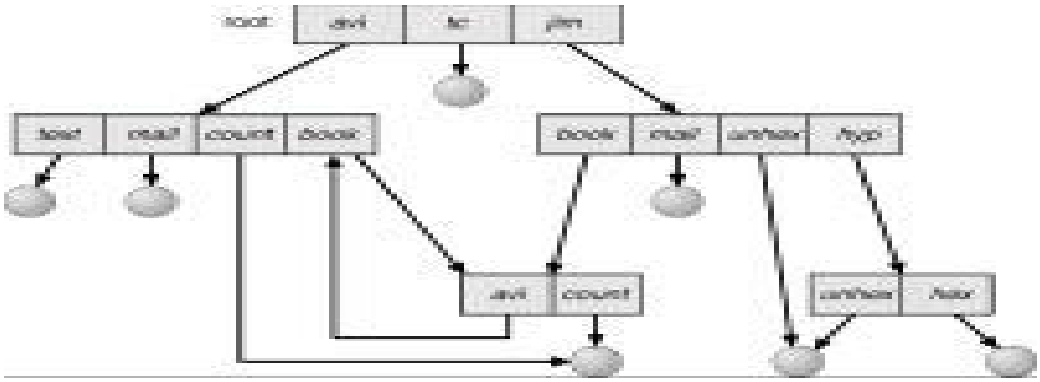


Cette structure est une généralisation à la structure d'arbre mais elle permet le partage de répertoires grâce aux liens qui peuvent lier deux répertoires différents. C'est une structure complète mais complexe puisqu'elle doit maintenir l'absence de cycles

Le partage dans le système linux se fait en utilisant les liens

5. Répertoire de graphes généraux :

Cette structure est une généralisation à la précédente, toutefois, elle rend difficile la gestion de certains problèmes tels que la recherche des références ou liens vers un fichier ou répertoire



III. 8.3 Opération sur les répertoires

Un ensemble d'opérations sur les répertoires est offert par le système :

- Créer un répertoire
- Supprimer un répertoire
- Ouvrir un répertoire, en lire un contenu
- Fermer un répertoire
- Renommer un répertoire
- …etc

III. 8.4 Implémentation des répertoires

La gestion des répertoires et l'accès à eux influent directement sur les performances du SGF. Ainsi deux implémentations ont été proposées pour représenter les répertoires :

- **Liste linéaire :** Cette méthode utilise une liste linéaire contenant les noms de fichiers ainsi que les pointeurs sur les blocs de données.

Avantage : simple à programmer

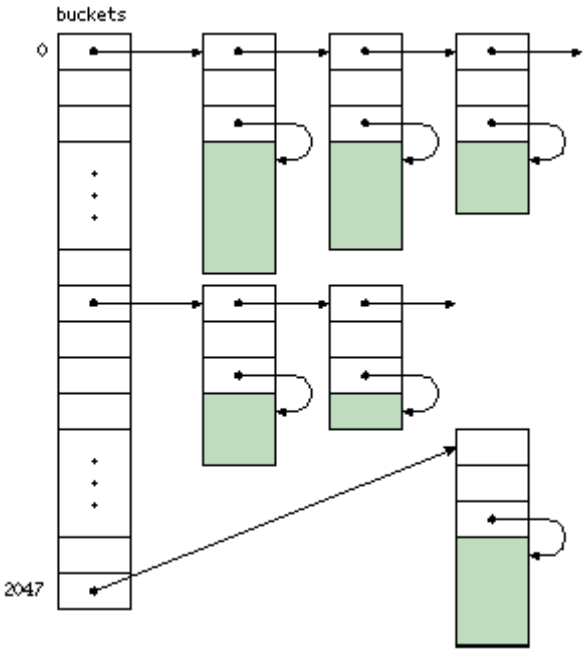
Inconvénient : méthode couteuse en temps d'exécution vu que l'ensemble des opérations nécessite un parcours linéaire de la liste

- **Table de hachage :** dans ce cas, une liste linéaire est utilisée pour contenir les entrées du répertoire, en plus, une table de hachage est utilisée, elle permet de réaliser la correspondance entre le nom du fichier et la position du fichier dans la liste linéaire.

Avantage : Réduction du temps de recherche dans le répertoire

Inconvénients : - Dépendance de la fonction de hachage de la taille de la liste

- Des noms de fichiers différents peuvent correspondre à la même clé, dans ce cas on utilise une liste linéaire pour chaque valeur distincte de la clé (chained overflow hash table)



III. 9 Partition et volume

L'organisation des fichiers sur disque nécessite un découpage logique, ainsi le système de fichier est décomposé en partitions ou volumes (appelés disques virtuels)

- Un volume est une structure logique de stockage, pouvant regrouper plusieurs disques physiques afin de donner l'impression d'un seul disque de stockage. Ceci présente l'avantage d'agrandir les capacités de stockage (cas fichiers volumineux ne pouvant pas résider sur un support)
- Une partition est une structure de bas niveau, permettant de stocker des fichiers de répertoires et donnant la possibilité à l'utilisateur de stocker ses fichiers dans des zones indépendantes comme si c'était des périphériques de stockage indépendants. Un disque doit contenir au moins une partition ou volume. Une partition ne peut contenir qu'un système de fichier. On distingue deux types de partitions :

    1. Une **partition primaire** (**Primary Partition**) est une partition physique qui est prête à recevoir un système de fichier quand elle est créée. Un disque dur est limité à **quatre** (04) partitions primaires.
    2. **La partition étendue** a été mise au point pour outrepasser la limite des quatre partitions principales, en ayant la possibilité de créer autant de lecteurs logiques que vous désirez dans celle-ci. Au moins un lecteur logique est nécessaire dans une partition étendue, car on ne peut pas y stocker de données directement. Une partition étendue (Extended Partition) ne peut pas être formatée, par conséquent elle ne peut pas supportée un système de fichier. Une partition étendue peut contenir des partitions logiques (Logical Partitions) qui sont des divisions logiques des secteurs de stockage qui peuvent être formatées afin de supporter un système de fichier.

***Remarque 1** : Beaucoup de machines sont formatées en une grande partition utilisant l'intégralité de l'espace disponible du lecteur. Ce n'est pourtant pas la solution la plus avantageuse en termes de performances et de capacité. La solution est de créer plusieurs partitions, ce qui va vous permettre:*

- *d'installer plusieurs systèmes d'exploitation sur votre disque*
- *d'augmenter la sécurité de vos fichiers*
- *d'organiser vos données plus facilement*

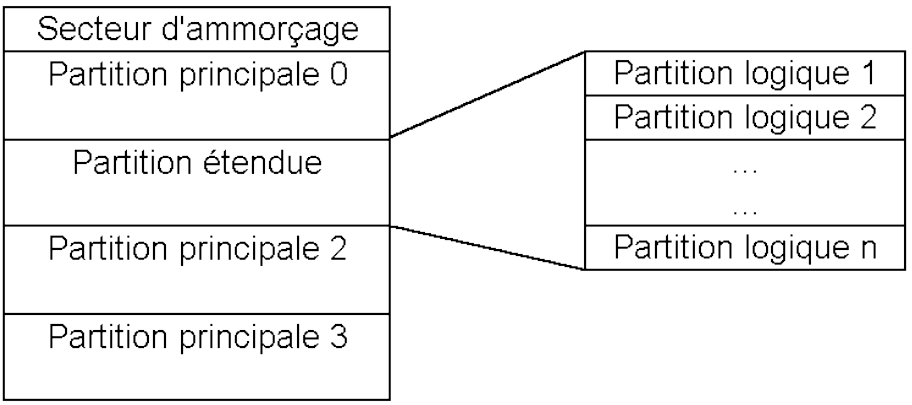
***Remarque 2** : Le partitionnement d'un disque dur se fait après le formatage physique de celui-ci et avant le formatage logique. Il consiste à créer des zones sur le disque dont les données ne seront pas mélangées. Cela sert par exemple à installer des systèmes d'exploitation différents n'utilisant pas le même système de fichiers. Il y aura donc au minimum autant de partitions que de systèmes d'exploitation utilisant des systèmes de fichiers différents. Dans le cas d'un utilisateur d'un système d'exploitation unique, une seule partition de la taille du disque peut suffire, sauf si l'utilisateur désire en créer plusieurs pour faire par exemple plusieurs lecteurs dont les données sont séparées.*

Par ailleurs, les informations sur les partitions sont conservées sur le disque lui-même dans des zones qu'on appelle tables de partitions. La table de partitions est contenue dans le premier secteur du disque ou secteur d'amorçage (Master boot record ou MBR) qui contient également le programme d'amorçage. Chaque ligne d'une table de partitions contient l'adresse de début de la partition et sa taille. Il peut s'agir de partitions primaires qui contiendront un système de fichiers ou de partitions étendues qui contiendront à leur tour une table de partitions ayant la même structure que la table principale.

Le secteur de démarrage (appelé Master Boot Record ou MBR en anglais) est le premier secteur d'un disque dur (cylindre 0, tête 0 et secteur 1), il contient la table de partition principale et le code, appelé boot loader, qui, une fois chargé en mémoire, va permettre d'amorcer (booter) le système.

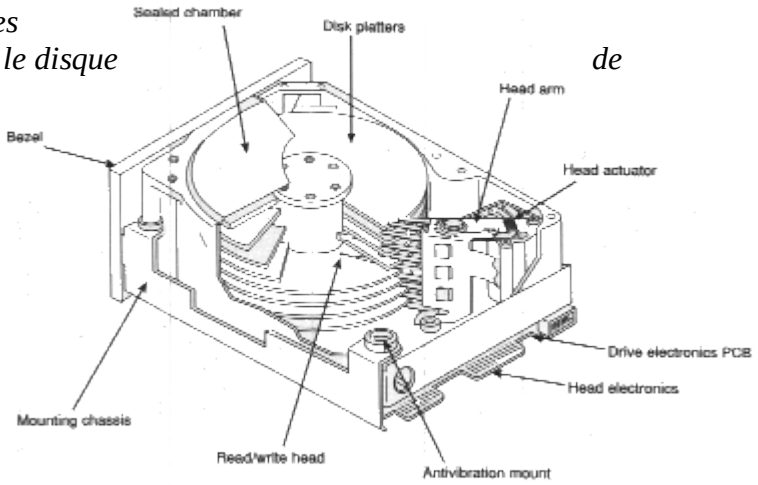
Ce programme, une fois en mémoire, va déterminer sur quelle partition le système va s'amorcer, et il va démarrer le programme (appelé bootstrap) qui va amorcer le système d'exploitation présent sur cette partition.
D'autre part, c'est ce secteur du disque qui contient toutes les informations relatives au disque dur (fabricant, numéro de série, nombre d'octets par secteur, nombre de secteurs par cluster, nombre de secteurs,...). Ce secteur est donc le secteur le plus important du disque dur, il sert au setup du BIOS à reconnaître le disque dur. Ainsi, sans celui-ci votre disque dur est inutilisable, c'est donc une cible de prédilection pour les virus.

Remarque 3 : On peut avoir un troisième type c'est la partition logique, une partition logique ne peut pas avoir d'existence en dehors d'une partition étendue. Une partition logique peut contenir un système d'exploitation (si celui ci peut être lancé à partir d'une partition logique) ou des données.



III. 9 Le Formatage d'un disque

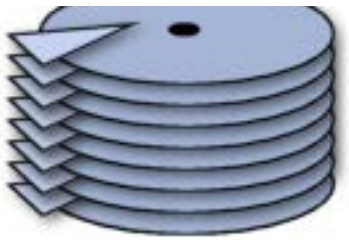
Le formatage sert à organiser l'agencement des données qui vont être stockées. Le formatage prépare le disque de manière à ce que les données puissent être écrites et récupérées rapidement.



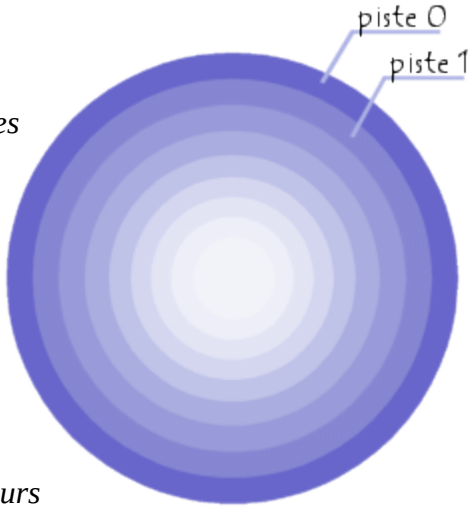
On distingue deux types de formatage. Le formatage physique (ou formatage de bas niveau) et le formatage logique (de haut niveau).

1. Le Formatage Physique

Les disques durs, aussi petits soient-ils, contiennent des millions de bits, il faut donc organiser les données afin de pouvoir localiser les informations, c'est le but du formatage. La surface de chaque cylindre, originalement uniforme est divisée (magnétisées) lors du formatage en petites parcelles qui pourront plus facilement être repérées.



Un disque dur est composé de plateaux montés sur un axe. Les plateaux tournent sur cet axe. Des têtes de lecture / écriture (au moins une par face de chaque plateau) montées sur des bras mobiles et qui se déplacent de manières transversales pour parcourir la surface des plateaux.
Les plateaux sont souvent en fer, recouvert d'une couche d'oxyde de fer (qui possède des propriétés magnétiques).



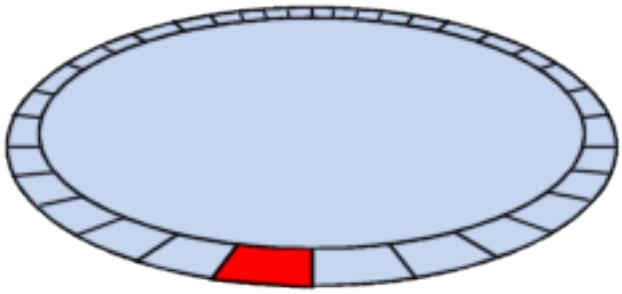
Un disque dur est, rappelons-le, constitué de plusieurs plateaux circulaires tournant autour d'un axe et recouverts de part et d'autre par un oxyde magnétique, qui, en étant polarisé, va pouvoir stocker des données.

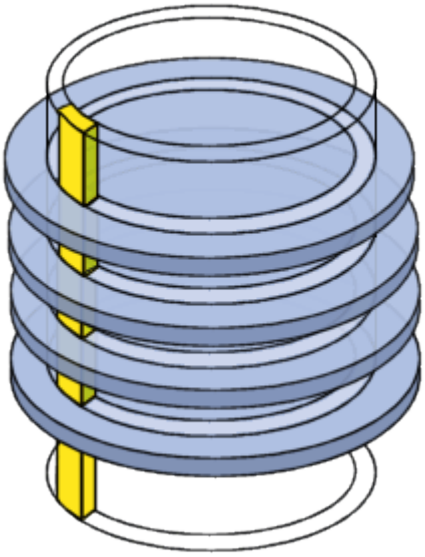
Les pistes sont des zones concentriques écrites de part et d'autre d'un plateau.

Enfin, ces pistes sont découpées en quartiers appelés secteurs. un secteur a une capacité de 512 octets (d'une manière générale)

Les pistes se comptent par milliers et comptent chacune de 60 à 120 secteurs environ.

On appelle cylindre l'ensemble Un cylindre est formé par l'ensemble des pistes de chacun des plateaux et qui se situent à la même distance du centre de ces plateaux..



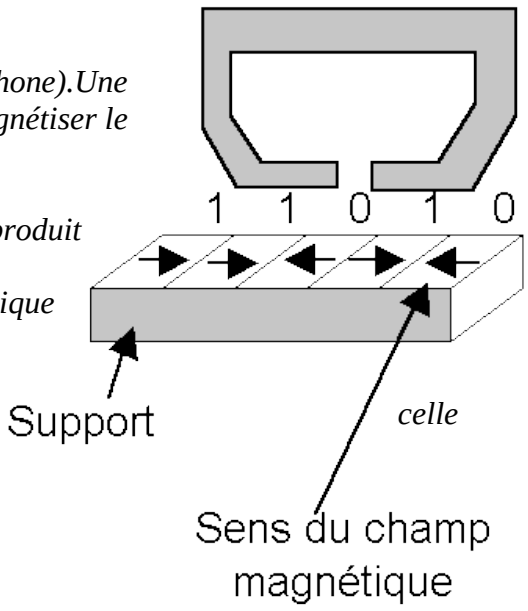


Tête de lecture / écriture

L'enregistrement et magnétique (analogue à la bande d'un magnétophone).Une tête de lecture / écriture est composée d'un aimant qui va pouvoir magnétiser le support sur lequel elle passe.

Pour l'écriture un faible courant passe dans la tête de lecture ce qui produit un champ magnétique au-dessus d'une cellule d'enregistrement du plateau. Le sens du courant détermine l'orientation du champ magnétique (positif ou négatif) qui sera interprété comme un 0 ou un 1.

Pour la lecture, lorsque la tête de lecture passe sur une cellule, ci, se comportant comme un aimant, va induire un courant dans une



bobine suitée sur la tête de lecture. Le sens de ce courant sera interprété comme un 0 ou un 1.

Les clusters

Un cluster est formé par le regroupement de plusieurs secteurs (un secteur = 512 octets). La taille d'un cluster dépend de la taille de la partition et du système de fichiers utilisé.

Il est important de comprendre l'influence de la taille des clusters. Un fichier (quel qu'il soit) occupe un nombre de cluster entier. Cela pose des problèmes si vous avez beaucoup de fichier de petite taille.

Capacité Disque = nb Surfaces × nb Cylindres × nbSecteursParPiste × taille Secteur

Le formatage physique *consiste à ainsi organiser la surface de chaque plateau en entités appelées pistes et secteurs, en polarisant grâce aux têtes d'écriture des zones du disque.*

Les pistes sont numérotées en partant de 0, puis les têtes polarisent concentriquement la surface des plateaux. Lorsque l'on passe à la piste suivante, la tête laisse un "trou" (appelé gap en anglais) et ainsi de suite. Chaque piste est elle-même organisée en secteurs (numérotés en commençant à partir de 1) séparé entre eux par des gaps. Chacun de ces secteurs commence par une zone réservée aux informations du système appelée préfixe et se termine par une zone appelée suffixe

Le formatage physique est la première étape. Ce formatage de bas niveau est effectué par le fabricant. Certains utilitaires du commerce ainsi que certains Bios permettent d'effectuer ce formatage.

Le but du formatage de bas niveau est de diviser la surface des disques en éléments basiques :

Pistes, secteurs, cylindres

Le formatage de bas niveau a donc pour but de préparer la surface du disque à accueillir des données (il ne dépend donc pas du système d'exploitation et permet grâce à des tests effectués par le constructeur de "marquer les secteurs défectueux.

Remarque1 : *Lorsque vous achetez un disque dur, celui-ci a déjà subi un formatage de bas niveau, IL N'EST DONC PAS NECESSAIRE D'EFFECTUER UN FORMATAGE DE BAS NIVEAU!*

Remarque2 : *Pendant le formatage ; des tests de contrôle (algorithme permettant de tester la validité des secteurs grâce à des sommes de contrôle) sont effectués eà chaque fois qu'un secteur est considéré comme défectueux, la somme de contrôle (invalide) est inscrite dans le préfixe, il ne pourra alors plus être utilisé par la suite, on dit qu'il est "marqué défectueux".*

Lorsque le disque lit des données, il envoie une valeur qui dépend du contenu du paquet envoyé, et qui est initialement stockée avec ceux-ci. Le système calcule cette valeur en fonction des données reçues, puis la compare avec celle qui était stockée avec les données. Si ces deux valeurs sont différentes, les données ne sont pas valides, il y a probablement un problème de surface du disque.
Le contrôle de redondance cyclique (CRC, en anglais cyclic redundancy check), est basé sur le même principe pour contrôler l'intégrité d'un fichier.

Les utilitaires d'analyse tels que scandisk ou chkdsk opèrent autrement :
ils inscrivent des données sur les secteurs à priori marqués valides, puis les relisent et les comparent. Si

ceux-ci sont similaires, l'utilitaire passe au secteur suivant, dans le cas contraire ils marquent le secteur défectueux.

2. Le Formatage logique

Le formatage logique s'effectue après le formatage de bas niveau, il crée un système de fichiers sur le disque (ou partition), qui va permettre à un système d'exploitation (DOS, Windows ; Linux, OS/2, ...) d'utiliser l'espace disque pour stocker et utiliser des fichiers.

Pour installer des systèmes d'exploitation de natures diverses il est nécessaire de créer des partitions qui auront des SGF différents. Le formatage logique consiste à placer des informations complémentaires, selon le système de gestion de fichiers employé, dans les secteurs préalablement définis lors du formatage de bas niveau. Les informations enregistrées lors d'un formatage logique sont :

- écriture du secteur d'amorçage des partitions,
- enregistrement de l'octet d'Identification système (ID System) dans la table des partitions du Disque dur,
- informations du système de fichiers sur l'espace disponible, l'emplacement des fichiers et des répertoires...
- repérage des zones endommagées...

Remarque : *On peut avoir deux options dans le formatage logique : le formatage complet et le formatage rapide chacun a ses propres caractéristiques*

III. 9 Implantation des fichiers

Le paramètre le plus important dans l'implantation des fichiers est sans doute la mémorisation des adresses de tous les blocs assignés à chaque fichier. Différentes méthodes sont utilisées suivant le système d'exploitation, nous présenterons quelques-unes :

- **ALLOCATION CONTIGUE**

C'est la méthode d'allocation la plus simple, elle consiste à stocker chaque fichier dans une suite de blocs consécutifs, ainsi seules deux informations sont nécessaire : l'adresse du premier bloc et du fichier et sa longueur, ces informations sont sauvegardés dans l'entrée du répertoire.

Avantages :

- facile à implémenter

- facilité d'accès à un fichier que ce soit un accès séquentiel ou direct

- Le temps de positionnement et le nombre de positionnement sont minimisés

Inconvénients :

- difficulté de trouver l'espace contigu nécessaire à un nouveau fichier

- une insertion ou suppression d'un enregistrement nécessite un décalage des enregistrements stockés, ce qui est couteux.

- L'allocation contigüe requiert la connaissance de la taille d'un fichier et une pré-allocation de l'espace nécessaire au fichier

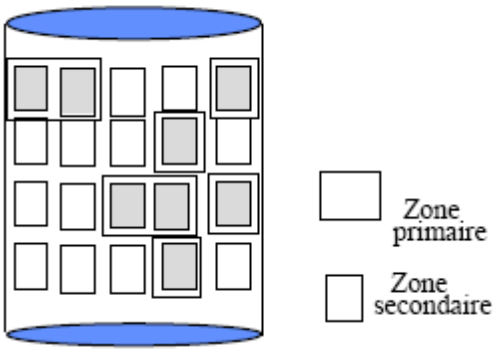
- Possibilité de fragmentation externe, en effet, la libération et utilisation des espaces disques subdivise le disque en parties occupées et libres. On peut se trouver donc devant la situation où une requête ne peut être satisfaite car la taille demandée est plus grande que le plus gros morceau libre. Un programme de compactage ou de ramasse miette est nécessaire

1111111		2222222	33333333		555555555	

Remarque : *Ce type d'allocation est utilisé pour les CDROM et DVD*

<u>Une variante de l'allocation contigüe</u>

Pour éviter le problème de fragmentation, une allocation par zones a été proposée de telle sorte qu'un fichier soit constitué de plusieurs zones physiques disjointes (exemple système MVS IBM) :
– une zone primaire allouée à la création et
– des zones secondaires (extensions) allouées au fur et à mesure.
Chaque zone est allouée de façon indépendante.



Problèmes précédents atténués mais toujours existants

- **ALLOCATION CHAINEE**

  Cette méthode consiste à chainer les blocs alloués à un fichier sans qu'ils soient contigus. L'entrée du répertoire correspondant contient un pointeur vers le premier bloc. La création d'un fichier permet de créer une entrée dans le répertoire avec un pointeur initialisé à NULL (fichier vide)

  La suppression ou insertion d'enregistrements nécessite la réorganisation des blocs de fichiers

<u>Avantages :</u>

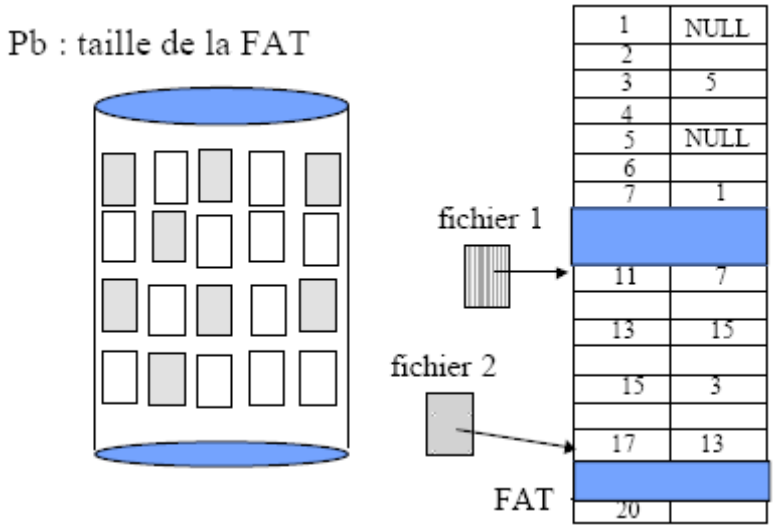
- Pas de fragmentation externe et aucune opération de compactage n'est nécessaire, les fichiers peuvent grandir tant qu'il ya de l'espace libre
- La taille de fichier ne doit pas être connue à l'avance

<u>Inconvénients :</u>

- Chaque bloc du disque nécessite un pointeur pour le chainage d'où un espace disque important sera consommé par les pointeurs, ce qui engendre une perte d'espace
- L'allocation chainée ne peut être utilisée qu'avec des fichiers à accès séquentiel vu qu'on doit parcourir toute la liste pour arriver au dernier. En plus chaque accès à un pointeur requière une lecture de disque et dans certain cas un positionnement de la tète de lecture est nécessaire

Une variante de l'allocation chainée

Une section de disque contient pour chaque partition une table appelée FAT (File Allocation table) qui contient une entré pour chaque bloc d'un fichier (cette technique est utilisée dans MSDOS, WINDOWS et OS/2. cette table est située au début de chaque partition disque. A chaque fichier lui correspond une entrée dans la table, qui fournit le numéro du bloc suivant, qui pointe à son tour au bloc suivant et ainsi de suite jusqu'à atteindre la fin de fichier. Si le bloc est le dernier du fichier ; son entrée dans la Fat contiendra une valeur spéciale de fin de fichier.



Avantage :

Gain de temps d'accès

Inconvénients :

La table FAT non seulement est dépendante de la taille du disque(une entrée pour chaque bloc) doit être entièrement en mémoire car elle est tout le temps sollicitée. Si on a un disque de 500 000 blocs de 1ko, On aura besoin de 500 000 entrées

- **ALLOCATION INDEXEE**

Le principe de l'allocation indexée est de prévoir pour chaque fichier une table appelée index. Celle- ci regroupe l'ensemble des pointeurs des blocs constituant le fichier. Chaque entrée de cette table pointe vers le $i^{ème}$ bloc du fichier Les adresses des blocs physiques constituant un fichier sont rangées dans une table appelée index, elle-même contenue dans un ou plusieurs blocs disque

. Avantage

- .Libère intégralement l'espace du bloc pour les données.
- . Facilite les accès aléatoires (direct).

- . L'entrée du répertoire contient un seul entier (le numéro du premier bloc) qui permet de trouver les autres blocs.
- Pas de fragmentation externe

.Inconvénient

- Difficulté de déterminer le nombre d'éléments que doit contenir le bloc index
- Gaspillage d'espace disque ; En effet, dans le cas d'un fichier constitué de peu de blocs de données, son bloc index comportera un petit nombre de pointeurs alors que tout un bloc physique lui sera alloué

Une Variante de l'allocation indexée

Cette variante a été utilisée dans Unix. Il s'agit d'utiliser une structure de donnée appelé nœuds d'information (inode ou index node) laquelle inclut les attributs et les adresses disque des blocs fichier. En fonction de l'inode on peut trouver tous les blocs du fichier. L'inode en plus de la partie attributs possede 13 pointeurs ; les 10 premiers sur les blocs de données et le 11 pointe sur un bloc index dont les pointeurs pointent sur des données (Simple indirect).

Le 12 ème pointeur pointe sur un bloc index dont les pointeurs pointent sur d'autres blocs index dont les pointeurs pointent sur des données (Double Indirect), Le 13 ème donne une triple indirection.

L'accès à l'inode se fait à travers le répertoire qui est compose, pour chaque fichier, du nom du fichier et du numéro d'inode.

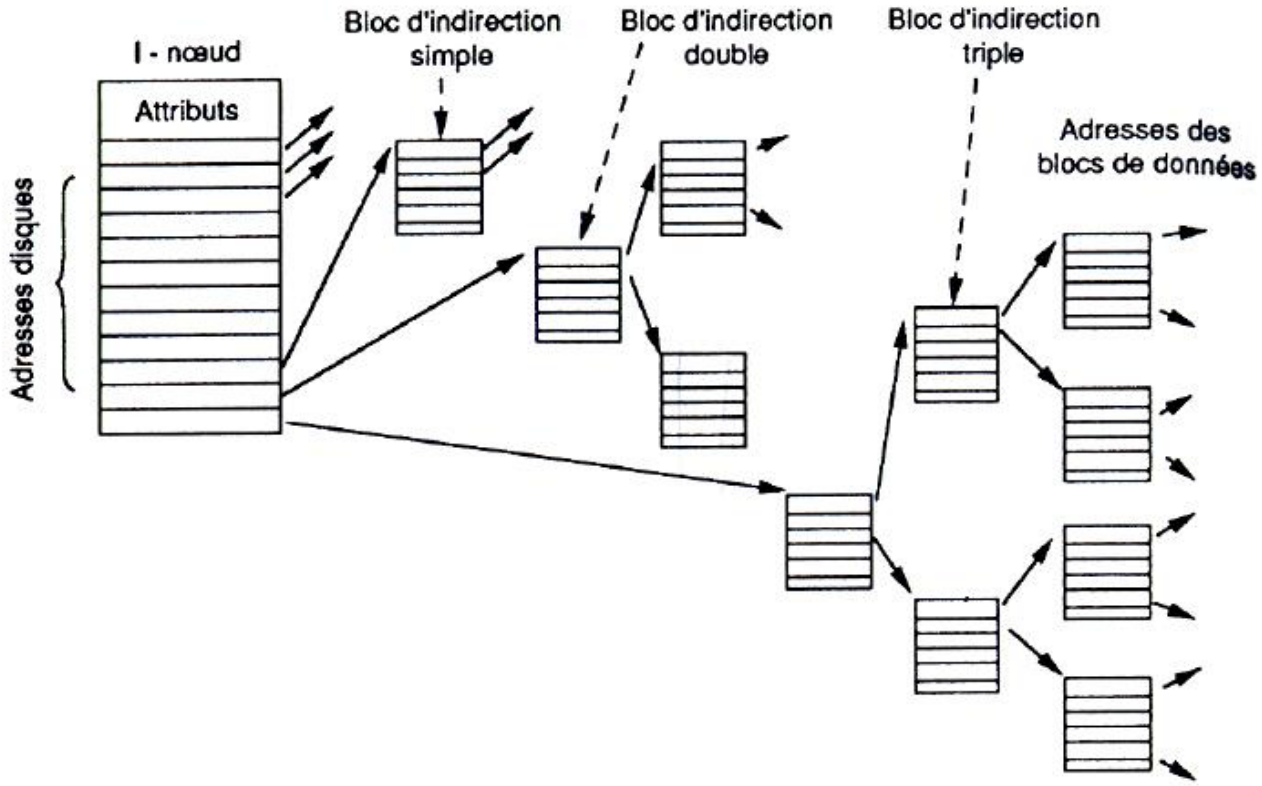


Figure 4.12 Structure d'un nœud d'information (i-nœud).

II.10 Gestion de l'espace libre

Le SGF doit connaitre l'espace libre sur disque afin d'allouer de l'espace aux fichiers. Il doit aussi pouvoir réutiliser l'espace des fichiers supprimés pour de nouveaux fichiers. Ainsi il maintient une liste contenant les blocs libres La mise à jour de cette liste à lieu soit à la création du fichier et l'augmentation de sa taille en rajoutant de nouveaux enregistrements soit à la suppression d'un fichier ou d'un ensemble de ses enregistrements. .

COMMMENT IMPLEMENTER CETTE LISTE ?

Il existe plusieurs manières d'implantation d'une telle liste :

1. Vecteur binaire

Le vecteur binaire est tableau de bits dit Bitmap où chaque bit désigne l'état d'un bloc alloué (1) ou libre (0)

Avantage

Facilité de recherche d'un ensemble de blocs libres contigus

Inconvénient

Difficulté de maintenir le vecteur dans sa totalité en mémoire dans le ca d'un disque à grande capacité

2. Liste chainée

Dans ce cas, Les blocs libres sont chainés entre eux ; Le SGF sauvegarde le pointeur du premier bloc libre

Inconvénient

Nécessité de lire chaque bloc pour parcourir la liste des blocs libres mais souvent on requiert un bloc à la fois.

3. Groupage

Dans cette méthode, on sauvegarde dans un bloc libre n-1 adresses de blocs libres et la $n^{ème}$ adresse contient les adresses des n prochains blocs libres, ainsi de suite.

Avantage

Rapidité de recherche d'un grand nombre de blocs libre

4. Compactage

Cette méthode consiste à éviter de réserver un espace de pointeur pour chaque bloc mais à la place, elle sauvegarde pour chaque zone libre, une adresse de bloc libre et le nombre de blocs libres qui le suivent

III.11 Le système de fichiers.

Un système de fichiers permet d'organiser le suivi de l'espace alloué et de l'espace libre, de gérer les noms et l'emplacement physique des fichiers ainsi que des dossiers.

Il existe plusieurs systèmes de fichiers les principaux étant :

• *FAT (File Allocation Table)*
• *FAT32 (File Allocation Table 32)*
• *NTFS (New Technology File System*
• *Linux Ext2, Ext3*

A/ La Table d'Allocation de Fichiers est un tableau de valeurs numériques dont chaque case permet de décrire l'allocation des clusters d'une partition, c'est-à-dire l'état (l'occupation ou non par un fichier) de chaque cluster de la partition dont elle fait partie.

- **FAT16**

C'est un système 16 bits, cela signifie qu'il ne peut pas adresser les clusters sur plus de 16 bits. Le nombre maximum de clusters repérables avec le système FAT est ainsi de 2^{16}, soit 65535 clusters. Or, étant donné qu'un cluster est constitué d'un nombre fixé (4,8,16,32,64) de secteurs de 512 octets contigus, la taille maximale d'une partition FAT se trouve en multipliant le nombre maximum de clusters par la taille d'un cluster.

Avec des clusters d'une taille 32Ko (64*512), la taille maximale d'une partition FAT est donc de 2Go (32 767 * 65535= 2 147 385 345). Ainsi, quelle que soit la taille de la partition ou du disque les clusters doivent être suffisamment grands pour que toute l'espace disque puisse être contenue dans ces 65525 clusters.

Calculer la taille du cluster si le disque est de 1,2 Go (1 207 959 552 octets) en FAT 16 On a 1207 959 552 / 65535 = 18433 octets / 512 = 36 secteurs

Ainsi, plus la taille du disque (ou de la partition) est importante, plus la taille des clusters doit être importante. Ceci conduit à une perte d'espace disque

Le système FAT16 permet de décrire un fichier par un nom d'une longueur de 8 caractères et une extension de 3 caractères.

- **FAT32**

Ce système de fichiers, appelé FAT32 utilise des valeurs 32 bits pour les entrées de la FAT. Avec l'apparition du système de fichiers FAT32, le nombre maximal de clusters par partition est passé de 65535 à 268 435 456. La FAT32 autorise donc des partitions d'une taille beaucoup plus élevée (jusqu'à 2 téraoctets). et l es clusters sont déjà plus petits

Puisqu'une partition FAT32 peut contenir beaucoup plus de clusters qu'une partition FAT16, il est possible de réduire de façon signifiante la taille des clusters et de limiter par la même occasion le gaspillage d'espace disque. A titre d'exemple, pour une partition de 2Go, il est possible d'utiliser des clusters de 4Ko avec le système FAT32 (au lieu de 32Ko en FAT16), ce qui diminue l'espace gaspillée par un facteur 8.

Le nombre de clusters étant limité, la taille maximale d'une partition dépend de la taille de chaque cluster. Par ailleurs Il ne dispose pas de ;

- zone indiquant le propriétaire du fichier
- zone contenant la date du dernier accès en lecture

- droits d'accès de groupe.

- **NTFS (New Technology File System)**

NTFS (utilisé par Windows NT, 2000 et XP) utilise un système basé sur une structure appelée « table de fichiers maître », ou MFT (Master File Table), permettant de contenir des informations détaillées sur les fichiers. Ce système permet ainsi l'utilisation de noms longs, mais, contrairement au système FAT32, il est sensible à la casse, c'est-à-dire qu'il est capable de différencier des noms en majuscules de noms en minuscules. Pour ce qui est des performances, l'accès aux fichiers sur une partition NTFS est plus rapide que sur une partition de type FAT car il utilise une technique plus performante pour localiser les fichiers (un arbre binaire). La limite théorique de la taille d'une partition est de 16 exaoctets (17 milliards de To), mais la limite physique d'un disque est de 2To.

Linux reconnaît ce système à titre expérimental mais mieux vaut se limiter alors à la lecture sans y écrire pour éviter les pertes de données. Ce système n'est pas envisageable pour des disques inférieurs à 400 Mo car il requiert beaucoup de place pour la structure du système. La taille des clusters ne dépend pas de la taille du disque (ou de la partition). Enfin, ce système sait réparer automatiquement les secteurs défectueux. D'autre part, les droits d'administration sont pris en charge. En effet, C'est au niveau de la sécurité que NTFS prend toute son importance, car il permet de définir des attributs de sécurité pour chaque fichier. La version 5 de ce système de fichiers (en standard sous Windows 2000 alias NT 5) amène encore de nouvelles fonctionnalités parmi lesquelles des performances accrues,

NTFS permet donc de

- mettre des droits très spécifiques sur les fichiers et répertoires : lecture, écriture, exécution, appropriation, etc. ;

- chiffrer des fichiers avec EFS (*Encrypting File System) ,* - compresser des fichiers ;

- d'établir des quotas par volume.

B/ L'**extended file system** ou **ext**, est le premier système de fichiers créé en avril 1992 spécifiquement pour le système d'exploitation Linux Il a été conçu par Rémy Card pour surmonter certaines limitations du système de fichiers Minix. Il a plus tard été remplacé par ext2 et ext3 qui peuvent être utilisé depuis Windows.

- **Linux Ext2** (Ext2FS), utilisé par le système Linux et non reconnu par MS-DOS et tous les systèmes Windows. Les disques peuvent aller jusqu'à 2 Go et les noms de fichiers jusqu'à 255 caractères. Les droits d'administration sont pris en charge.
- **Linux Ext3** (Ext3FS), utilisé par le système Linux et non reconnu par MS-DOS et tous les systèmes Windows. Les disques peuvent aller jusqu'à 4 To. C'est une amélioration de l'Ext2FS auquel a été ajoutée la journalisation des fichiers afin de permettre de rattraper très vite toute corruption. (la ta Ce système vous permettra d'accéder sans problème à vos données Linux depuis Windows. ille maximale d'un fichier est de 2 TO

• II.12 Augmenter les Performances d'un SGF (cache)

En général, un programme d'application effectue plusieurs accès à un bloc d'un fichier : les parcours séquentiels sont fréquents ; des fichiers temporaires sont écrits dans une phase d'une application puis

relus par la suite, etc. On souhaite regrouper le plus possible les échanges avec le disque concernant un même bloc.
Le principe de la solution est simple : on conserve en mémoire centrale une copie d'un certain nombre de blocs de disque, cet ensemble étant appelé le cache du disque. Avant tout transfert, le SGF commence par rechercher si le bloc est déjà dans le cache ; si ce n'est pas le cas, il détermine un bloc à chasser du cache pour y charger le nouveau. La programmation du cache pose deux difficultés, le traitement des écritures(quand écrire sur disque) et le choix du bloc à effacer.

En l'absence de panne, on pourrait se contenter de recopier un bloc sur le disque au moment où on doit le chasser du cache. Compte tenu des grandes tailles des mémoires, ce sont en pratique des centaines ou des milliers de blocs que le SGF conserve dans le cache. Dans une utilisation individuelle d'un ordinateur, la plupart des fichiers manipulés par une application sont donc en totalité dans le cache. Autrement dit, une panne peut provoquer la perte d'une longue séance de travail. Le SGF doit donc recopier « sans trop tarder » les blocs modifiés, la solution maximaliste consistant à répercuter immédiatement chaque modification sur le disque (écriture immédiate ou « write through »).

Le choix du bloc à effacer est également complexe, car le SGF ne peut pas anticiper sur les futurs accès aux fichiers. On admet en général que les derniers blocs utilisés sont ceux qui ont la plus grande probabilité d'être récupérés. On peut alors chaîner les blocs dans l'ordre de leur utilisation et, quand le SGF a besoin d'un nouveau bloc, il efface le bloc le moins récemment utilisé.

Cet algorithme, appelé LRU pour « Least Recently Used » est d'emploi très fréquent dans les systèmes. Dans le cas du cache du disque, il peut être implanté de façon efficace (il suffit à chaque accès à un bloc de modifier son rang dans une liste). On peut aussi l'améliorer en tenant compte de la nature (fichier, catalogue de travail, catalogue intermédiaire) de l'élément auquel appartient le bloc.
Comme les accès séquentiels sont très fréquents, on peut également utiliser le cache en demandant par anticipation le chargement, dans le cache, du bloc suivant de chaque fichier.

Notion de fichiers journalisés
Un **journal** est la partie d'un système de fichiers journalisé qui trace les opérations d'écriture tant qu'elles ne sont pas terminées et cela en vue de garantir l'intégrité des données en cas d'arrêt brutal.

La grande taille des caches des disques permet de garantir que la plupart des accès aux fichiers se font sur des blocs présents dans le cache. Dans le cas des lectures, l'accès se limite au cache ; dans le cas des écritures, il faut bien répercuter sur le disque les modifications. Dans les implémentations traditionnelles, les blocs de fichiers résident à des adresses disque fixes et l'écriture d'un ensemble de blocs implique des déplacements du bras pendant lesquels aucun transfert n'a lieu.
L'idée introduite est de prévoir la notion des fichiers journalisés (« Log Structured File Systems ») consiste à regrouper les blocs modifiés dans un tampon en mémoire pour les écrire sur disque en une seule opération d'écriture dans des secteurs contigus, c'est à dire avec un seul positionnement du bras. Plus le nombre de blocs écrits n'est grand, meilleur sera le débit utile du disque.

III.13 Sécurité des fichiers

La sécurité peut être réalisée selon différents schémas :

- **Protection contre les accès inappropriés**

Elle est réalisée par des accès contrôlés en utilisant :
- Soit des mots de passe ou un mot de passe est associé à un fichier mais le problème se pose quand le nombre de fichier croit, une des solutions est d'associer un mot de passe à un répertoire

- Soit en utilisant des droits d'accès : lecture (r), écriture (w), exécution (x), destruction ...
A chaque fichier est associée une liste d'accès, spécifiant pour chaque utilisateur, les types d'accès qui lui sont autorisés. La liste d'accès peut être longue et difficile à gérer → définition de groupes auxquels sont associés des droits. Un utilisateur hérite des droits du groupe auxquels il appartient

- **Protection contre les dégâts physiques**

La seule façon de résister à des défaillances, qu'elles soient matérielles ou logicielles, est de conserver des informations de façon redondante. Cette redondance peut s'envisager à plusieurs niveaux :

1. Au niveau des structures de données sur disque, pour pouvoir reconstituer en cas d'effacement partiel d'un disque le vecteur des blocs libres ou les différents catalogues ou descripteurs de fichier,

2. Au niveau des fichiers par l'exécution régulière de copies de sauvegarde. Elle est réalisée au moyen de la redondance interne : L'information existe en double exemplaire : une version primaire, une version secondaire
Le système maintient la cohérence entre les deux versions exemple : MSDOS dispose de deux exemplaires de la FAT. La redondance est réalisée d'une manière périodique :
• sauvegarde complète : la totalité des objets est dupliquée même si ils n'ont pas été modifiés
• sauvegarde incrémentale : seuls les objets modifiés depuis la dernière sauvegarde sont dupliqués.

3. Au niveau des disques eux mêmes, avec des techniques RAID qui permettent de reconstituer des informations devenues inaccessibles,

En informatique, le mot **RAID** désigne une technologie permettant de stocker des données sur de multiples disques durs afin d'améliorer, en fonction du type de RAID choisi, la tolérance aux pannes et/ou les performances de l'ensemble.

RAID était à l'origine l'acronyme de ***Redundant Array of Inexpensive Disks***, ce qui signifie « matrice redondante de disques bon marché ». Aujourd'hui, le mot est devenu l'acronyme de ***Redundant Array of Independent Disks***, ce qui signifie « matrice redondante de disques indépendants »

La technologie **RAID** permet de constituer une unité de stockage à partir de plusieurs disques durs. L'unité ainsi créée (appelée grappe) a donc une grande tolérance aux pannes (haute disponibilité), ou bien une plus grande capacité/vitesse d'écriture. La répartition des données sur plusieurs disques durs permet donc d'en augmenter la sécurité et de fiabiliser les services associés.

Cette technologie a été mise au point en 1987 par trois chercheurs (Patterson, Gibson et Katz) à l'Université de Californie (Berkeley). Depuis 1992 c'est le RAID Advisory Board qui gère ces spécifications. Elle consiste à constituer un disque de grosse capacité (donc coûteux) à l'aide de plus petits disques peu onéreux (c'est-à-dire dont le MTBF (Mean Time Between Failure) soit le temps moyen entre deux pannes, est faible).
Les disques assemblés selon la technologie RAID peuvent être utilisés de différentes façons, appelées Niveaux RAID. L'Université de Californie en a défini 5, auxquels ont été ajoutés les niveaux 0 et 6. Chacun d'entre-eux décrit la manière de laquelle les données sont réparties sur les disques :

- Niveau 0: appelé striping
- Niveau 1: appelé mirroring, shadowing ou duplexing
- Niveau 2: appelé striping with parity (obsolète)
- Niveau 3: appelé disk array with bit-interleaved data
- Niveau 4: appelé disk array with block-interleaved data

- Niveau 5: appelé disk array with block-interleaved distributed parity
- Niveau 6: appelé disk array with block-interleaved distributed parity

Niveau 0

Le niveau RAID-0, appelé striping (traduisez entrelacement ou agrégat par bande,) consiste à stocker les données en les répartissant sur l'ensemble des disques de la grappe. De cette façon, il n'y a pas de redondance, on ne peut donc pas parler de tolérance aux pannes. En effet en cas de défaillance de l'un des disques, l'intégralité des données réparties sur les disques sera perdue.

Toutefois, étant donné que chaque disque de la grappe a son propre contrôleur, cela constitue une solution offrant une vitesse de transfert élevée.

Le RAID 0 consiste ainsi en la juxtaposition logique (agrégation) de plusieurs disques durs physiques. En mode RAID-0 les données sont écrites par "bandes" (en anglais stripes) :

Disque 1	**Disque 2**	**Disque 3**
Bande 1	Bande 2	Bande 3
Bande 4	Bande 5	Bande 6
Bande 7	Bande 8	Bande 9

On parle de facteur d'entrelacement pour caractériser la taille relative des fragments (bandes) stockés sur chaque unité physique. Le débit de transfert moyen dépend de ce facteur (plus petite est chaque bande, meilleur est le débit).

Si un des éléments de la grappe est plus grand que les autres, le système de remplissage par bande se trouvera bloqué lorsque le plus petit des disques sera rempli. La taille finale est ainsi égale au double de la capacité du plus petit des deux disques :

Deux disques de 20 Go donneront un disque logique de 40 Go alors qu'un disque de 10 Go utilisé conjointement avec un disque de 27 Go permettra d'obtenir un disque logique de 20 Go (17 Go du second disque seront alors inutilisés).

Cette architecture donne de meilleures performances mais n'assure en rien la sécurité des données ; en effet, si l'un des disques tombe en panne, la totalité des données du RAID est perdue.

> ***Remarque :*** Il est recommandé d'utiliser des disques de même taille pour faire du RAID-0 car dans le cas contraire le disque de plus grande capacité ne sera pas pleinement exploité

Exemple

On met en place un niveau RAID 0 au moyen de 4 disques avec les caractéristiques suivantes :

	Disque 1	**Disque 2**	**Disque 3**	**Disque 4**
Capacité	300 Go	**18Go**	20Go	45Go
Débit en lecture	90Mo/s	160Mo/s	**20Mo/s**	35Mo/s
Débit en écriture	82Mo/s	125Mo/s	20Mo/s	**15Mo/s**

Les valeurs les plus faibles pour chacune des caractéristiques ont été misent en gras sur le tableau ci-dessus.

Lorsque l'on branche ces disques sur une carte RAID et que l'on défini un niveau RAID 0, on obtient le disque virtuel suivant :

Capacité: 4*18Go = 72Go ; Débit en lecture : 4*20Mo/s = 80Mo/s

Débit en écriture : 4*15Mo/s = 60Mo/s ;

Le disque virtuel composé des quatre disques physiques s'avère donc plus lent et plus petit que le disque dur N°1 utilisé seul. Cet exemple est donc une utilisation catastrophique du niveau RAID 0.

Niveau 1

Un système RAID 1 duplique les données sur tous les disques. Tout comme le RAID 0, le RAID 1 nécessite un minimum de deux disques durs pour fonctionner. Les disques ont un contenu identique ce qui implique une perte d'espace proportionnelle au nombre de disques rajoutés. En effet, si on met 3 disques de 40 Go en RAID 1 alors les 40 Go écris sur le disque 1 seront copiés sur les deux autres disques. On perdra donc 2 * 40 = 80 Go d'espace disque sur la capacité totale des 3 disques (120 Go).

En cas de dysfonctionnement de l'un des disque dur (cela arrive fréquemment sur des serveurs recevant beaucoup de requêtes par secondes), le contrôleur RAID utilisera alors uniquement les disques en état de marche. Le RAID 1 apporte donc la tolérance de pannes. L'utilisation du RAID 1 est particulièrement recommandée sur des serveurs stockant des données sensibles qui doivent rester fonctionnels 24 heures sur 24.
Lorsque l'on met en place un RAID 1 avec des disques dur de capacités différentes, alors la capacité de stockage de l'ensemble en RAID 1 sera celle du disque de plus faible capacité.

Lorsqu'un accès en lecture est réalisé sur un ensemble de disques en mode RAID 1, alors tous les disques lisent chacun une partie de la données ce qui améliore grandement les performances en lecture. En revanche lors de l'écriture sur un ensemble de disques RAID 1, les performances ne sont pas améliorées. En effet, les disques vont écrire les données de façon synchronisée et ce au fur et à mesure que les données vont arriver au contrôleur. Le travail lors d'un accès en écriture n'est donc pas divisé entre les disques mais est commun à tous les disques. Le débit en écriture d'un ensemble RAID 1 est donc le même que celui d'un disque seul. C'est pourquoi dans un ensemble RAID 1, contenant 5 disques les accès en lecture seront grandement améliorés alors que les accès en écriture stagneront.
Le niveau 1 a pour but de dupliquer l'information à stocker sur plusieurs disques, on parle donc de mirroring, ou shadowing pour désigner ce procédé.

Disque1	**Disque2**	**Disque3**
Bande 1	Bande 1	Bande 1
Bande 2	Bande 2	Bande 2
Bande 3	Bande 3	Bande 3

On obtient ainsi une plus grande sécurité des données, car si l'un des disques tombe en panne, les données sont sauvegardées sur l'autre. D'autre part, la lecture peut être beaucoup plus rapide lorsque les deux disques sont en fonctionnement. Enfin, étant donné que chaque disque possède son propre contrôleur, le

serveur peut continuer à fonctionner même lorsque l'un des disques tombe en panne, au même titre qu'un camion pourra continuer à rouler si un de ses pneus crève, car il en a plusieurs sur chaque essieu...

En contrepartie la technologie RAID1 est très onéreuse étant donné que seule la moitié de la capacité de stockage n'est effectivement utilisée.

Niveau 2

Le RAID 2 est une alternative au RAID 1. En effet, le RAID 2 est identique au RAID 0 sauf sur un point : lors des opérations de lecture, un seul des disques travaille ce qui permet aux autres disques de se « reposer ». Les autres disques étant moins sollicités, cela augmente leur durée de vie. En outre le RAID 2 implémente un système de corrections des erreurs. Cependant, cela est devenu inutile car tous les disques durs SCSI intègrent dorénavant leur propre système de correction d'erreurs. Le RAID 2 augmente donc la fiabilité du RAID 1 mais induit une baisse des performances. Il a peu été utilisé et n'est plus utilisé de nos jours à cause de son obsolescence.

Niveau 3

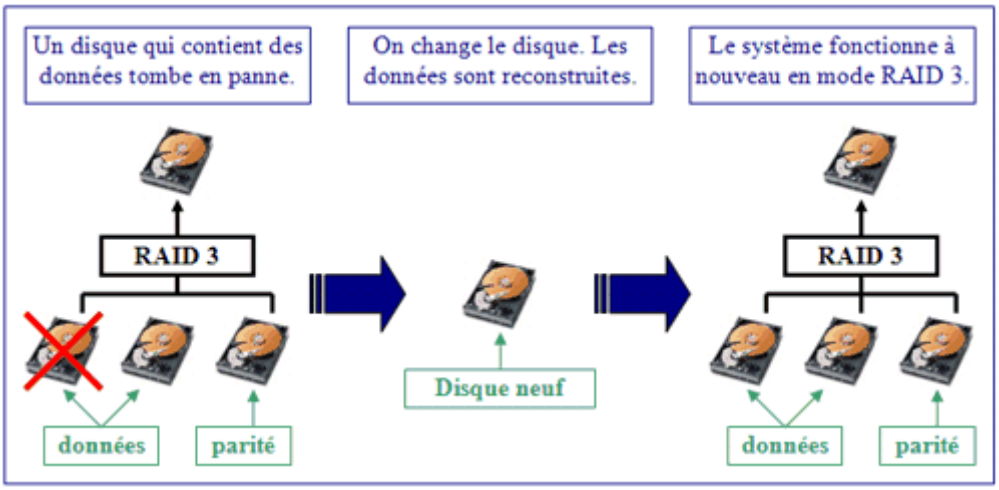
Le RAID 3 apporte les avantages du RAID 0 (amélioration des performances) et du RAID 1 (tolérance de panne). Le mode RAID 3 fonctionne avec un minimum de trois disques. A l'instar du RAID 0, tous les disques sauf un qui a une tâche spéciale se divisent les opérations de lecture et d'écriture ce qui augmente les performances. En RAID 3, la taille des segments n'est pas modifiable et est fixée à 512 octets (en RAID 3 : un segment = un secteur de disque dur = 512 octets). L'un des disques est un disque de parité.

Disque 1	**Disque 2**	**Disque 3**	**Disque 4**
Octet 1	Octet 2	Octet 3	Parité 1+2+3
Octet 4	Octet 5	Octet 6	Parité 4+5+6
Octet 7	Octet 8	Octet 9	Parité 7+8+9

La parité est un processus algorithmique utilisant l'intégrale de parité inventée par Gérard Langlet. Ses applications sont très nombreuses en informatique et en particulier dans le domaine du RAID. Avec un ensemble RAID3, lors de chaque opération d'écriture les données vont êtres réparties entres tous les disques (sauf le disque de parité) de façon équitable.

Au moment où les données sont écrites, des informations indiquant sur quel disque a été stocké tel ou tel fragment des données sont enregistrées dans le disque de parité. Ce sont les informations de parité. Ainsi, si un fragment d'une donnée est défectueux ou manquant, la comparaison entre les informations de parité et les autres fragments de la donnée (qui sont stockés sur les autres disques durs) vont permettre de reconstituer le fragment.

Si l'un des disques contenant des données tombe en panne, alors le disque de parité et les disques restant permettent de reconstruire les données qui étaient stockées sur le disque endommagé.



Lorsque le disque de parité tombe en panne le système continue de fonctionner en mode RAID 0 avec les disques restants qui contiennent les données. Lorsque l'on rajoute un nouveau disque de parité, les informations de parité sont reconstruites et le RAID 3 redevient actif.

Niveau 4

Ce mode nécessite au moins trois disques et est très proche du RAID 3. La seule différence notable avec le RAID 3 est l'organisation des données. En effet, avec le RAID 4, la taille des segments est variable et se modifie en temps réel. Cela implique que les informations de parité doivent être mise à jour à chaque écriture afin de vérifier si la taille des segments a été modifiée. Dans un système RAID 4, le disque de parité devient donc un facteur encore plus limitant lors des opérations d'écriture. Cependant, les performances en lecture sont les même qu'avec le RAID 3 (c'est-à-dire excellentes). Le mode RAID 4 gère donc la tolérance de panne et apporte un gain en lecture mais pas en écriture. Généralement on lui préfère le RAID 5.

Disque 1	Disque 2	Disque 3	Disque 4
Bloc 1	Bloc 2	Bloc 3	Parité 1+2+3
Bloc 4	Bloc 5	Bloc 6	Parité 4+5+6
Bloc 7	Bloc 8	Bloc 9	Parité 7+8+9

Niveau 5

Ce mode est très proche du RAID 4 car il utilise au moins trois disques durs et il procure une augmentation des performances et gère la perte d'un disque dur.

Par rapport au RAID 4, la configuration RAID 5 répartie la parité sur l'ensemble des disques ce qui élimine le goulot d'étranglement qu'est le disque de parité en RAID 3 et en RAID 4. Un autre avantage de répartir la parité sur chacun des disques est que les disques travaillent tous autant. Cela empêche l'usure prématurée de l'un des disques (comme c'est le cas avec le disque de parité en RAID 3 et surtout en RAID 4).

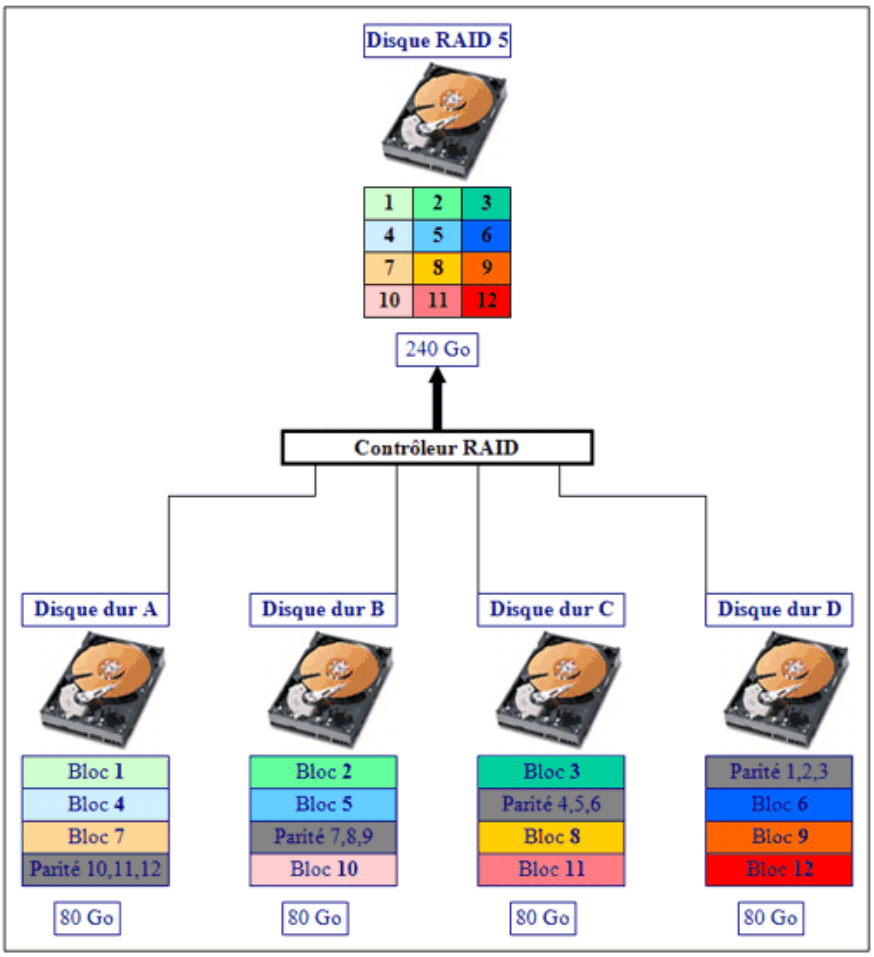
Le RAID 5 écrit donc simultanément les données sur plusieurs disques ce qui améliore les performances en lecture et en écriture. Ainsi par rapport à un disque seul, les performances d'une grappe RAID 5 utilisant n disques seront (n-1) fois plus élevées (en effet, l'un des disques doit écrire la parité, la donnée écrite est donc répartie sur les n-1 disques restants).

Tout comme pour le mode RAID 0, on doit sélectionner une taille de segment adaptée lorsque l'on met en place un niveau RAID 5.

Dans l'exemple suivant on a crée une grappe RAID 5 avec quatre disques durs identiques ayant une capacité de 80 Go chacun. Au final seulement 240 Go sont exploitables, ce qui représente les ¾ de l'espace disque total (320 Go). En fait l'espace d'un disque entier soit 80 Go est nécessaire afin de stocker les informations de parité. En RAID 5, on peut calculer la capacité utilisable de la manière suivante :

Capacité utilisable = (taille du disque le plus petit) * (nombre de disques – 1)

Ainsi la capacité utilisable est bien de 80 Go * (4-1) disques = 240 Go. On remarque bien sur le schéma de quelle façon les informations de parité sont réparties sur les 4 disques.



De par ses nombreux avantages le RAID 5 est très populaire dans le monde professionnel car il apporte la tolérance de panne tout en conservant d'excellentes performances en lecture et en écriture. De plus si on utilise une interface qui supporte le branchement à chaud ou "hot Plug" (SCSI ou Sata), il n'est même plus nécessaire d'éteindre le serveur. Bien entendu la mise en place d'un ensemble RAID 5 fait perdre de l'espace disque (en fait l'espace réservé à la parité qui représente la taille d'un disque). De plus la répartition de la parité sur l'ensemble des disques entraîne une remise en ordre un peu lente lorsqu'un disque dur est échangé suite à une panne.

De cette façon, RAID 5 améliore grandement l'accès aux données (aussi bien en lecture qu'en écriture) car l'accès aux bits de parité est réparti sur les différents disques de la grappe.
Le mode RAID-5 permet d'obtenir des performances très proches de celles obtenues en RAID-0, tout en assurant une tolérance aux pannes élevée, c'est la raison pour laquelle c'est un des modes RAID les plus intéressants en termes de performance et de fiabilité.

Remarque : L'espace disque utile sur une grappe de n disques étant égal à n-1 disques, il est intéressant d'avoir un grand nombre de disques pour "rentabiliser" le RAID-5.

Niveau 6

Le RAID 6 met en place une double redondance des données de parité. Cela signifie que les informations de parité sont stockées en doubles exemplaires. Tout comme pour le RAID 5, les informations de parité sont réparties équitablement sur l'ensemble des disques. La double redondance des données permet la perte de deux disques sans qu'aucune donnée ne soit perdue. Le RAID 6 nécessite au moins quatre disques durs pour fonctionner. En ce qui concerne la capacité utilisable de stockage, on perd l'espace de deux disques. Donc dans un système utilisant n disques, la capacité utilisable est définie par la relation suivante :Capacité utilisable = (taille du disque le plus petit) * (nombre de disques – 2)Les performances en lecture comme en écriture sont augmentées. Avec un nombre de disques équivalents, un ensemble RAID 6 sera moins performant qu'un ensemble RAID 5 (car un disque de plus est utilisé pour la parité).

Niveau 7

Le **RAID 7** est utilisé pour avoir un niveau de sécurité ou des performances très supérieures aux autres modes RAID. Il utilise un grand nombre de disques (avec un maximum de 48 disques). On peut définir manuellement le nombre de disques dédiés au stockage de la parité et au stockage des données. Le nombre de disques que le système peut perdre est proportionnel au nombre de disques dédiés au stockage des informations de parité. Le RAID 7 fait appel à une carte microprocesseur qui calcule la parité, la gestion du disque et qui gère la surveillance des disques en temps réel. Tous les transferts de données se font en mode asynchrone ce qui augmente de 1,5 à 6 fois les performances en écriture.

Comparaison

Les solutions RAID généralement retenues sont le RAID de niveau 1 et le RAID de niveau 5.

Le choix d'une solution RAID est lié à trois critères :

- **la sécurité :** RAID 1 et 5 offrent tous les deux un niveau de sécurité élevé, toutefois la méthode de reconstruction des disques varie entre les deux solutions. En cas de panne du système, RAID 5 reconstruit le disque manquant à partir des informations stockées sur les autres disques, tandis que RAID 1 opère une copie disque à disque.
- **Les performances :** RAID 1 offre de meilleures performances que RAID 5 en lecture, mais souffre lors d'importantes opérations d'écriture.
- **Le coût :** le coût est directement lié à la capacité de stockage devant être mise en œuvre pour avoir une certaine capacité effective. La solution RAID 5 offre un volume utile représentant 80 à 90% du volume alloué (le reste servant évidemment au contrôle d'erreur). La solution RAID 1 n'offre par contre qu'un volume disponible représentant 50 % du volume total (étant donné que les informations sont dupliquées).

Mise en place d'une solution RAID

Il existe plusieurs façons différentes de mettre en place une solution RAID sur un serveur :

- **de façon logicielle** : il s'agit généralement d'un driver au niveau du système d'exploitation de l'ordinateur capable de créer un seul volume logique avec plusieurs disques.
- **de façon matérielle**
    - avec des matériels DASD (Direct Access Stockage Device) : il s'agit d'unités de stockage externes pourvues d'une alimentation propre. De plus ces matériels sont dotés de connecteurs permettant l'échange de disques à chaud (on dit généralement que ce type de disque est hot swappable). Ce matériel gère lui-même ses disques, si bien qu'il est reconnu comme un disque SCSI standard.
    - avec des contrôleurs de disques RAID : il s'agit de cartes s'enfichant dans des slots PCI ou ISA et permettant de contrôler plusieurs disques durs.

Conclusion

Pour conclure, les modes les plus intéressants sont les modes 0, 1 et 5. Les autres modes sont peu ou pas utilisés car :

☐ le RAID 2 est obsolète
☐ le RAID 3 et 4 utilisent un disque de parité ce qui crée un goulot d'étranglement.
☐ le RAID 6 est onéreux
☐ le RAID 7 est très onéreux et difficile à mettre en place

Voici un petit tableau comparatif des trois modes les plus intéressants fonctionnant avec 10 disques durs d'une capacité unitaire de 120 Go et ayant un débit réel en lecture/écriture de 70/50 Mo/s.

	RAID 0	RAID 1	RAID 5
Capacité	1200 Go (100 %)	120 Go (10 %)	1080 Go (90%)
débit en lecture	700 Mo/s	700 Mo/s	620 Mo/s
débit en écriture	500 Mo/s	50 Mo/s	450 Mo/s
Tolérance de panne(nombre de disques)	aucun	jusqu'à 9 disques	un disque

Le système ayant le meilleur rapport capacité/performances/sécurité est sans conteste le RAID 5. C'est d'ailleurs lui le plus utilisé en Entreprise. Cependant, il est possible de combiner différents niveaux de RAID, ce qui permet d'obtenir des alternatives très intéressantes au RAID 5. Dans la seconde partie nous allons voir ces niveaux combinés.