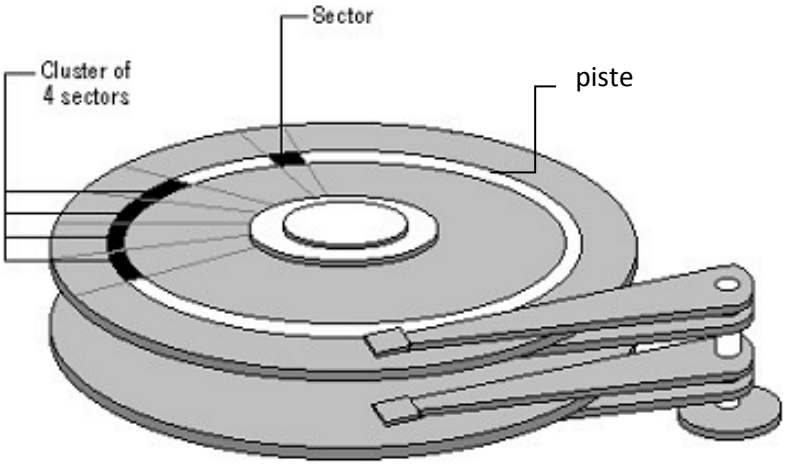
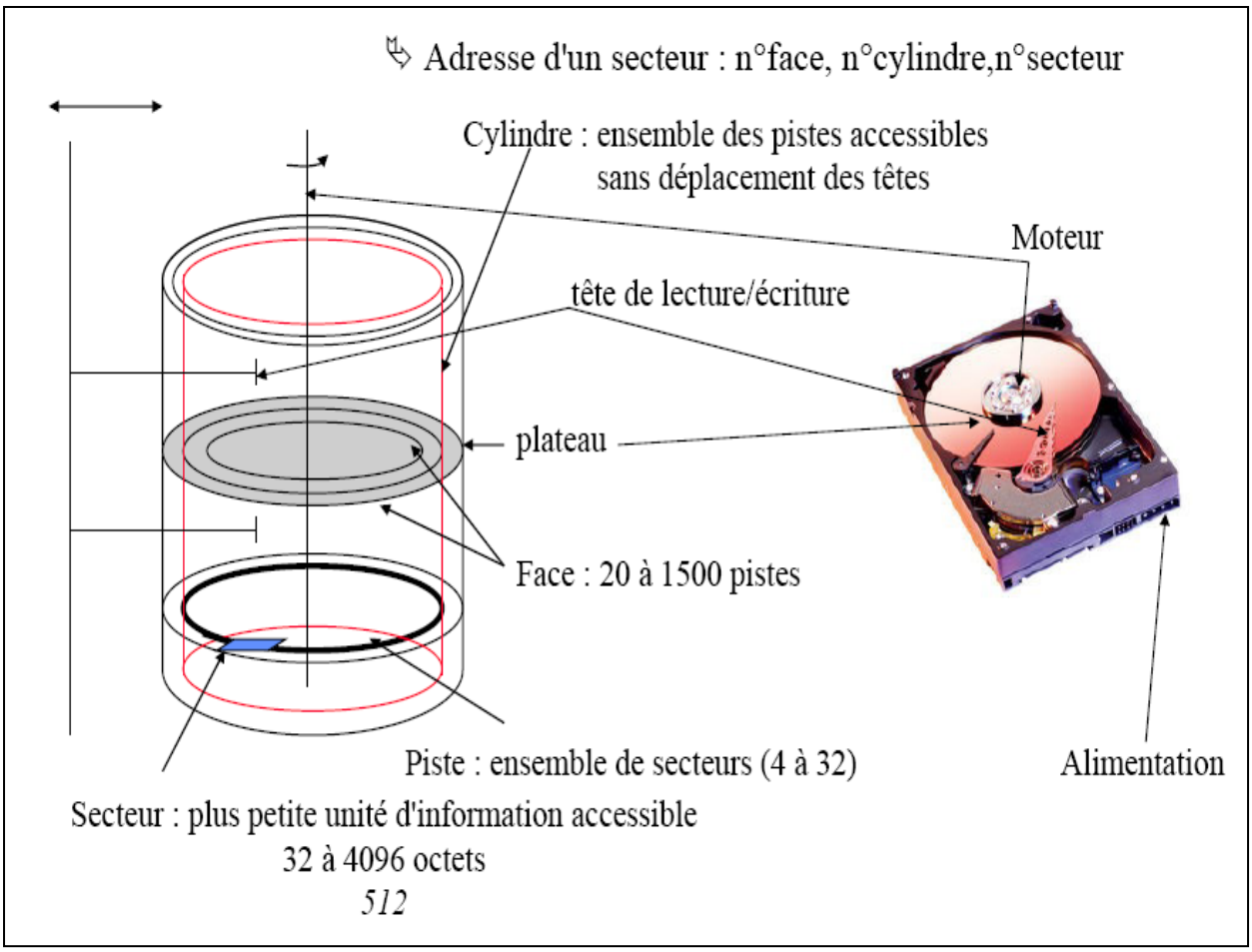
SYSTEME DE GESTION DE FICHIERS (SGF)



Disque dur



SGF est la partie la plus visible d'un système d'exploitation qui se charge de gérer le stockage et la manipulation de fichiers (sur une unité de stockage : partition, disque, CD, disquette.

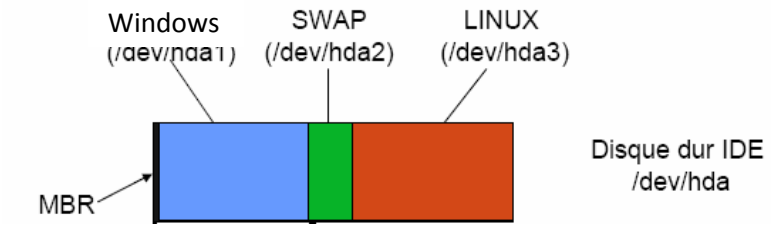
Formatage et Partitionnement

A- Partitionnement : permet la cohabitation de plusieurs systèmes d'exploitation sur le même disque. L'information sur le partitionnement d'un disque est stockée dans son premier secteur (secteur zéro), le MBR (Master Boot Record).

Deux types de partitionnement :

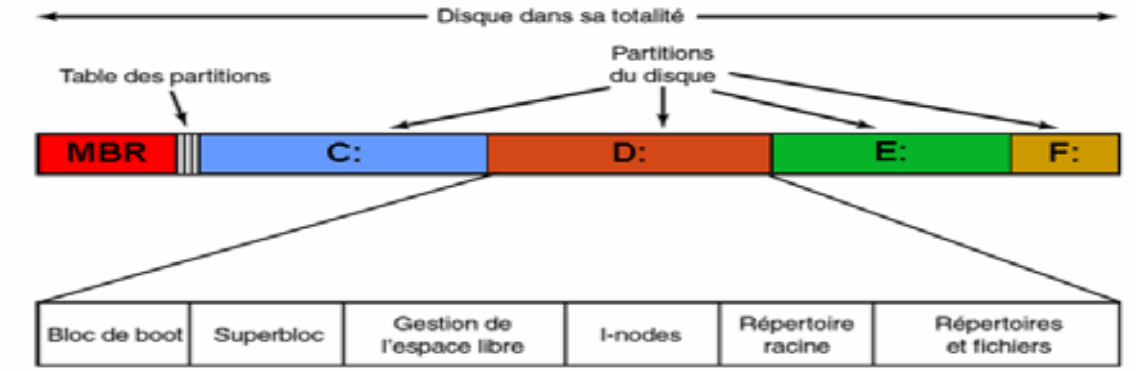
- Primaire : créer jusqu'à 4 partitions primaires sur un disque.
- Etendue : diviser une partition primaire en sous-partitions (une ou plusieurs **partitions logiques** qui se comportent comme les partitions primaires (pas de secteurs de démarrage))

Dans un même disque, on peut avoir un ensemble de partitions (multi-partition), contenant chacune un système de fichier (par exemple Windows et Linux)



Multipartition d'un disque

B-Formatage : Avant qu'un système de fichiers puisse créer et gérer des fichiers sur une unité de stockage, son unité doit être formatée selon les spécificités du système de fichiers. Le formatage inspecte les secteurs, efface les données et crée le répertoire racine du système de fichiers. Il crée également un superbloc pour stocker les informations nécessaires à assurer l'intégrité du système de fichiers.



Organisation du système de fichier

Le MBR est un élément important du système qui contient :

- Les informations sur le disque (nbre de secteurs par cluster, par piste….)
- La table des partitions
- Le programme lancé par le BIOS (Bootstrap pour le SE)

Un superbloc contient:

- L'identifiant du système de fichiers (C:, D : ..),
- Le nombre de blocs dans le système de fichiers,
- La liste des blocs libres,
- l'emplacement du répertoire racine,
- la date et l'heure de la dernière modification du système de fichiers,

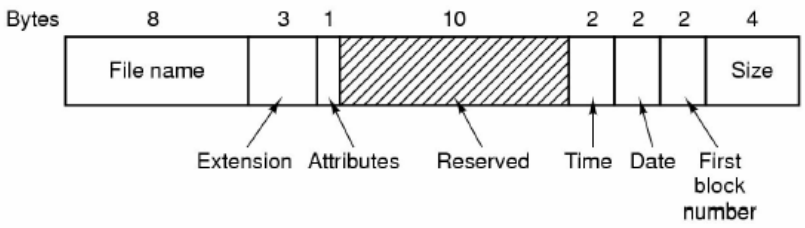
Le concept de fichier

- C'est l'unité de stockage logique mise à la disposition des utilisateurs pour l'enregistrement de leurs données.
- Les fichiers sont généralement créés par les utilisateurs, certains fichiers sont générés par les systèmes ou certains outils tels que les compilateurs.
- Chaque fichier a un ensemble d'attributs qui le décrivent.
  - Le nom,
  - l'extension,
  - la date et l'heure de sa création ou de sa dernière modification,
  - la taille, la protection.

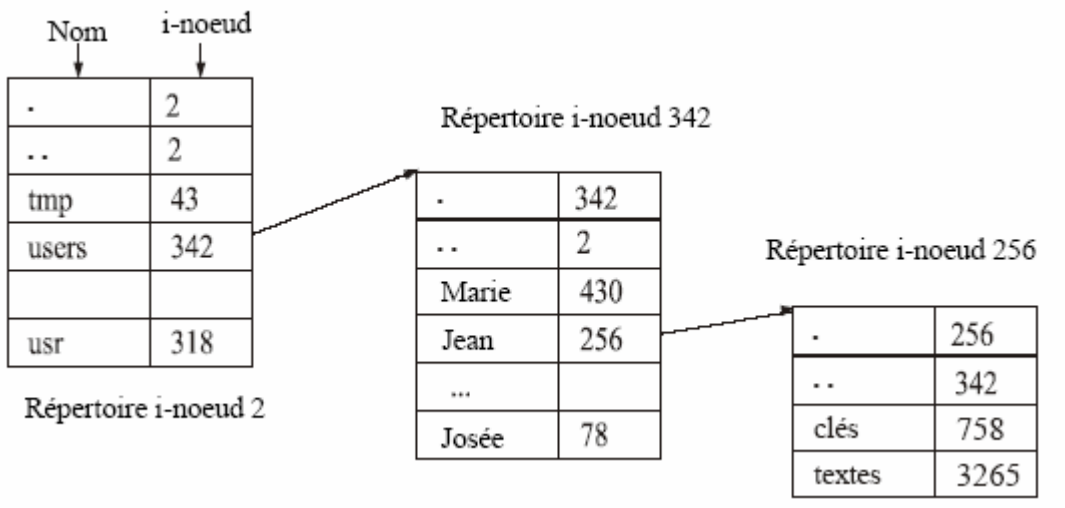
La notion de répertoire

- Un répertoire (**catalogues** ou **directory)** est lui-même un fichier puisqu'il est stocké sur le disque et est destiné à contenir des fichiers.
- Du point de vue SGF, un répertoire est considéré comme un tableau qui contient une entrée par fichier. Chaque entrée peut contenir des informations sur le fichier (attributs du fichier) ou faire référence à (pointer sur) des structures qui contiennent ces informations.

Exemple



Structure d'un répertoire : cas de MS-DOS (32 octets)



Structure d'un répertoire : cas d'UNIX (14 octets)

Le nom complet d'un fichier est formé d'une liste des répertoires qu'il faut traverser à partir du haut de la hiérarchie (le *répertoire racine* (root directory)) plus le nom_du_fichier. Les répertoires sont séparés par un caractère qui dépend du système d'exploitation : "/" pour UNIX, " \" pour Dos et Windows.
Un tel chemin (exprimé à partir de la racine) est appelé chemin **absolu**. Voici un exemple de chemin absolu sous MS-DOS c:\cours\chapitre4.txt et sous Unix /home/user1/rapport.txt. Par contre, un chemin qui ne commence pas par la racine est un chemin **relatif.**

En Unix, le répertoire racine (le répertoire /) contient les sous répertoires suivants :

/bin	commandes binaires utilisateur essentielles (pour tous les utilisateurs)
/boot	fichiers statiques du chargeur de lancement
/dev	fichiers de périphériques
/etc	configuration système spécifique à la machine
/home	répertoires personnels des utilisateurs
/lib	bibliothèques partagées essentielles et modules du noyau
/mnt	point de montage pour les systèmes de fichiers montés temporairement
/proc	système de fichiers virtuel d'information du noyau et des processus
/root	répertoire personnel de root (optionnel)
/sbin	binaires système (binaires auparavant mis dans /etc)
/sys	*état des périphériques (model device) et sous-systèmes (subsystems)*
/tmp	fichiers temporaires

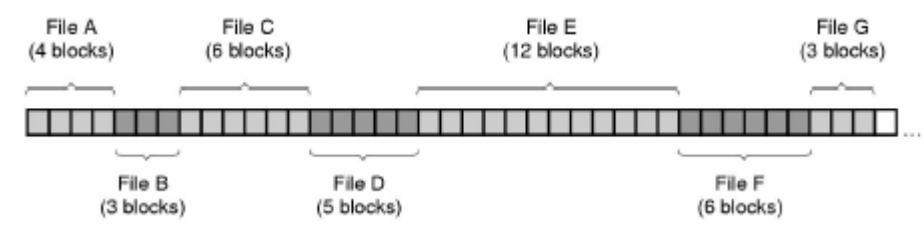
La gestion de l'organisation de l'espace disque

- Un fichier est sauvegardé sur un ensemble de clusters (blocs).
- Le SGF manipule alors des blocs numérotés de 0 à N-1 (N = taille du disque/taille d'un bloc).
- Chaque fichier (ordinaire ou répertoire) d'un système de fichiers est stocké sur l'unité de stockage du système de fichiers.
- Ses données sont dans des blocs de taille fixe (512, 1024, ou 2048 octets, …) et à chaque fichier est alloué un nombre de blocs.

La lecture ou l'écriture d'un élément d'un fichier impliquera le transfert vers la mémoire du bloc entier qui contient cet élément.

Techniques d'allocation des blocs sur le disque

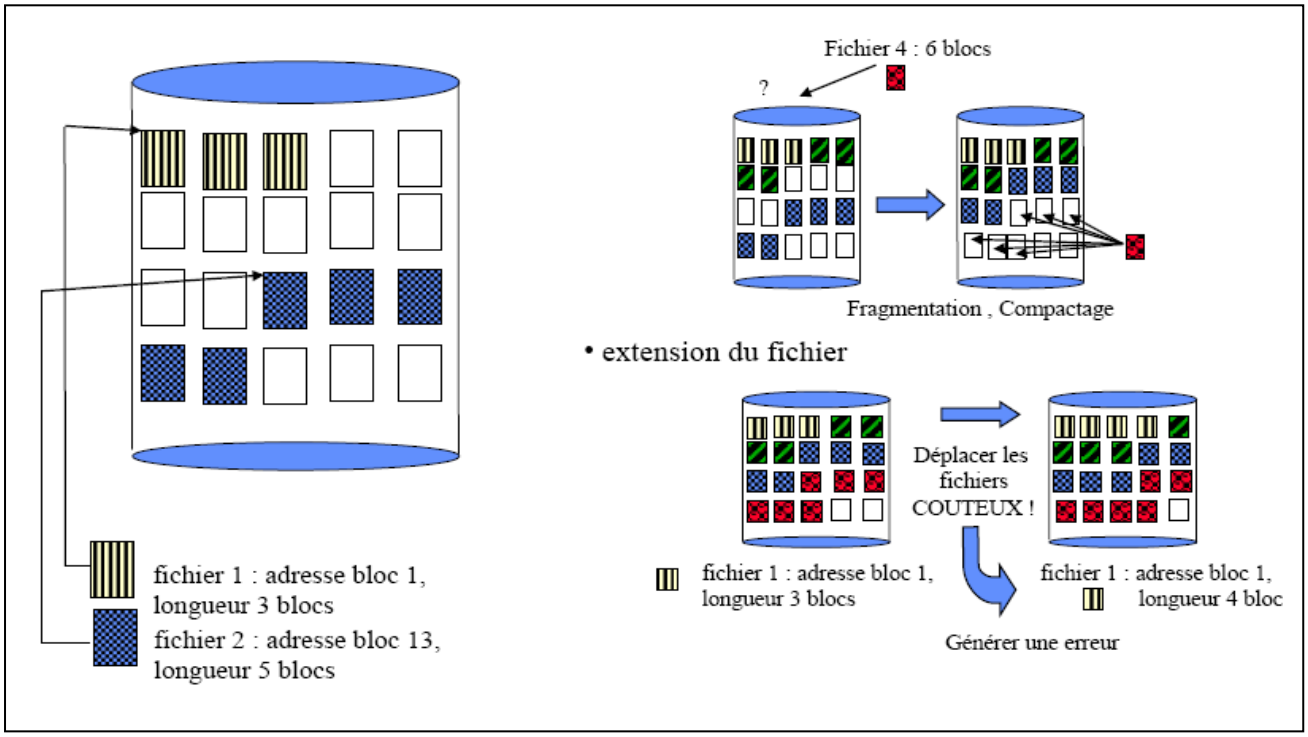
- ***Allocation contiguë*** : Le fichier est constitué de plusieurs blocs contigus.



Table

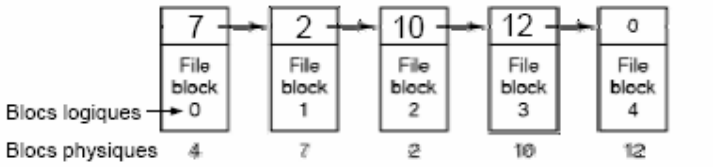
Fichier	Début	Fin
A	0	3
B	4	6
C	7	12
D	13	17
E	18	29
F	30	35
G	36	38

Allocation contiguë d'espace disque pour 7 fichiers



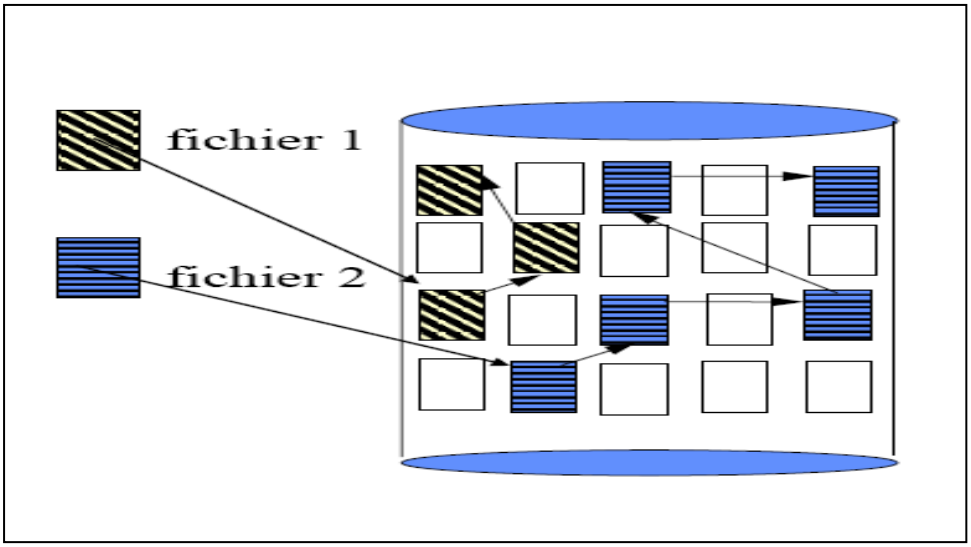
- Avantage : la rapidité d'accès (les blocs étant contigus, on limite les déplacements de la tête le lecture/écriture, coûteux en temps).
- Inconvénients :
  - ✓ **gaspille de la place**. Le pourcentage de place perdue est d'autant plus grand que la taille moyenne des fichiers est faible (difficile de prévoir la taille qu'il faut réserver au fichier).
  - ✓ **fragmentation externe** : c'est l'espace perdu entre les fichiers. On peut effacer des données ou supprimer des fichiers ce qui libère des blocs sur le disque.

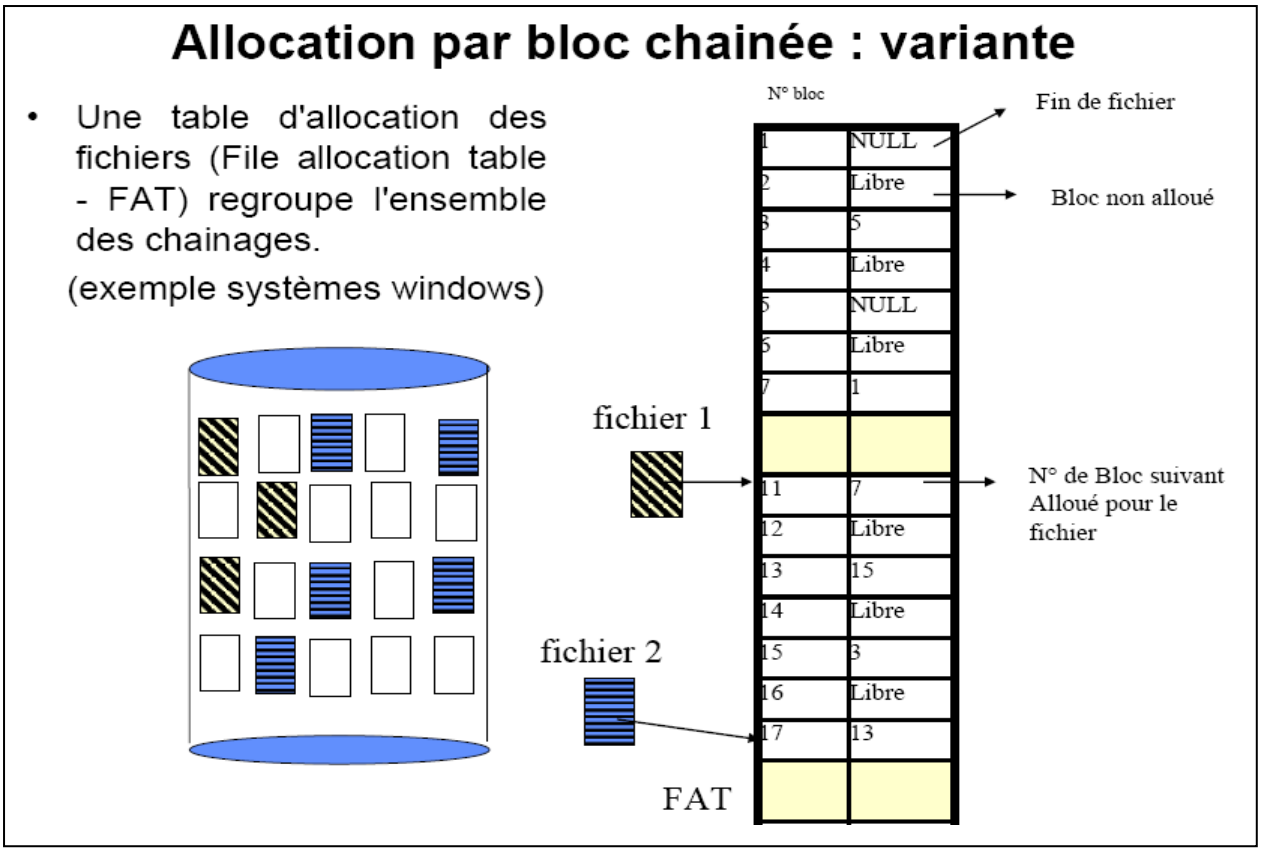
- ***Allocation chaînée (non contiguë) : A***llouer des blocs chaînés entre eux aux fichiers.



Allocation chaînée

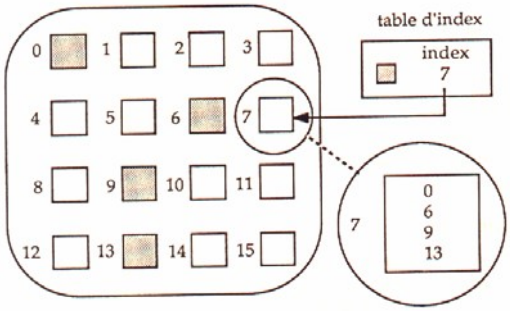
- **Avantage** : l'élimination du problème de fragmentation externe.
- **Inconvénients** :
  - ✓ L'accès au fichier est totalement séquentiel,
  - ✓ La perte d'un chaînage entraîne la perte de tout le reste du fichier.



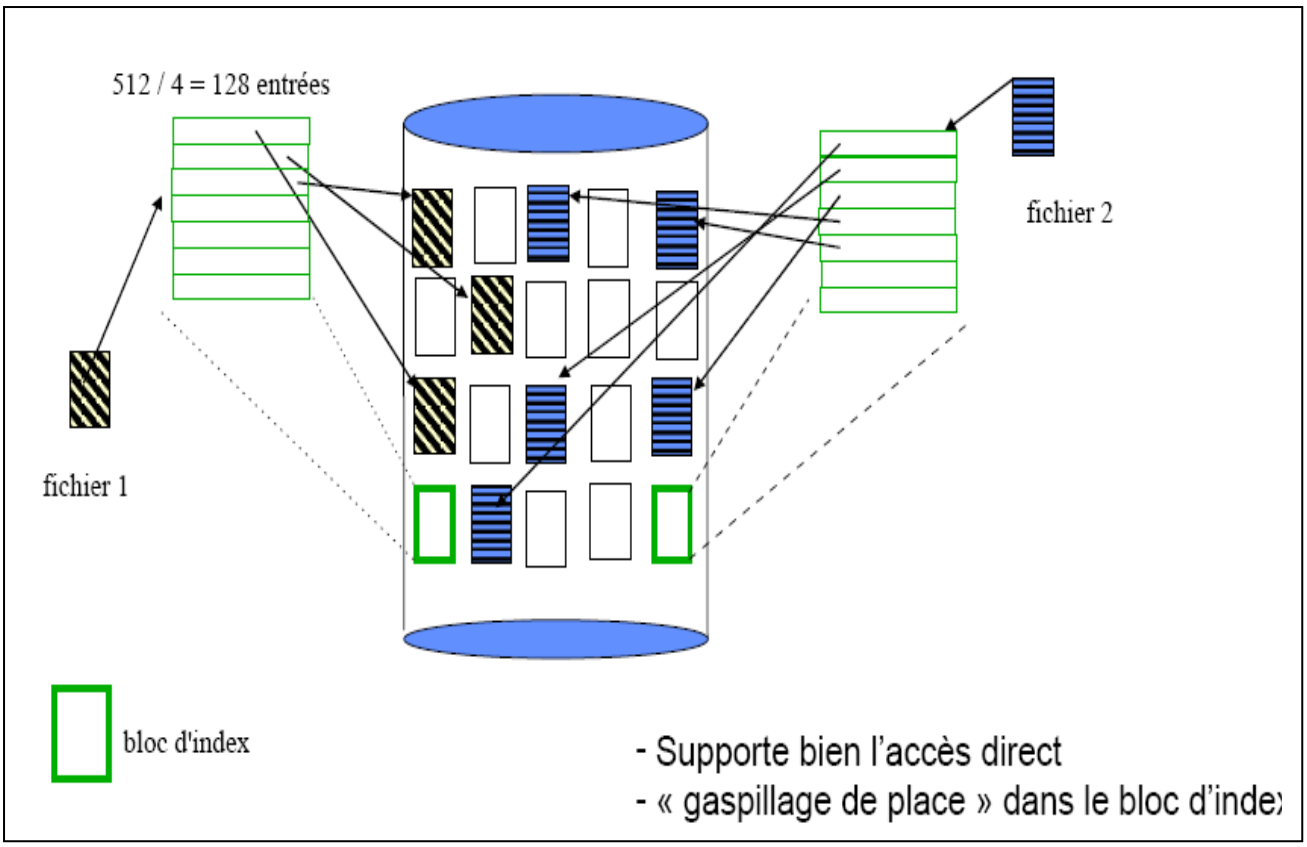


La plus part des systèmes actuels appliquent ce mode. MS-DOS utilise la **FAT** (File Allocation Table) pour y conserver les chaînages entre les blocs. Windows NT utilise la **MFT** (Master File Table) associé au système **NTFS** (New Technology File System) .UNIX, GNU/Linux utilisent le **I-Node** (Index node).

- ***Allocation non contiguë indexée*** : les inconvénients de l'allocation chaînée peuvent être résolus d'une manière simple : les pointeurs des blocs sont dans une structure de données gardée en mémoire centrale, ainsi, les informations sur les numéros de blocs peuvent être obtenues à tout moment.



Allocation indexée



FAT (File Allocation Table)

On parle généralement de système de fichiers **FAT16** et **FAT32.**

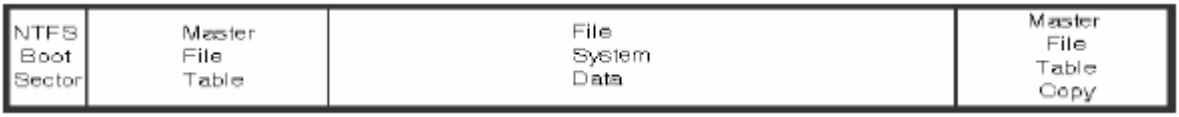
- Le FAT16 est utilisé par MS-DOS. En FAT16, les numéros de blocs sont écrits sur 16 bits. Si on suppose que la taille d'un bloc est 32Ko, la taille maximale adressable est alors 2Go (2^{16} x 32 Ko = 2097152 Ko = 2Go)

- Le FAT32 est pris en charge par Windows 95 et les versions qui ont suivis. Les numéros de blocs sont écrits sur 32 bits (en réalité, sur 28bits, 4 bits étant réservés). Si on suppose que la taille d'un bloc est de 32 ko, la taille maximale adressable théoriquement est de 8 To (2^{28} x 32 Ko = 8 To). Toutefois, Microsoft la limite volontairement à 32 Go sur les systèmes Windows 9x afin de favoriser NTFS.

NTFS (New Technology File System)

- Le système de fichiers NTFS est utilisé par Windows2000, WindowsNT, Windows XP et Windows Vista.
- Il utilise un système basé sur une structure appelée 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.



Partition NTFS

Structure d'un I-Node

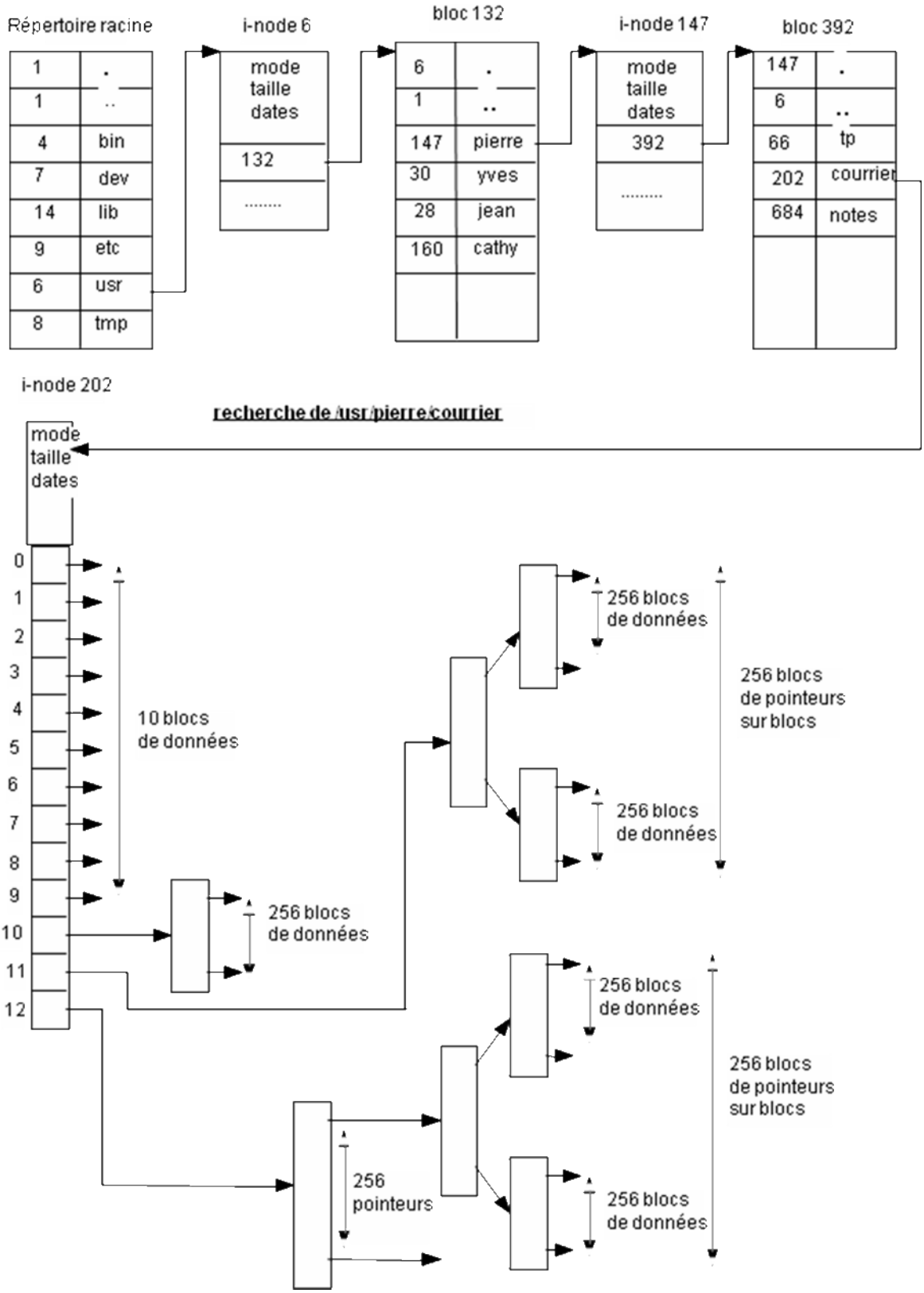
La structure d'I-Node est utilisée par le système de gestion de fichier **ext3fs** d'Unix ou GNU/Linux (ext3fs pour *third extented file system*). Un nœud d'index est constitué d'attributs décrivant le fichier ou le répertoire et d'adresses de blocs contenant des données. Cette structure possède plusieurs entrées, elle permet au système de disposer d'un certain nombre de données sur le fichier :

- la taille,
- l'identité du propriétaire et du groupe : un fichier en Unix est crée par un propriétaire, qui appartient à un groupe,
- Les droits d'accès : pour chaque fichier, Unix définit trois droits d'accès (lecture (r), écriture (w) et exécution (x)) pour chaque classe d'utilisateurs (trois types d'utilisateur {propriétaire, membre du même groupe que le propriétaire, autres}). Donc à chaque fichier, Unix associe neuf droits,
- les dates de création, de dernière consultation et de dernière modification,
- le nombre de références existant pour ce fichier dans le système,
- les dix premiers blocs de données,
- d'autres entrées contiennent l'adresse d'autres blocs (on parle alors de bloc d'indirection) :
    - une entrée pointe sur un bloc d'index qui contient 128 ou 256 pointeurs sur bloc de données(simple indirection)
    - Une entrée pointe sur un bloc d'index qui contient 128 ou 256 pointeurs sur bloc d'index dont chacun contient 128 ou 256 pointeurs sur bloc de données (double indirection)
    - Une entrée pointe sur un bloc d'index qui contient 128 ou 256 pointeurs sur bloc d'index dont chacun contient 128 ou 256 pointeurs sur bloc d'index dont chacun contient 128 ou 256 pointeurs sur bloc de données (triple indirection)

La structure d'I-Node est conçue afin d'alléger le répertoire et d'en éliminer les attributs du fichier ainsi que les informations sur l'emplacement des données.
Une entrée dans un I-Node d'un répertoire contiendra donc un nom d'un fichier ou sous-répertoire et l'I- Node associé.

Exemple

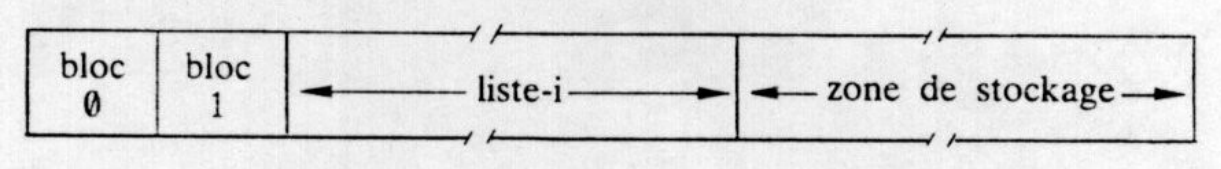
Si on suppose que la taille d'un bloc est de 1Ko, un fichier sous Unix peut avoir la taille maximale suivante : 10 x 1Ko + 256 x 1Ko + 256 x 256 x 1Ko + 256 x 256 x 256 x 1Ko, ce qui donne en théorie plus de 16Go.
Pour les fichiers les plus longs, trois accès au disque suffisent pour connaître l'adresse de tout octet du fichier.



Organisation des disques / Structure du disque

C'est à la création du système de fichier que l'organisation structurelle du FS est créée afin de permettre une gestion de ces fichiers.

Un disque est vu par le système comme un tableau de blocs (de 512 octets ou plus suivant la taille de base des blocs). Le SGF découpe le disque en 4 régions distinctes.



Les régions d'un disque UFS.

Voici les composants de cette structure :

- Boot Bloc ou Bloc 0 : Utilisé au chargement du système. Il contient des informations sur la structure du disque et aussi le « boot loader » qui permet de démarrer le système.

- Super Bloc ou Bloc 1 :Informations générales sur le disque logique. Propre à chaque partition. Parmi ces informations, on trouve :
  - ✓ la taille du FS
  - ✓ la liste des blocs libres
  - ✓ la liste des blocs utilisés
  - ✓ le nom de la partition
  - ✓ la date de modification du FS

- Liste des inodes : la table des inodes. Chacun des inodes est une structure de 64 octets. Ce nombre identifie de manière unique un fichier.

- Zone de stockage / Blocs : les blocs de données. Ils sont chaînés lors de la création du disque. Ils sont situés à la fin du disque. C'est aussi dans cette zone que l'on trouve la liste des chaînée correspondant à l'espace libre sur le disque.

La création de fichier par le SE

Un fichier ou un répertoire sont tout deux créée suivant les mêmes étapes qui sont les suivantes :

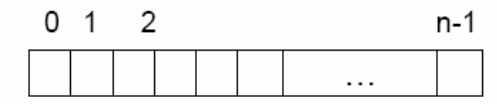
- La création d'une structure de données pour décrire le fichier. Les attributs du fichier sont sauvegardés dans cette structure de données.
- La création du fichier proprement dit. Il s'agit d'allouer au fichier un certain nombre de blocs sur le disque selon sa taille. Les blocs d'un fichier vont contenir des données sans aucune structure particulière. Les blocs d'un répertoire ont par contre une structure bien particulière, en effet ils contiennent des noms et des attributs de fichiers et de sous répertoires.

La gestion de l'espace libre sur le disque

Les systèmes d'exploitation utilisent essentiellement deux approches pour mémoriser l'espace libre : une statique et une dynamique.

Bitmap : Approche statique utilise une table de bits (vecteur de bits n blocs) comportant autant de

bits que de blocs sur le disque. A chaque bloc du disque, correspond un bit dans la table, positionné à 1 si le bloc est occupé, à 0 si le bloc est libre (ou vice versa).



Vecteur de bits à n bloc

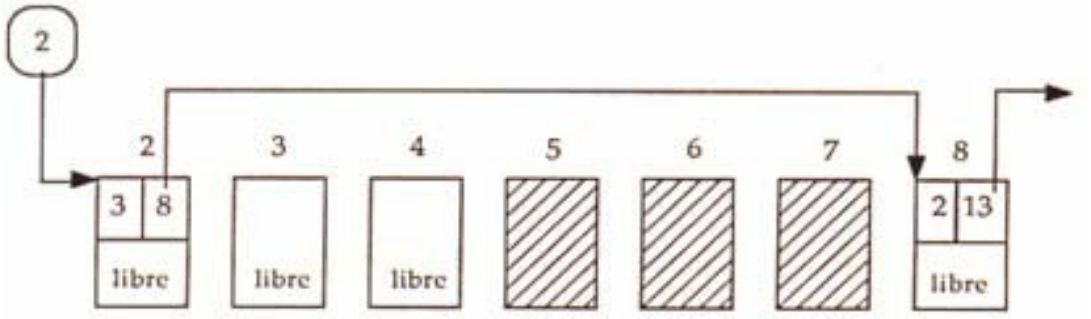
Si les blocs 3, 4, 5, 9, 10, 15, 16 sont libres : 11100011100111100…

Cette solution est utilisée pour trouver n blocs contigus, elle est utilisée dans les systèmes : NTFS, ext2fs

Exemple 3. 4

Par exemple, un disque de 300 Mo, organisé en blocs de 1 Ko, sera géré par une table de 300 Kbits qui occupera 38 des 307.200 (300x1024) blocs.

- **Liste chaînée :** Approche dynamique utilise une liste chaînée constituée d'éléments, chacun mémorisant des numéros de blocs libres. Tous les blocs libres sont liés ensemble par des pointeurs.



Exemple d'utilisation d'une liste chaînée pour la gestion de l'espace libre

Exemple 3. 5

Par exemple, un disque de 300 Mo, organisé en blocs de 1 Ko. Supposons que chaque bloc soit adressé par 4 octets. Chaque bloc de la liste pourra contenir 255 (1024/4) adresses de blocs libres. La liste comprendra donc au plus 307.200/255 = 1205 blocs. Cette solution mobilise beaucoup plus de place que la précédente.

Etude de cas : Systèmes de fichiers LINUX

Sous linux, tout est fichier, organisé suivant une unique arborescence (dont la racine est nommée / et dont l'administrateur est root)

Les différentes catégories de fichiers

- **fichiers normaux (-) :** fichiers normaux : **** texte*** : courrier, sources des programmes, scripts, configuration ; **** exécutables*** : programmes en code binaire
  La commande `ls -l` donne :

  ```
  -rwxrw-r-- 1 etudiant 2LR 34568 avril 3 14 :34 mon-fichier
  ```

- **fichiers répertoires (*d*) :** ce sont des fichiers conteneurs qui contiennent des références à d'autres fichiers. ils permettent d'organiser les fichiers par catégories
  La commande `ls -l` sur un répertoire donne

  ```
  drwxr-x--- 1 etudiant 2LR 13242 avril 2 13 :14 mon-répertoire
  ```

- **fichiers spéciaux :** situés dans `/dev`, ce sont les points d'accès préparés par le système aux périphériques. Le montage va réaliser une correspondance de ces fichiers spéciaux vers leur répertoire "point de montage". Par exemple, le fichier `/dev/hda` permet l'accès et le chargement du 1er disque IDE

- **fichiers liens symboliques (*l*)** : ce sont des fichiers qui ne contiennent qu'une référence (un pointeur) à un autre fichier. Cela permet d'utiliser un même fichier sous plusieurs noms sans avoir à le dupliquer sur le disque.
  La commande `ls -l` pour un lien donne

  ```
  lrwxrwxrwx 1 root root 14 Aug 1 01:58 Mail -> ../../bin/mail*
  ```

Remarque : Sous un système UNIX, un fichier quel que soit son type est identifié par un numéro appelé numéro d'inode (i-nœud). Ainsi derrière la façade du shell, un répertoire n'est qu'un fichier, identifié aussi par un inode, contenant une liste d'inode représentant chacun un fichier.

Pour connaître le numéro d'inode d'un fichier, on utilise la commande

```
$ls -i mon-fichier
```

3.6.1.1 Parcourir et lister les répertoires

Voici les commandes indispensables (suivies bien sûr d'une validation) pour visiter l'arborescence.

`$ls` (*commande générale d'accès aux infos des fichiers du répertoire courant* (`ls`, `ls -l`, `ls -a`))

`$cd [chemin]` (*le chemin peut être absolu ou relatif)*

`cd ..` (*remonter un niveau (vers le répertoire parent))*

`$cd` (*raccourci vers le répertoire personnel)*

`$pwd` (*donne le nom complet du répertoire courant)*

`$mkdir rep` (*pour créer un sous-repertoire du répertoire courant)*

`$rmdir rep` (*pour supprimer un sous-répertoire vide*

`$mv répertoire répertoire-d'accueil/` (*déplacement d'un répertoire)*

`$mv répertoire nouveau-nom` (*Changement de nom d'un répertoire)*

Commandes de gestion des fichiers

Pour gérer les fichiers vous disposez des commandes suivantes:

`$touch mon-fichier` (*création d'un fichier vide)*,

`$more mon-fichier` (*visualisation d'un fichier page à page)*,

`$rm mon-fichier` (*suppression d'un fichier)*,

$mv mon-fichier répertoire_accueil (déplacement d'un fichier),

$mv mon-fichier nouveau-nom(*changement de nom d'un fichier)*,

$cp nom-fichier répertoire-accueil/autre-nom(*copie de fichier)*,

$file mon-fichier (*pour savoir si on a un fichier binaire (exécutable) ou un fichier texte. On obtient pour un fichier texte, comme sortie mon-fichier : ascii text)*

Créer des liens (`ln`)

Les liens sont utiles pour faire apparaître un même fichier dans plusieurs répertoires, ou sous des noms différents. Ils évitent les duplications et assurent la cohérence des mises à jour.
On distingue en fait deux sortes de liens :

1. **Les liens durs** associent deux ou plusieurs fichiers à un même espace sur le disque, les deux fichiers restant indépendants.
   `$ln rapport.txt  /home/etudiant/rapport-lien-dur.txt`

   Le fichier `rapport-lien-dur` est créé dans le répertoire `/home/etudiant`. On peut constater que ces 2 fichiers ont la même taille. Au niveau gestion ils sont indépendants, tout en partageant le même espace disque et donc le même inode. Toute modification de l'un, modifie l'autre. Mais la suppression de l'un, casse le lien, mais ne supprime pas physiquement l'autre.

2. **Les liens symboliques**

   `$ln -s rapport.txt /home/etudiant/rapport-lien-s.txt`

   La commande `ls -F` passée dans le répertoire `/home/etudiant` montre que le fichier `rapport-lien-s.txt` pointe sur `rapport.txt` (ainsi, une requête sur `rapport-lien-s.txt`, va ouvrir `rapport.txt`)

   Le lien symbolique fait référence à un fichier dans un répertoire. La suppression du fichier source entraînera un changement de comportement du fichier lien qui sera considéré comme "cassé" ("broken").

Remarque : La différence entre un lien hard et symbolique se trouve au niveau de l'inode, un lien hard n'a pas d'inode propre, il a l'inode du fichier vers lequel il pointe. Par contre un lien symbolique possède sa propre inode.