SYSTÈME DE GESTION DES FICHIERS

SE

Madame Khaoula ElBedoui-Maktouf

2ème année Ingénieur Informatique

Plan

1. Concept de Fichier
2. Méthodes d'accès à un Fichier
3. Méthodes d'accès au contenu d'un Fichier
4. Implantation d'un SGF
5. Gestion de l'espace mémoire libre
6. Étude de cas : Linux et Windows

Concept de Fichier

❖ Problématique

Le stockage dans l'espace d'adressage souffre de :

- **Une capacité de stockage restreinte**
- **Une perte d'information après la fin du processus**
- **Un conflit d'accès par d'autres processus**

Concept de Fichier

❖ Solution

Stockage à long terme qui offre :

- **Une grande capacité de stockage**
- **Une conservation des informations après la fin du processus**
- **Un accès simultané par plusieurs processus**

➢ **Fichiers**

Concept de Fichier

❖ Définition

- Un Fichier est **l'unité logique** de stockage de l'information qui fait **abstraction** des propriétés physiques des périphériques de stockage

Concept de Fichier

❖ Définition

- Un Fichier est l'unité logique de stockage de l'information qui fait abstraction des propriétés physiques des périphériques de stockage
- Le Fichier est physiquement stocké sur un support de mémoire de masse (permanent)

Concept de Fichier

❖ Définition

- Un Fichier est l'unité logique de stockage de l'information qui fait abstraction des propriétés physiques des périphériques de stockage
- Le Fichier est physiquement stocké sur un support de mémoire de masse (permanent)
- Le SE gère les Fichiers via le Système de Gestion des Fichiers (file system): nommage, utilisation, accès, protection et implantation
- Un SGF fournit le mécanisme de stockage et d'accès aux données

Concept de Fichier

❖ Types

Un Fichier peut être :

- **Fichier régulier :** texte, binaire (exécutable),
- **Fichier spécial:** répertoire, périphériques, …

Concept de Fichier

❖ Types

Un Fichier peut être :

- **Fichier régulier :** texte, binaire (exécutable),
- **Fichier spécial:** répertoire, périphériques, …

Sous Unix la commande file permet de visualiser le type d'un Fichier :

-, d, s, c, b,

Concept de Fichier

❖ Attributs

Un Fichier a un ensemble de caractéristiques (attributs):

- **la taille (octets)**
- **les dates : création, dernière modification, dernier accès**
- **le propriétaire**
- **les droits d'accès**
- **le type**
- **…**

Concept de Fichier

❖ Attributs

Un Fichier a un ensemble de caractéristiques (attributs):

- **la taille (octets)**
- **les dates : création, dernière modification, dernier accès**
- **le propriétaire**
- **les droits d'accès**
- **le type**
- **…**

Certains de ces attributs sont fournis par l'utilisateur, d'autres sont compétés par le SE

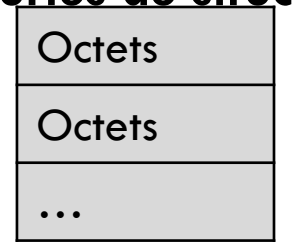
Sous Unix pour afficher tous les attributs, on utilise la commande ls -l

Concept de Fichier

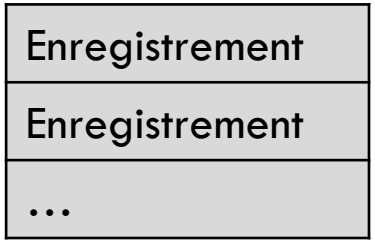
❖ Structure interne

Le Fichier peut avoir l'une des trois sortes de structures :

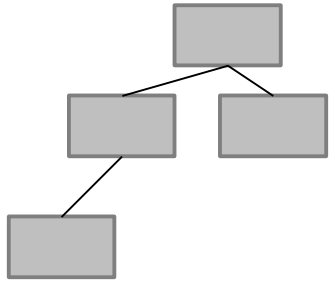
- **Séquence d'octets non structurés:**



- **Séquence d'enregistrements:**
  - **Enregistrements sont de longueur fixe**
  - **Opérations manipulent des enregistrements**



- **Arborescence d'enregistrements:**
  - **Enregistrements ne sont pas tous de même longueur**
  - **Clé dont la position est fixe**
  - **Arbre trié en fonction des clés (recherche rapide)**



Concept de Fichier

❖ Cas de Répertoire

Le répertoire forme un moyen d'organisation des Fichiers.

Concept de Fichier

❖ Cas de Répertoire

Le répertoire forme un moyen d'organisation des Fichiers.

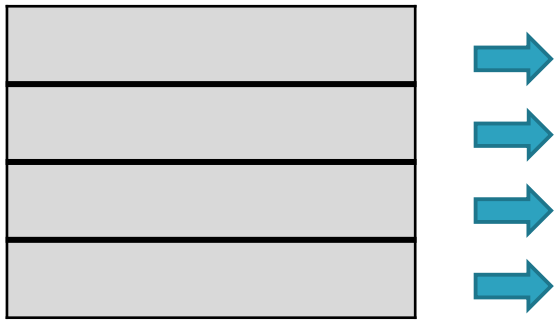
De point de vue S.D, un répertoire est un tableau :

Concept de Fichier

❖ Cas de Répertoire

Le répertoire forme un moyen d'organisation des Fichiers.

De point de vue S.D, un répertoire est un tableau :

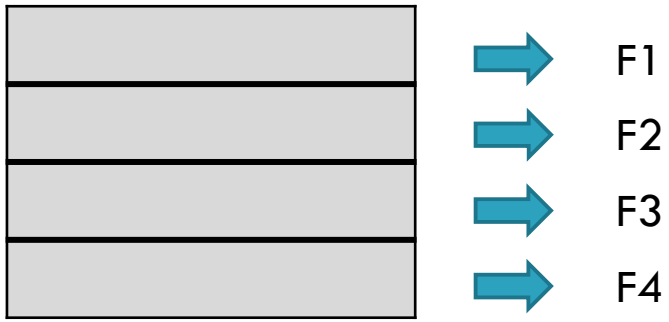


Concept de Fichier

❖ Cas de Répertoire

Le répertoire forme un moyen d'organisation des Fichiers.

De point de vue S.D, un répertoire est un tableau :



Concept de Fichier

❖ Cas de Répertoire

Structure

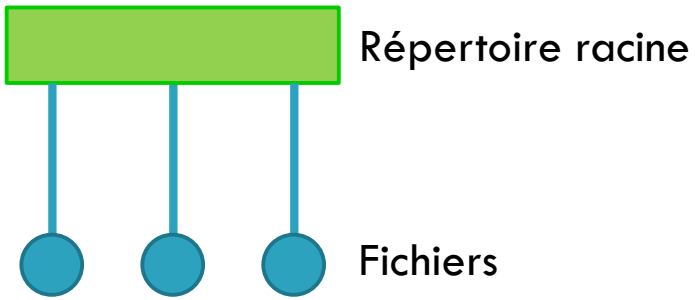
Nous distinguons 3 structures logiques:

- **Systèmes à un niveau de répertoire**
- **Systèmes à deux niveaux de répertoire**
- **Systèmes à répertoires hiérarchiques**

Concept de Fichier

❖ Cas de Répertoire

Structure plate ou à un niveau :

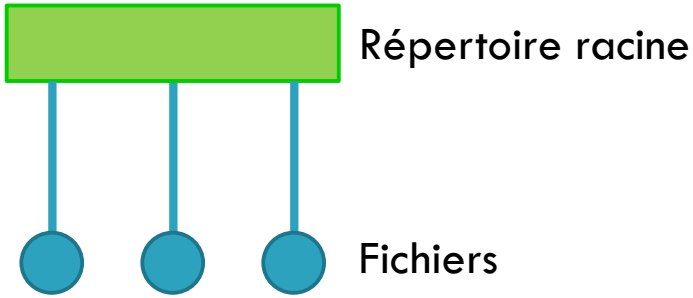


- **Tous les fichiers appartiennent au même répertoire**
- **Avantages:**
  - **Facilité de gestion et d'interprétation**

Concept de Fichier

❖ Cas de Répertoire

Structure plate ou à un niveau :



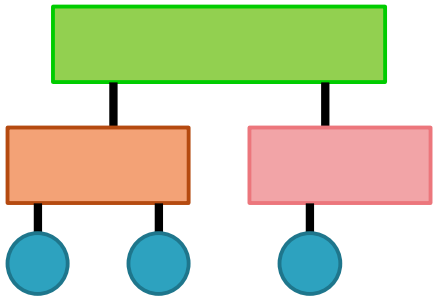
- **Inconvénients:**
  - **Problème de nommage: noms uniques**
  - **Difficile à exploiter dans un système multi-utilisateurs**

Application : Système embarqué

Concept de Fichier

❖ Cas de Répertoire

Structure à deux niveaux :

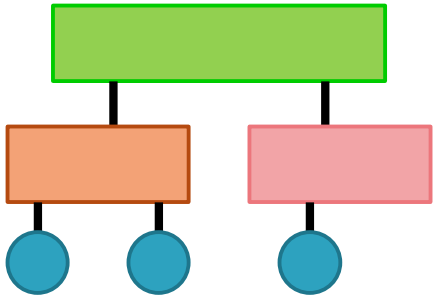


- Un **répertoire de fichiers maître** contient des comptes utilisateur
- Chaque utilisateur possède son propre **répertoire de fichiers utilisateur**

Concept de Fichier

❖ Cas de Répertoire

Structure à deux niveaux :



MFD : Master File Directory

UFD : User File Directory

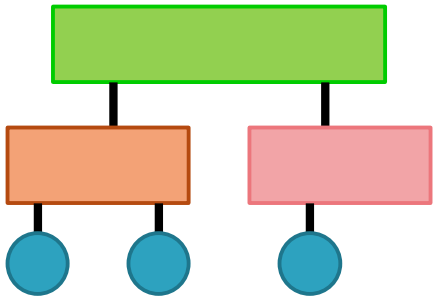
Fichiers

- **Avantages**
  - **Résout le problème de collision des noms entre différents utilisateurs**

Concept de Fichier

❖ Cas de Répertoire

Structure à deux niveaux :



MFD : Master File Directory

UFD : User File Directory

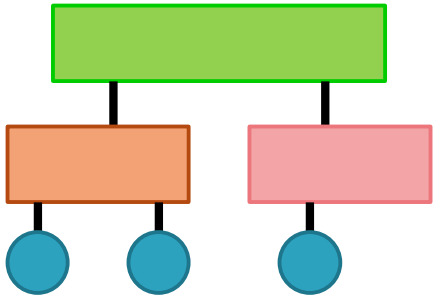
Fichiers

- **Inconvénients**
  - **Collaboration des utilisateurs**
  - **Recherche des Fichiers (Chemin de recherche)**

Concept de Fichier

❖ Cas de Répertoire

Structure à deux niveaux :



MFD : Master File Directory

UFD : User File Directory

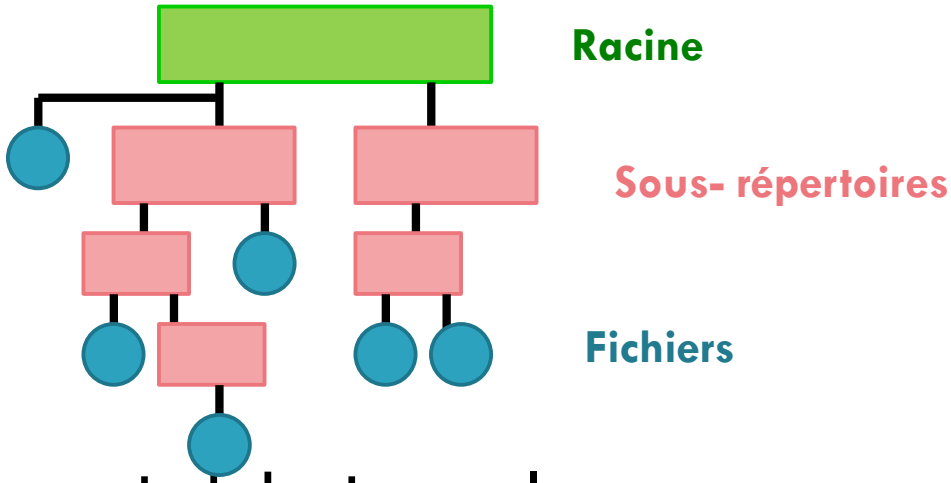
Fichiers

Problème : Nécessité d'authentification (login+ password)

Concept de Fichier

❖ Cas de Répertoire

Structure multi-niveaux ou arborescente :

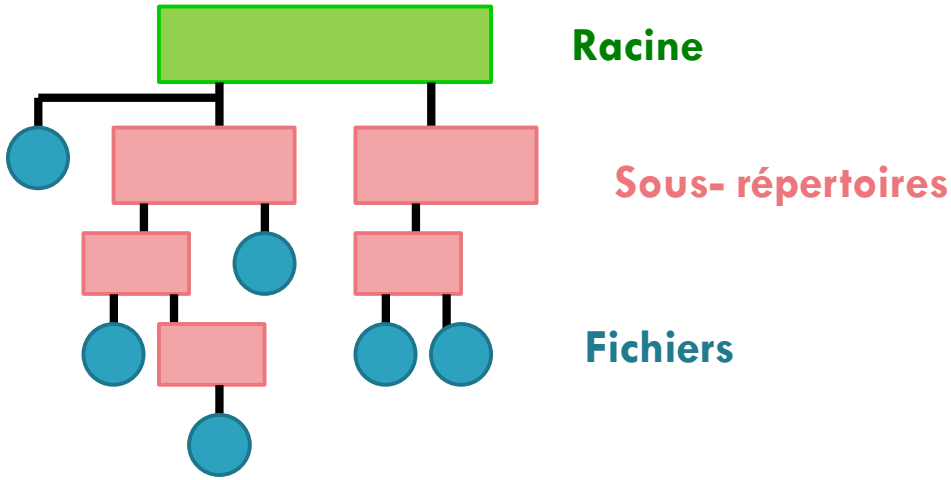


- **Structure arborescente de hauteur quelconque**
- **Répertoire racine, sous-répertoires, fichiers**

Concept de Fichier

❖ Cas de Répertoire

Structure multi-niveaux ou arborescente :

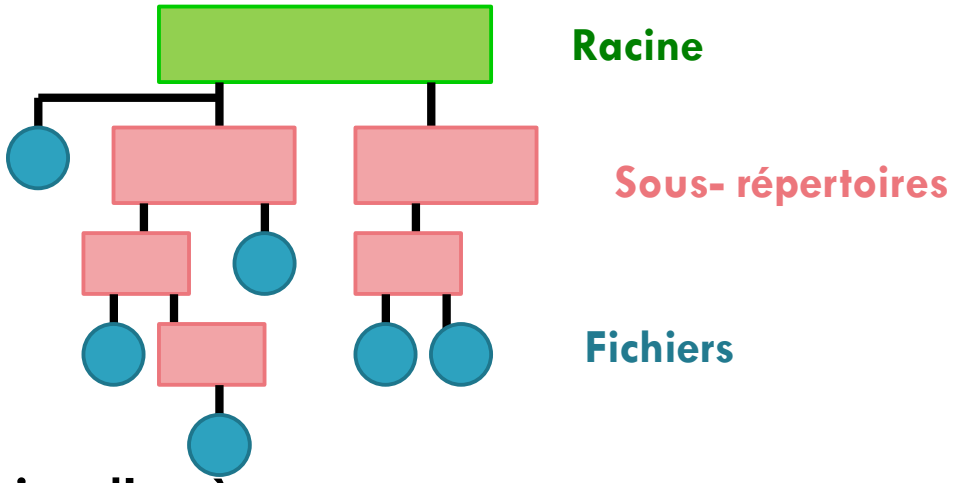


Utilisation des chemins d'accès: suite des répertoires à traverser, depuis la racine pour accéder au fichier, séparés par un caractère spécial (séparateur).

Concept de Fichier

❖ Cas de Répertoire

Structure multi-niveaux ou arborescente :

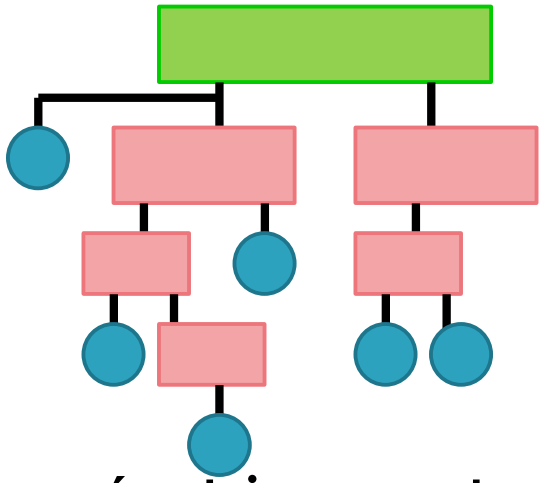


- **Deux types de chemins d'accès :**
  - **Chemin d'accès absolu: chemin de la racine jusqu'au fichier**
  - **Chemin d'accès relatif: à partir de répertoire de travail**

Concept de Fichier

❖ Cas de Répertoire

Structure multi-niveaux ou arborescente :

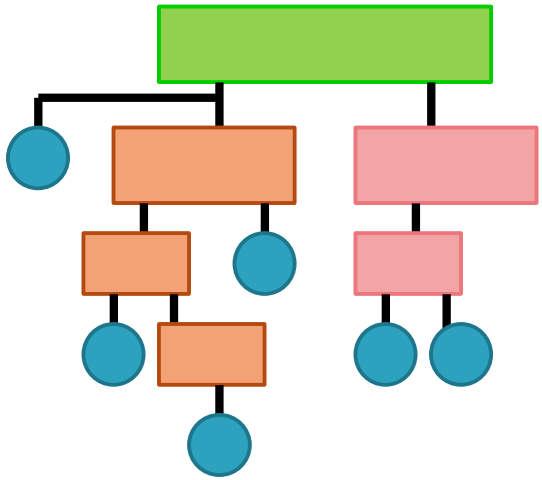


- "." fait référence au répertoire courant
- ".." fait référence au répertoire parent

Concept de Fichier

❖ Cas de Répertoire

Structure multi-niveaux ou arborescente :



Problème : Accès peut être complexe

Concept de Fichier

❖ Cas de Répertoire

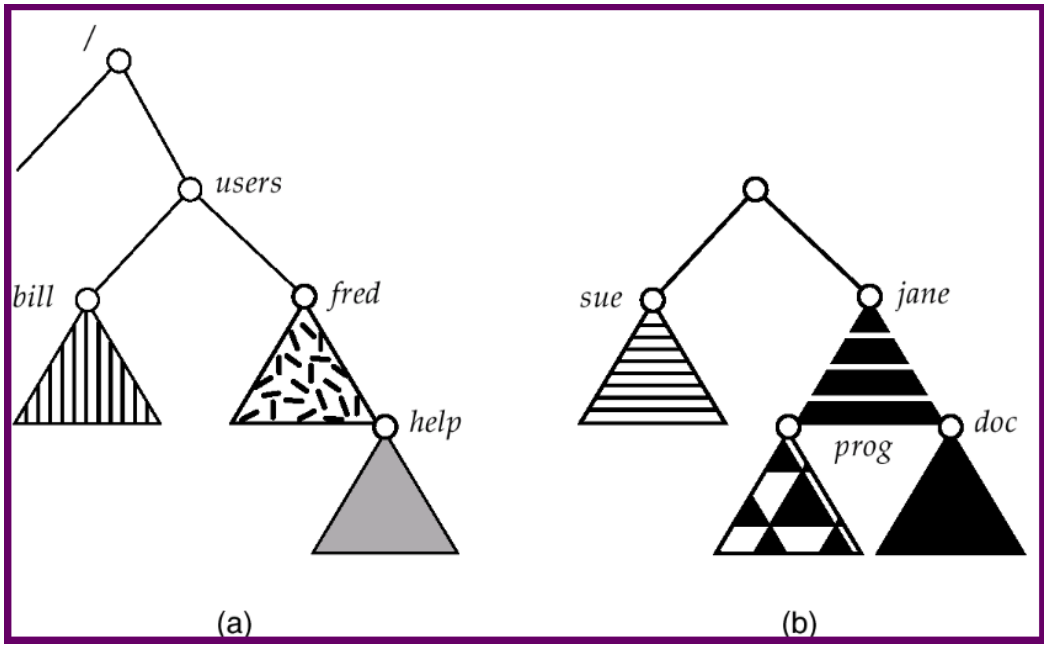
Montage

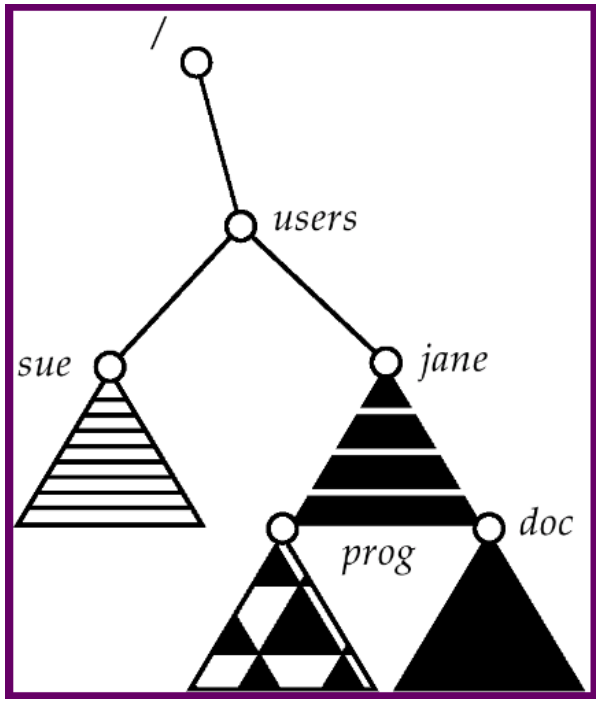
- **Définition: Intégration** d'un système de fichiers spécial (CD, clé USB, partition, second disque dur) à l'arborescence existante.
- **Objectif**: permettre et faciliter l'accès aux données qui se trouve dans le système "**monté**"
- **Point de montage**: est un répertoire à partir duquel sont accessibles les données se trouvant dans le système de fichiers qui a été intégré
- **Sémantique**:
  - Le montage se fait sur des répertoires vides
  - Si le point de montage contient déjà des fichiers, ces derniers deviennent inaccessibles jusqu'au **démontage**

Concept de Fichier

❖ Cas de Répertoire

Montage





Avant montage

Après montage

Méthodes d'accès à un Fichier

❖ Accès Séquentiel

Si l'arrivée à un Fichier oblige le parcourt des Fichiers précédents.

Ceci est fortement lié au support de stockage.

Exp. Bandes magnétiques

Méthodes d'accès à un Fichier

❖ Accès Direct

Si l'arrivée à un Fichier peut être sans nécessité de parcourt des Fichiers précédents. Ceci est aussi fortement lié au support de stockage.

Exp. CD, Disque, Flash,…

Méthodes d'accès au contenu d'un Fichier

❖ Accès Séquentiel

Il faut parcourir le contenu du Fichier donnée par donnée (à partir du début).

Exp. Compilateur, imprimante

Méthodes d'accès au contenu d'un Fichier

❖ Accès Direct

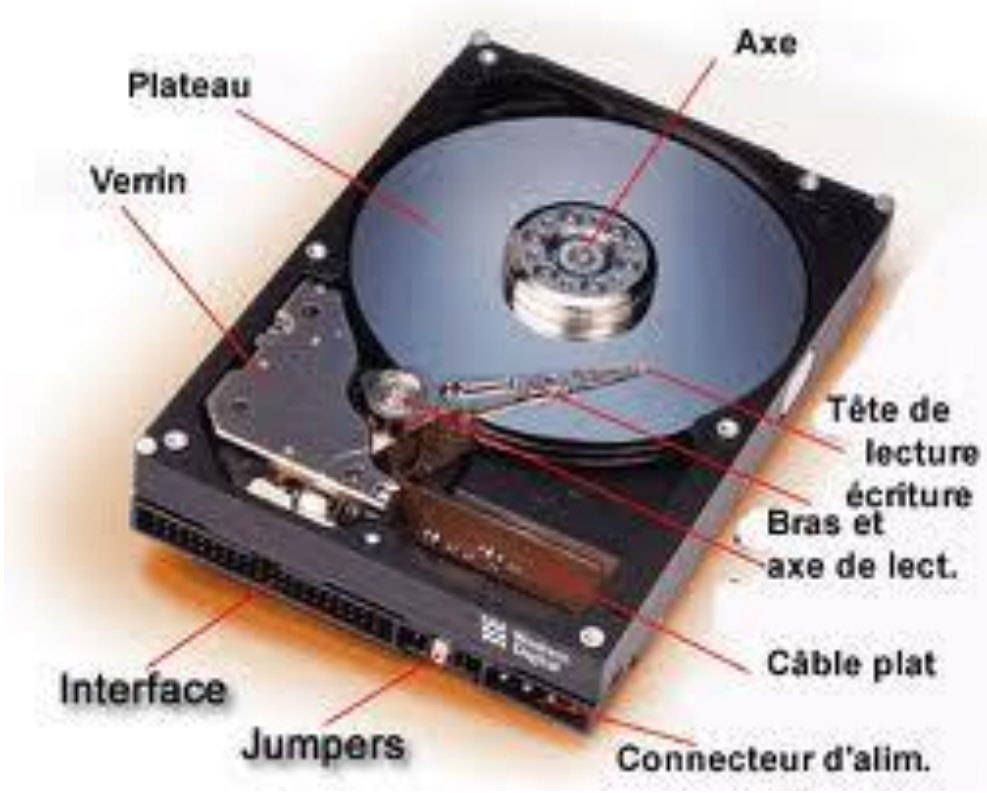
On peut accéder à une donnée directement en connaissant son numéro, sa clé, …

Exp. Base des données

Implantation d'un SGF

❖ Organisation

le disque dur

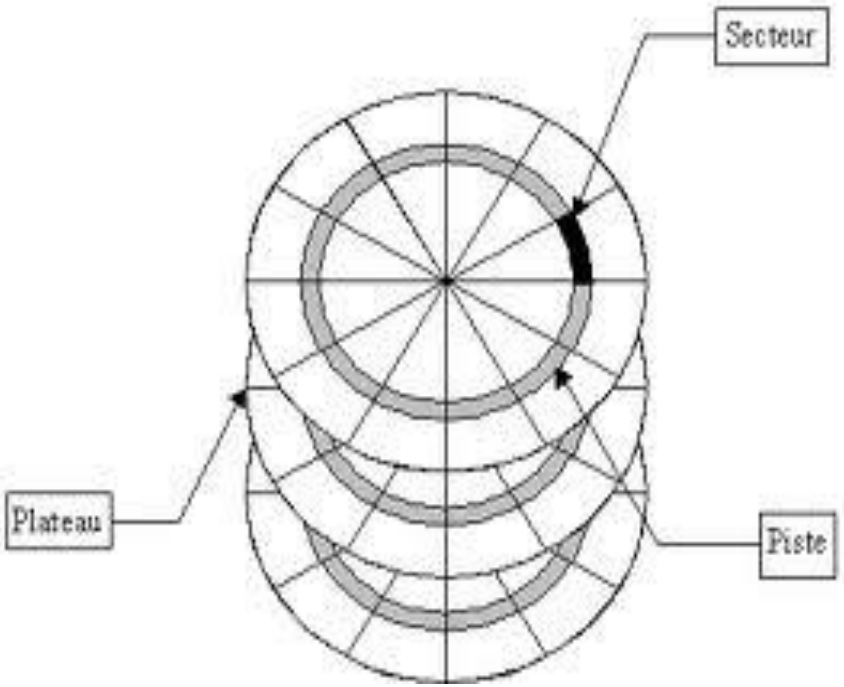


Implantation d'un SGF

❖ Organisation

le disque dur

Vue Physique



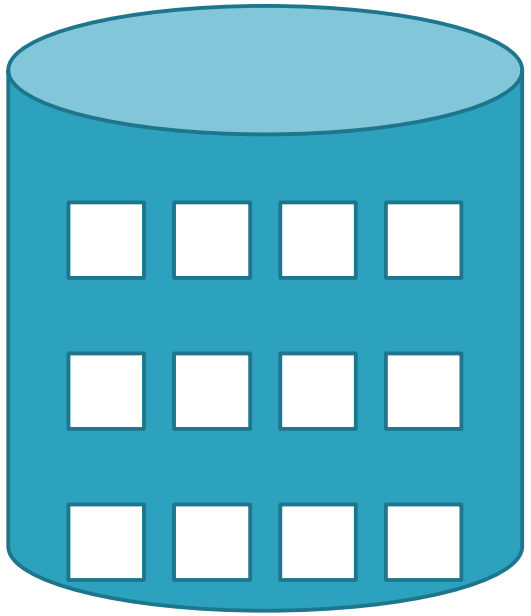
Implantation d'un SGF

❖ Organisation

le disque dur

Vue Logique

Le disque dur est un ensemble de blocs



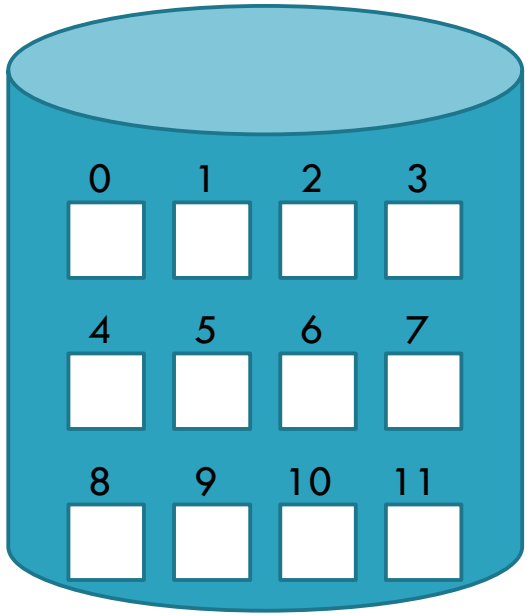
Implantation d'un SGF

❖ Organisation

le disque dur

Vue Logique

Les blocs sont numérotés



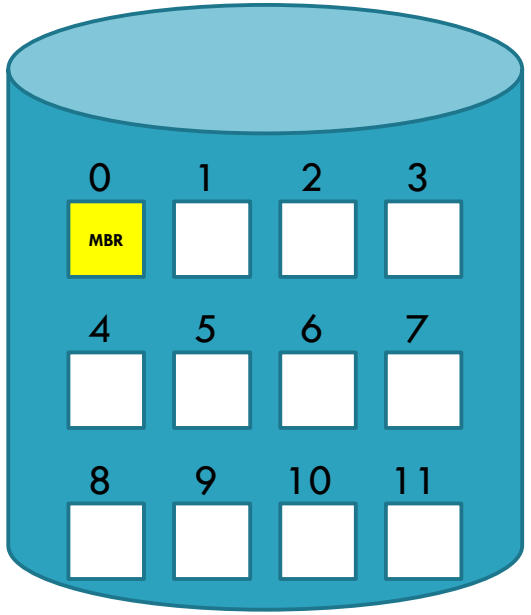
Implantation d'un SGF

❖ Organisation

le disque dur

Vue Logique

Le bloc 0 du disque dur contient
le MBR (Master Boot Record)
qui sert à booter la machine et charger le SE



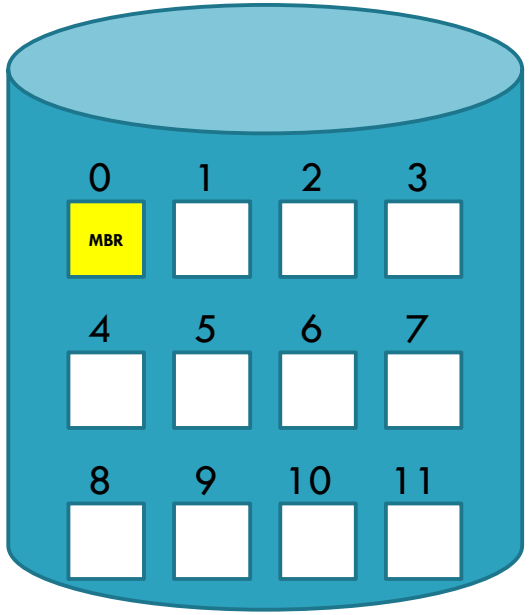
Implantation d'un SGF

❖ Organisation

le disque dur

Vue Logique

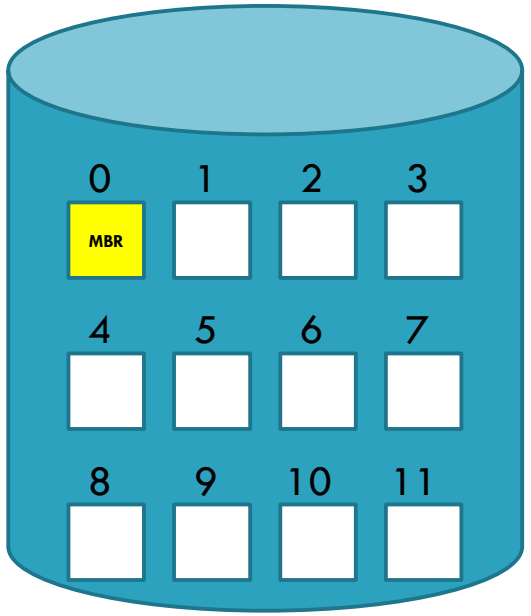
Chaque Fichier est stocké sur un ensemble de blocs.



Implantation d'un SGF

❖ Organisation

le disque dur



Vue Logique

Chaque Fichier est stocké sur un ensemble de blocs.

Comment choisir ces blocs ?

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Principe :

Le Fichier est stocké sur un ensemble adjacents de blocs

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Mise en œuvre :

L'entrée à un Fichier est structurée de la façon suivante :

Nom Fichier	Attributs	Adresse de début	Longueur (nombre de blocs)

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Exemple :

F1	…	2	2
F2	…	5	3

Répertoire A

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Exemple :

Nom	Att	début	Long
F1	…	2	2
F2	…	5	3

Répertoire A

Implantation d'un SGF

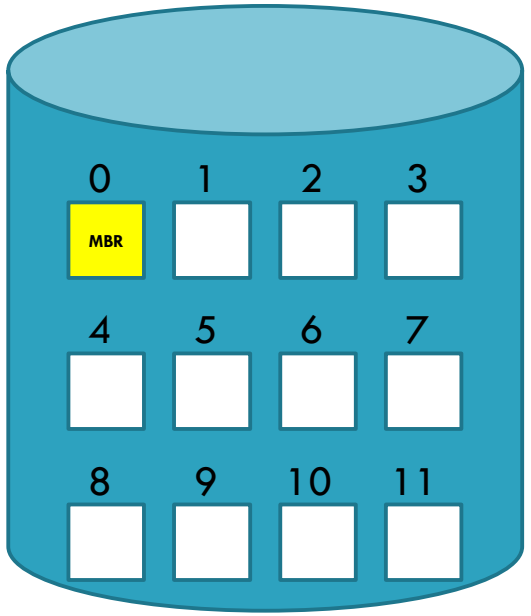
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Exemple :

Nom	Att	début	Long
F1	…	2	2
F2	…	5	3

Répertoire A



Implantation d'un SGF

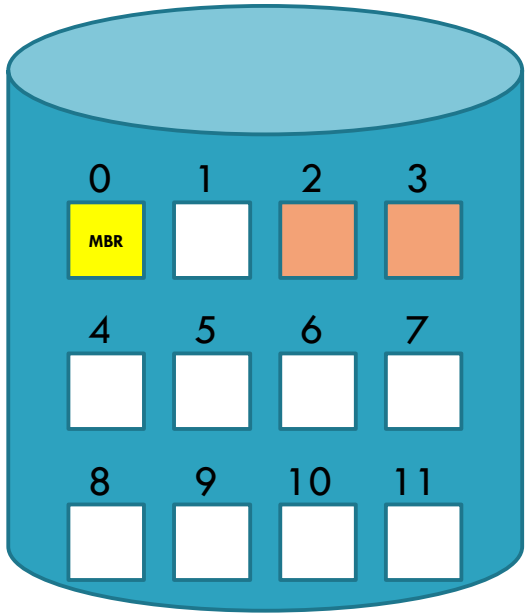
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Exemple :

Nom	Att	début	Long
F1	…	2	2
F2	…	5	3

Répertoire A



Implantation d'un SGF

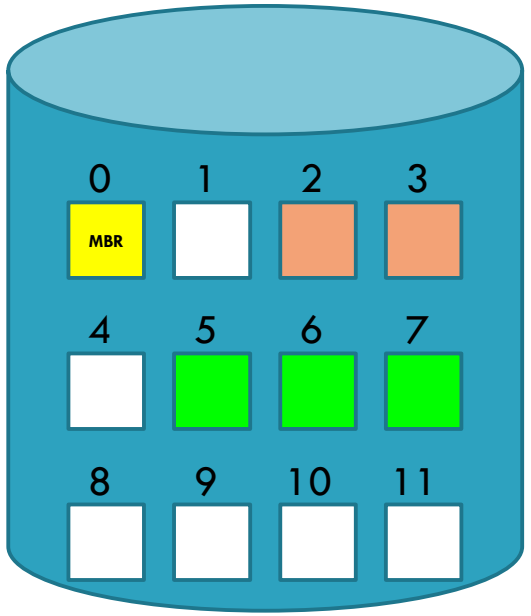
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Exemple :

Nom	Att	début	Long
F1	…	2	2
F2	…	5	3

Répertoire A



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Avantages

+ **Simple**

+ **Accès direct et accès séquentiel facile**

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Inconvénients

– **Nécessité de connaitre en avance la taille de Fichier !!!**

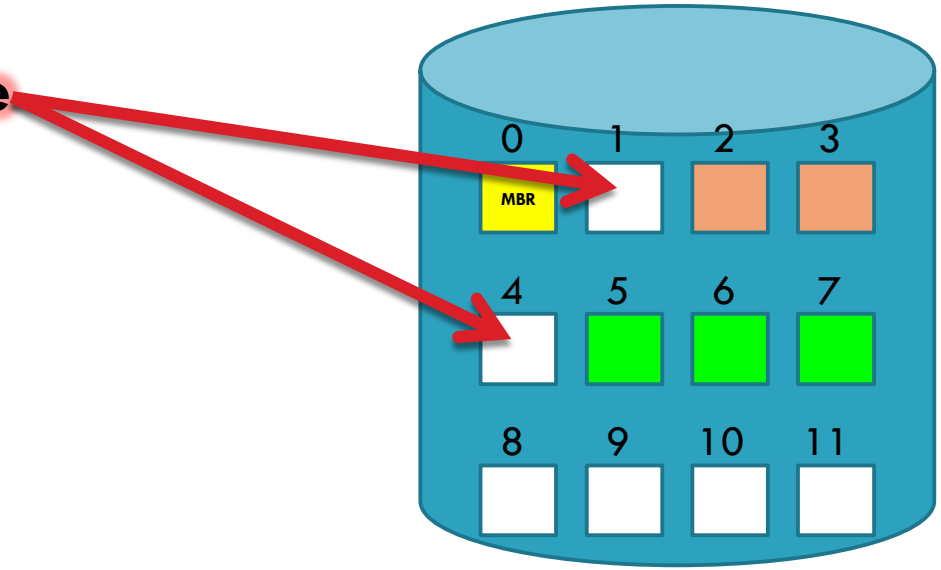
Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Inconvénients

- **Nécessité de connaitre en avance la taille de Fichier !!!**
- **Fragmentation externe**



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Inconvénients

- **Nécessité de connaitre en avance la taille de Fichier !!!**
- **Fragmentation externe**
- **Fragmentation interne**

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Inconvénients

- **Nécessité de connaitre en avance la taille de Fichier !!!**
- **Fragmentation externe**
- **Fragmentation interne**

Exemple : F1 de taille 900 Ø

Taille bloc : 512 Ø

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

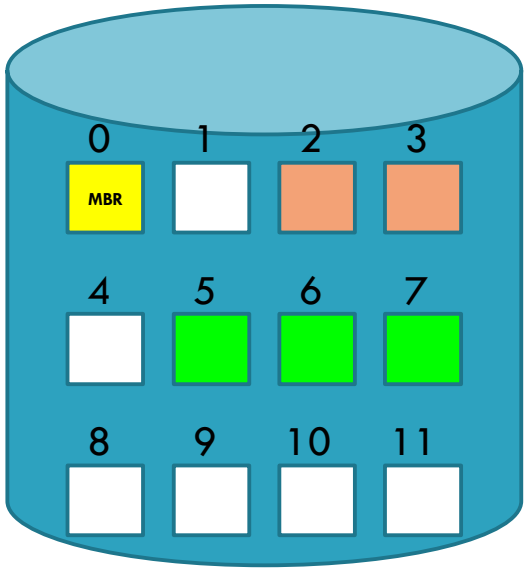
❁ Allocation Contiguë

Inconvénients

- **Nécessité de connaitre en avance la taille de Fichier !!!**
- **Fragmentation externe**
- **Fragmentation interne**

F1 de taille 900 Ø

Taille bloc : 512 Ø



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

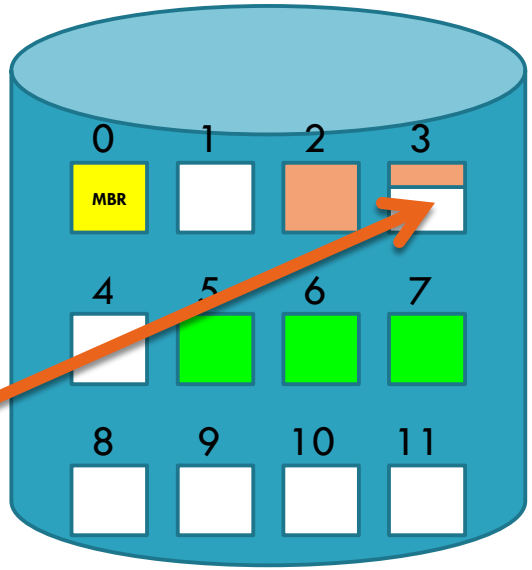
Inconvénients

- **Nécessité de connaitre en avance la taille de Fichier !!!**
- **Fragmentation externe**
- **Fragmentation interne**

F1 de taille 900 Ø

Taille bloc : 512 Ø

Fragmentation interne = 2*512 – 900 = 124 Ø



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Remarques :

- **Nécessité de connaitre en avance la taille de Fichier !!!**

Cette méthode est utilisée pour la gravure.

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Contiguë

Remarques :

- **Fragmentation externe**

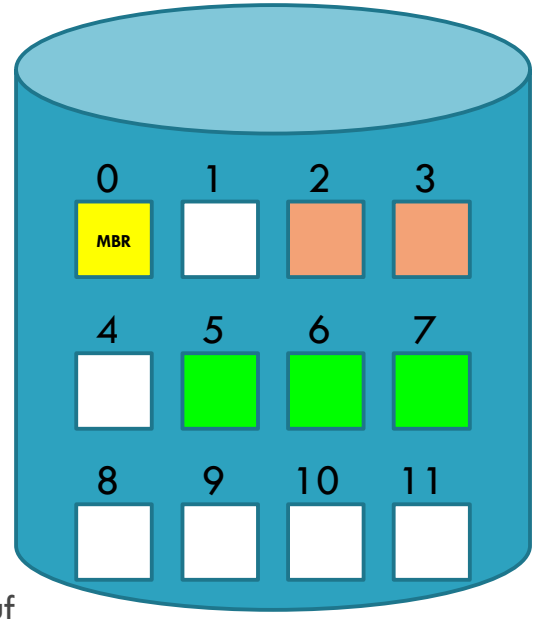
La solution est le compactage (défragmentation)

Cette technique est coûteuse en terme de temps et nécessite la mise à jour des adresses.

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

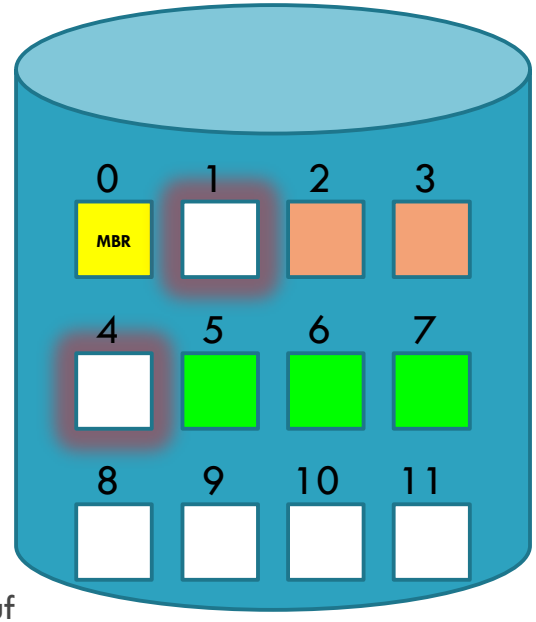
Le compactage (défragmentation)



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

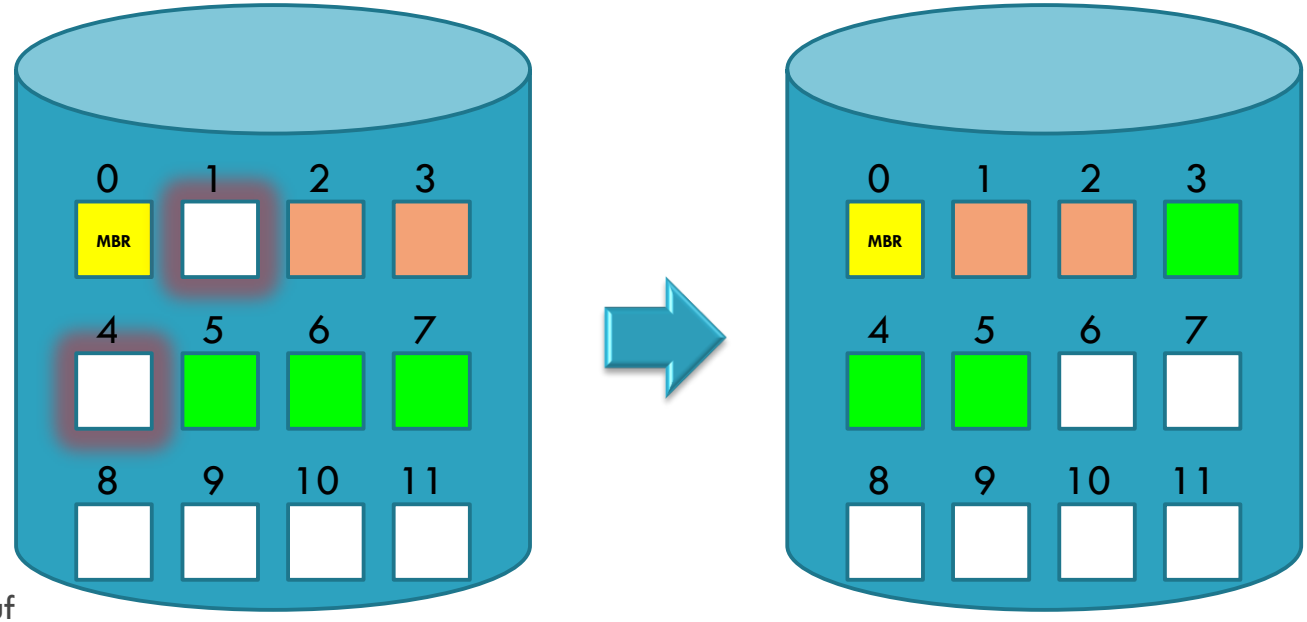
Le compactage (défragmentation)



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Contiguë**

Le compactage (défragmentation)



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Principe :

Le Fichier est stocké sur un ensemble de blocs liés enter eux par des pointeurs.

Chaque bloc se termine par un pointeur qui indique l'adresse du bloc suivant.

Implantation d'un SGF

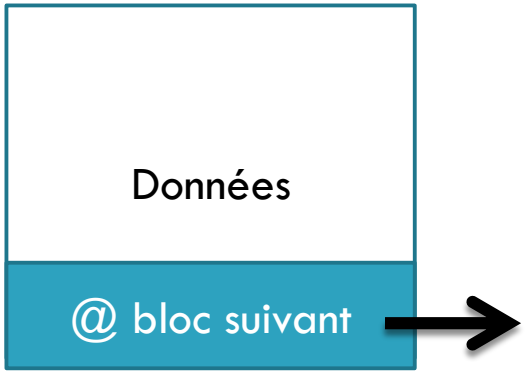
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Principe :

Le Fichier est stocké sur un ensemble de blocs liés enter eux par des pointeurs.

Chaque bloc se termine par un pointeur qui indique l'adresse du bloc suivant.



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Mise en œuvre :

L'entrée à un Fichier est structurée de la façon suivante :

Nom Fichier	Attributs	Adresse de début

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Chaînée**

Exemple :

F1	…	2
F2	…	5

Répertoire A

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A

Implantation d'un SGF

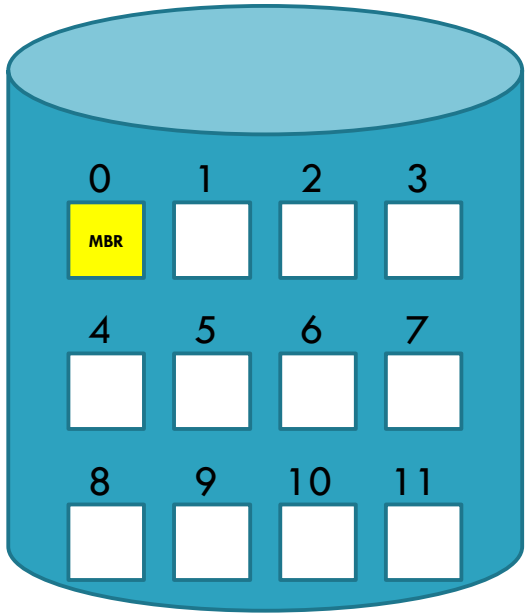
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

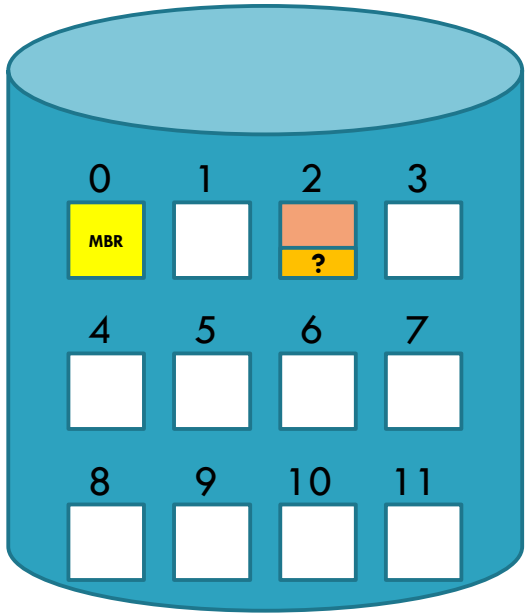
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

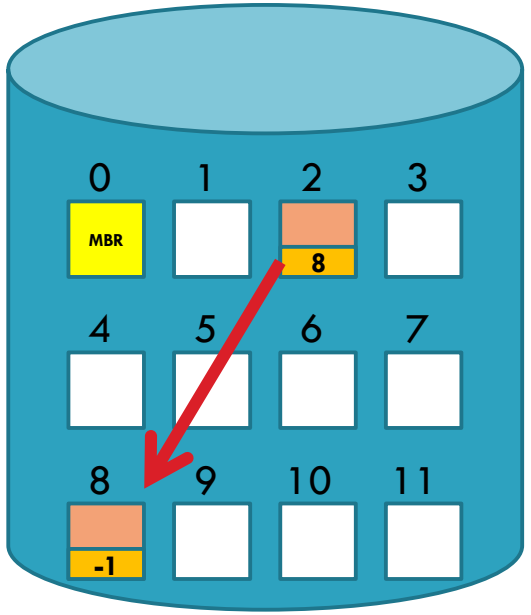
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

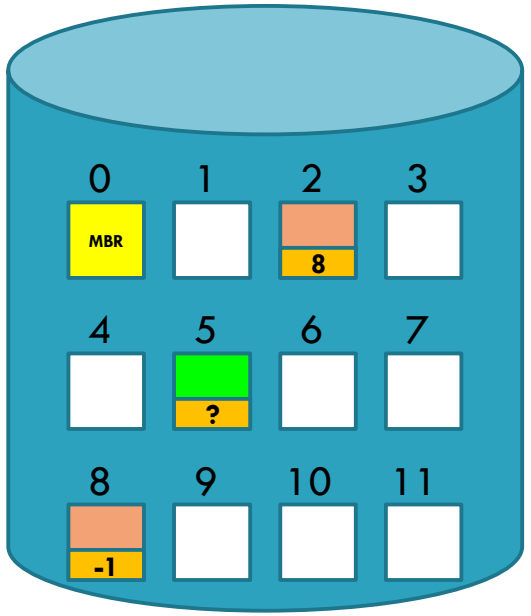
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

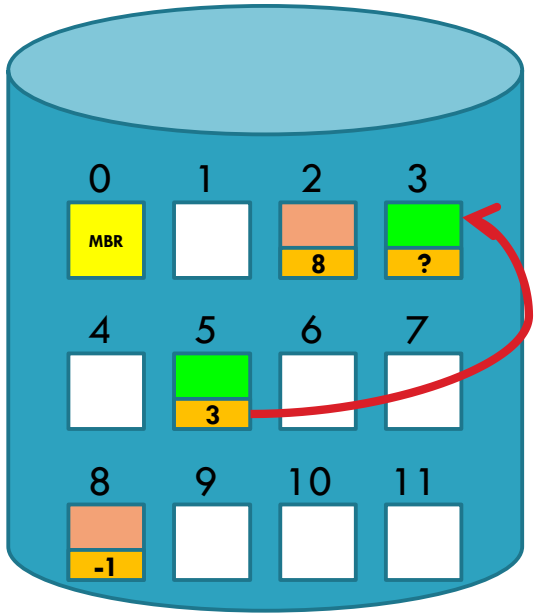
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

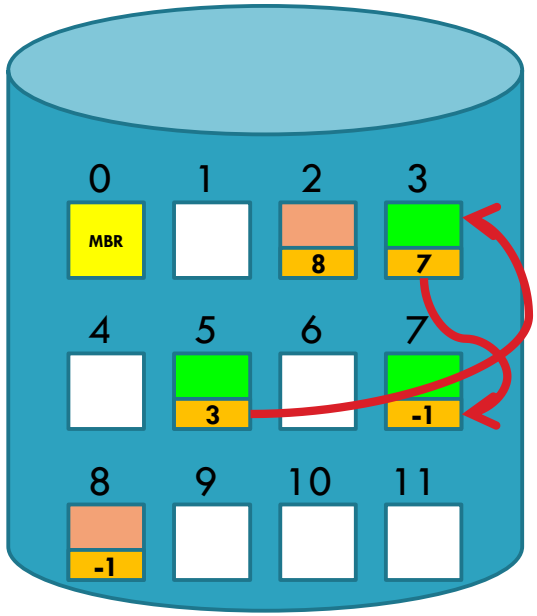
❖ Méthodes d'allocation de blocs de disque

❁ Allocation Chaînée

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A



Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Chaînée**

Avantages

+ Pas de fragmentation externe

+ Pas de problème de taille de Fichier (il n'est pas nécessaire de connaître la taille de Fichier en avance).

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Chaînée**

Inconvénients

- **Fragmentation interne**
- **Accès direct impossible**
- **Accès séquentiel lent**
- **La perte d'un pointeur engendre la perte du Fichier**

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation Chaînée**

Inconvénients

- Fragmentation interne
- Accès direct impossible
- Accès séquentiel lent
- **La perte d'un pointeur engendre la perte du Fichier**

Mélange entre donnée utilisateur et donnée système

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation chaînée avec table FAT

Principe :

Le chaînage entre les blocs d'un fichier est stocké dans une table et non pas au niveau du bloc

La table est dite FAT (File Allocation Table).

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation chaînée avec table FAT

Principe :

La table FAT

Elle est indexée par le numéro de bloc et indique pour chaque bloc l'adresse du bloc suivant.

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation chaînée avec table FAT

Principe :

La table FAT

N° bloc	@ du bloc suivant
0	
1	
2	
…	

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation chaînée avec table FAT**

Mise en œuvre :

L'entrée à un Fichier est structurée de la façon suivante :

Nom Fichier	Attributs	Adresse de début

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

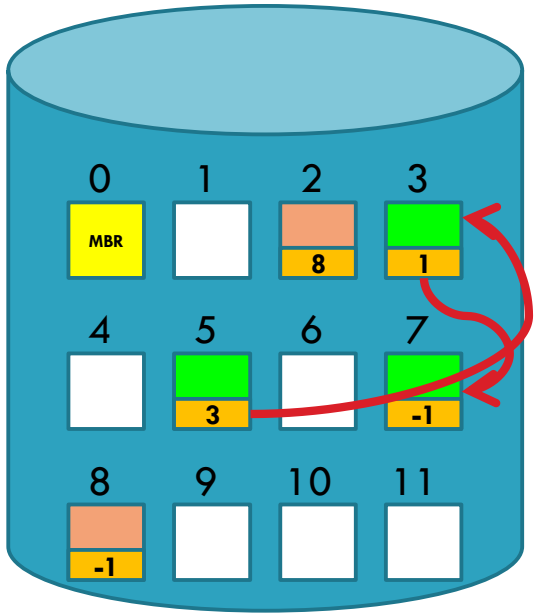
❁ Allocation chaînée avec table FAT

Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A

En cas de FAT ?



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation chaînée avec table FAT

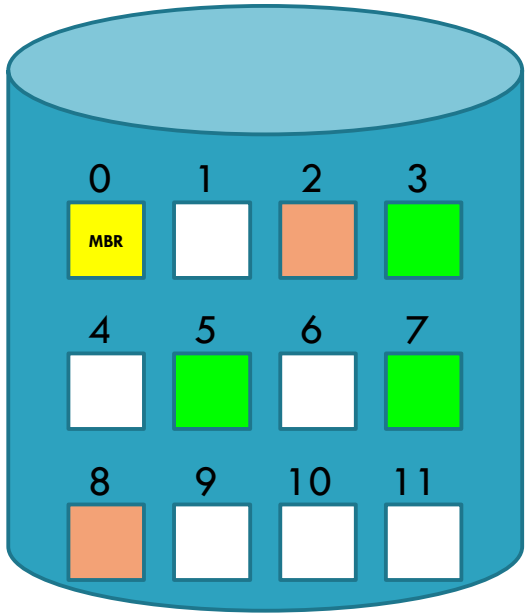
Exemple :

Nom	Att	début
F1	…	2
F2	…	5

Répertoire A

	@ bloc suivant
0	-1
1	
2	8
3	7
4	
5	3
6	
7	-1
8	-1
9	
10	
11	

FAT



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation chaînée avec table FAT

Avantages

+ Pas de fragmentation externe

+ Pas de problème de taille de Fichier (il n'est pas nécessaire de connaître la taille de Fichier en avance).

+ pas de mélange entre donnée utilisateur et donnée système

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation chaînée avec table FAT**

Inconvénients

- **Fragmentation interne**
- **Retour régulier à la table FAT**
- **Accès séquentiel et direct lent**
- **Consommation de l'espace mémoire pour le stockage de FAT**

(et ceci quelque soit le nombre de Fichiers)

Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Principe :

Le chaînage entre les blocs d'un fichier est stocké dans un nœud d'index

(index node : i-node)

Chaque Fichier a son propre i-node

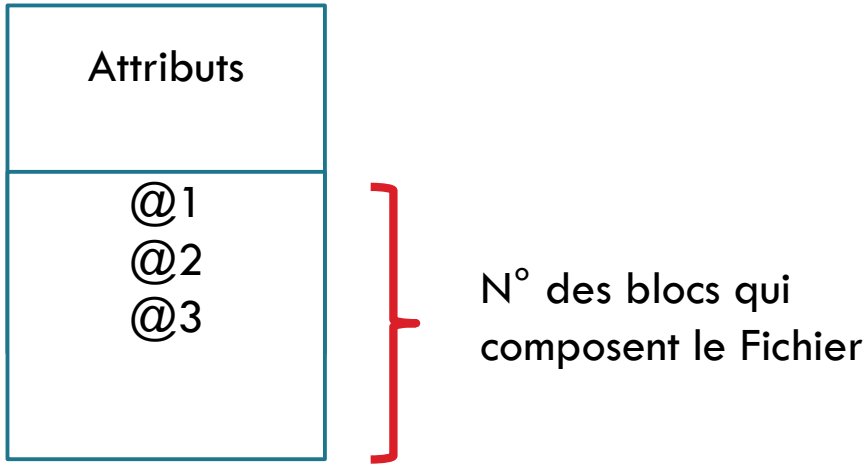
Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Principe :

L'i-node



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Mise en œuvre :

L'entrée à un Fichier est structurée de la façon suivante :

Nom Fichier	Adresse de l'i-node

Implantation d'un SGF

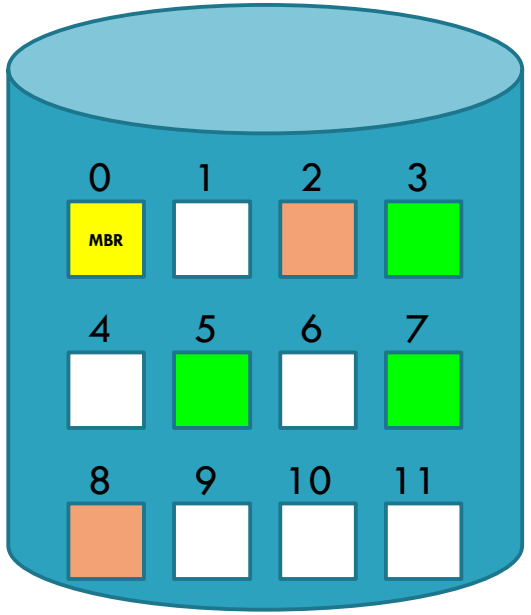
❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Exemple :

Nom	i-node
F1	4
F2	8

Répertoire A



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Exemple :

Nom	i-node
F1	4
F2	8

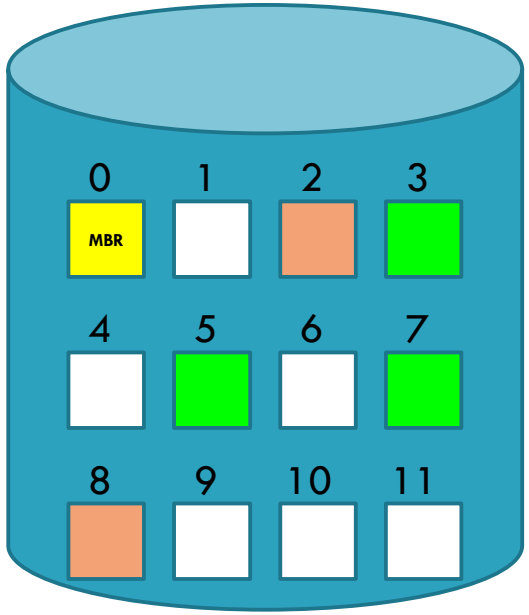
Répertoire A

i-node 4

Att F1
2 8

i-node 8

Att F2
5 3 7



Implantation d'un SGF

❖ Méthodes d'allocation de blocs de disque

❁ Allocation indexée

Exemple :

Nom	i-node
F1	4
F2	8

Répertoire A

i-node 4

Att F1
2 8

i-node 8

Att F2
5 3 7

Sous Unix, pour connaître le numéro d'i-node d'un Fichier, on utilise la commande ls -i

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation indexée**

Avantages

+ **Pas de fragmentation externe**

+ **Pas de problème de taille de Fichier (il n'est pas nécessaire de connaître la taille de Fichier en avance).**

+ **pas de mélange entre donnée utilisateur et donnée système**

Implantation d'un SGF

- **Méthodes d'allocation de blocs de disque**
  - **Allocation indexée**

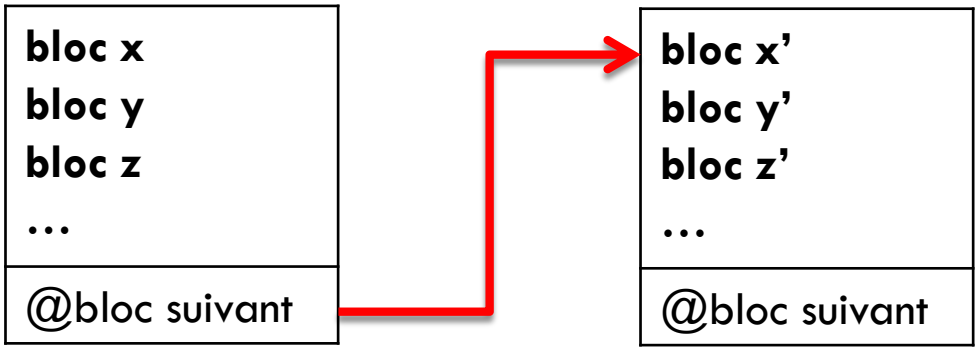
Inconvénients

- **Fragmentation interne (au niveau de l'i-node et du Fichier)**
- **Retour régulier à l'i-node**
- **Accès séquentiel et direct lent**
- **Consommation de l'espace mémoire pour le stockage des i-nodes**

Gestion de l'espace mémoire Libre

❖ Méthode dynamique

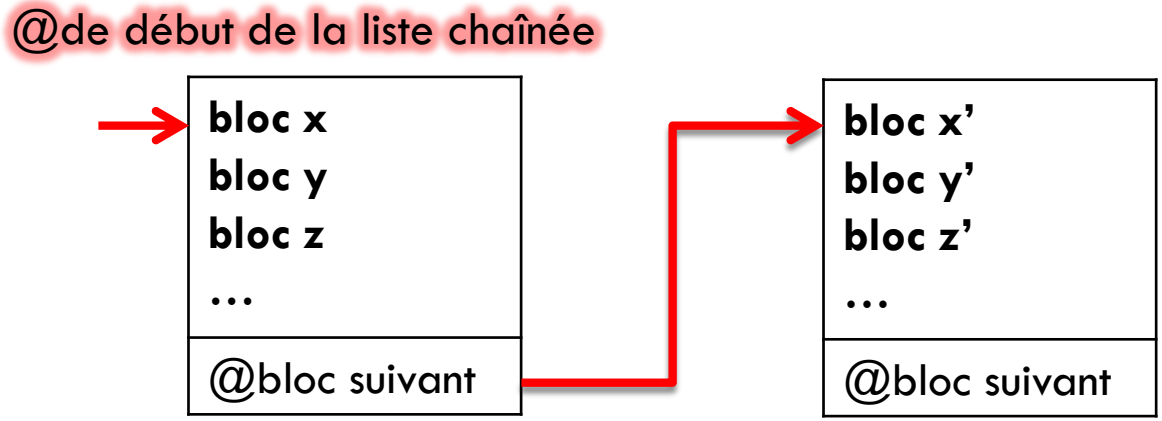
Se base sur l'utilisation d'une liste chaînée de blocs spéciaux qui contiennent les numéros des blocs libres.



Gestion de l'espace mémoire Libre

❖ Méthode dynamique

Se base sur l'utilisation d'une liste chaînée de blocs spéciaux qui contiennent les numéros des blocs libres.



Gestion de l'espace mémoire Libre

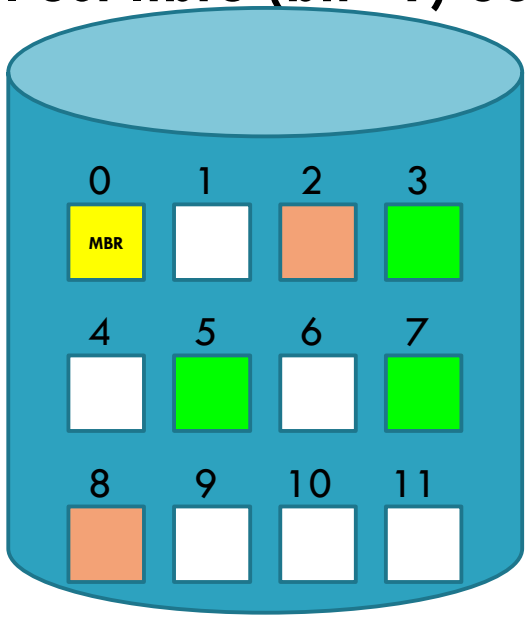
❖ Méthode Statique

Se base sur l'utilisation d'une table de bits où chaque bits fait référence à un bloc et indique s'il est libre (bit=1) ou non (bit=0).

Gestion de l'espace mémoire Libre

❖ Méthode Statique

Se base sur l'utilisation d'une table de bits où chaque bits fait référence à un bloc et indique s'il est libre (bit=1) ou non (bit=0).

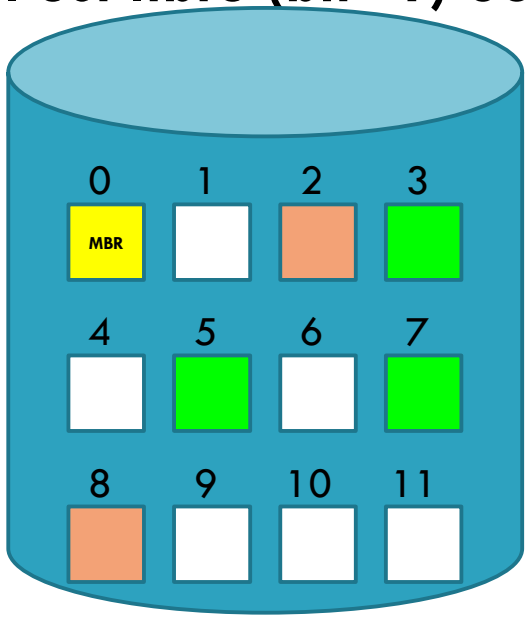


Gestion de l'espace mémoire Libre

❖ Méthode Statique

Se base sur l'utilisation d'une table de bits où chaque bits fait référence à un bloc et indique s'il est libre (bit=1) ou non (bit=0).

N°Bloc	0	1	2	3	4	5	6	7	8	9	10	11
Etat	0	1	0	0	1	0	1	0	0	1	1	1



Cas de Linux

❖ Organisation

Disque dur

Cas de Linux

❖ Organisation

MBR **(sur le bloc 0) : le programme qui permet de démarrer la machine et de lancer le SE**

Cas de Linux

❖ Organisation

Super Bloc **(sur le bloc 1) : contient des informations nécessaires:**

- **@ de début de la liste des blocs libres**
- **Nombre de blocs libres**
- **Taille de la table d'i-nodes**
- **Nombre de blocs de disque**
- **…**

Cas de Linux

❖ Organisation

Table d'i-nodes : contient les i-nodes numérotés de 1 à max i-node

Chaque i-node décrit un Fichier unique.

- **L'i-node 1 contient la liste des blocs défectueux**
- **L'inode 2 concerne la racine**

Cas de Linux

❖ Organisation

Table d'i-nodes : contient les i-nodes numérotés de 1 à max i-node

Chaque i-node décrit un Fichier unique.

Lorsqu'un Fichier est supprimé alors son i-node est marqué libre et peut être attribué à un nouveau Fichier.

Cas de Linux

- **Structure d'un i-node**

Cas de Linux

❖ Structure d'un i-node

Les attributs

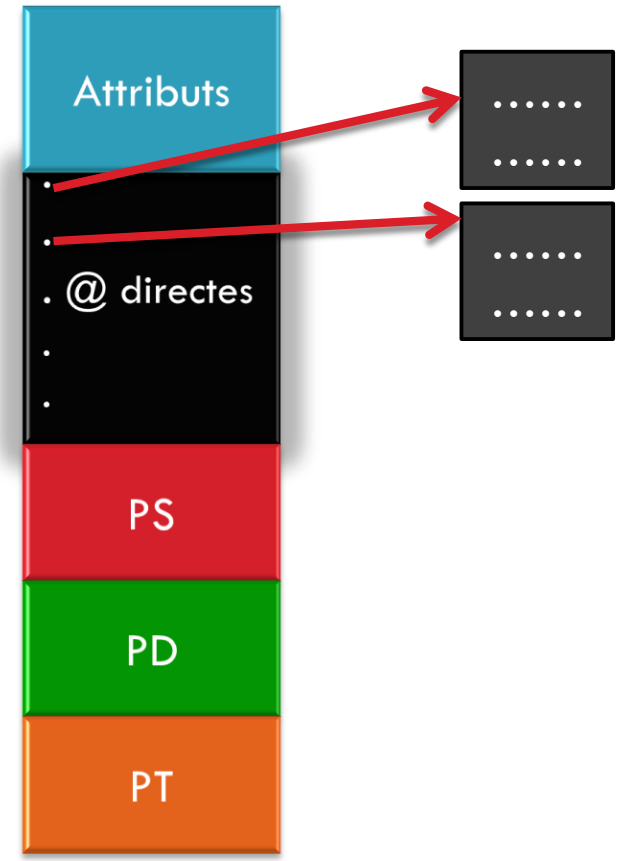
- Mode de protection
- Type
- UID
- GID
- Taille en Octet (réelle)
- Dates
  - atime : dernier accès
  - ctime : création
  - mtime : dernière modification
- …

Cas de Linux

❖ Structure d'un i-node

Les adresses directes

Indiquent les adresses des blocs de données (sur le disque) qui composent le Fichier.



Cas de Linux

❖ Structure d'un i-node

Si les adresses directes ne sont pas suffisantes pour lister les adresses des blocs des données qui composent le Fichier, on peut utiliser :

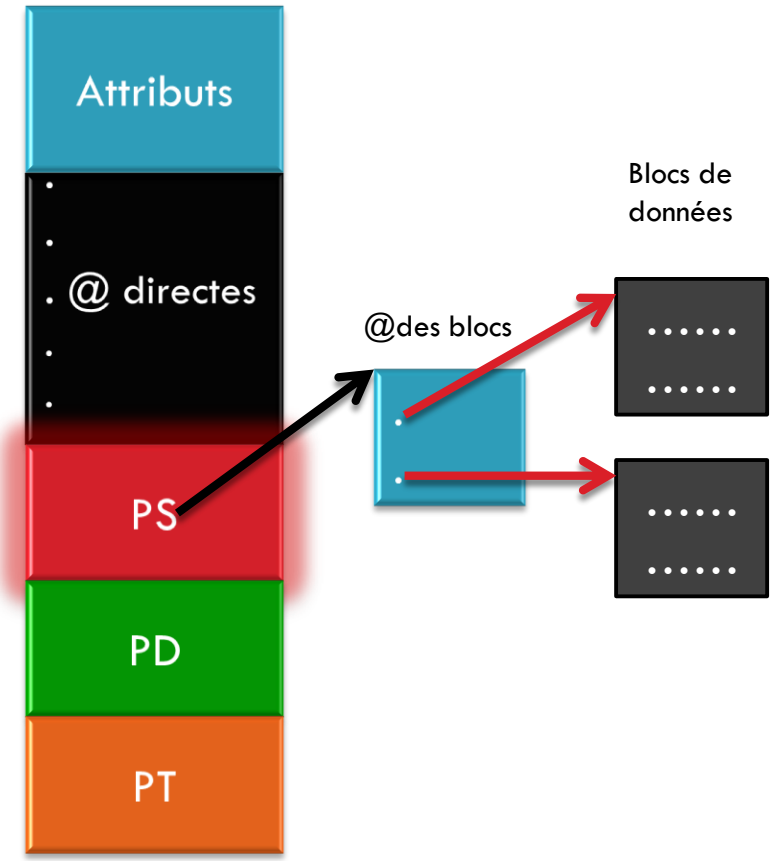
un pointeur d'indirection simple

Cas de Linux

❖ Structure d'un i-node

Si les adresses directes ne sont pas suffisantes pour lister les adresses des blocs des données qui composent le Fichier, on peut utiliser :

un Pointeur d'indirection Simple



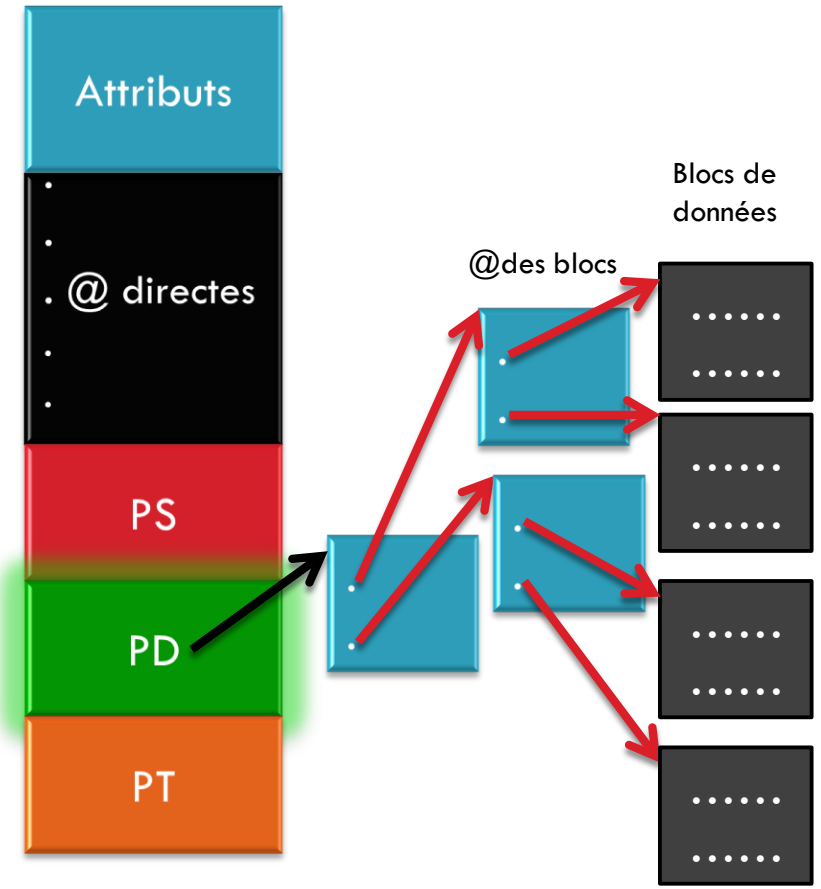
Cas de Linux

❖ Structure d'un i-node

Si les adresses directes ne sont pas suffisantes pour lister les adresses des blocs des données qui composent le Fichier, on peut utiliser :

un Pointeur d'indirection Simple

un Pointeur d'indirection Double



Cas de Linux

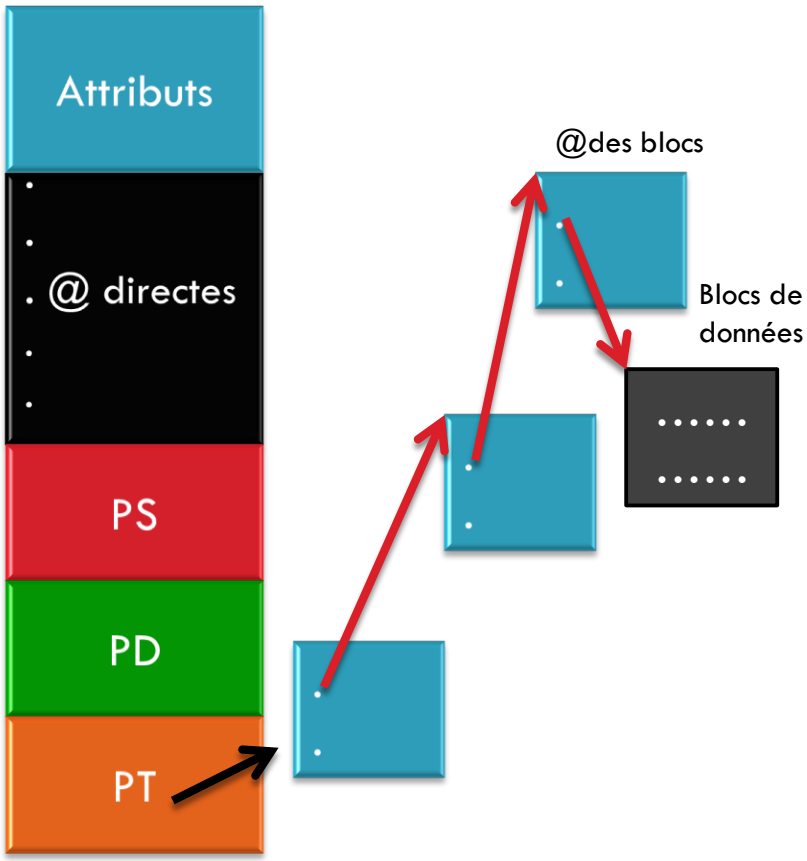
❖ Structure d'un i-node

Si les adresses directes ne sont pas suffisantes pour lister les adresses des blocs des données qui composent le Fichier, on peut utiliser :

un Pointeur d'indirection Simple

un Pointeur d'indirection Double

un Pointeur d'indirection Triple



Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1K Ø
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1K Ø
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

➡ **Chaque bloc peut contenir :**  @

Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1 KØ
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

➡ Chaque bloc peut contenir : 1 KØ / 4Ø @

Cas de Linux

- **UFS (Unix File System)**

La taille d'un bloc est	**1 KØ**
La taille d'i-node est	**64 Ø**
La taille d'une adresse est	**4 Ø**
L'i-node contient	**10 adresses directes**

Chaque bloc peut contenir : 256 @

Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1K Ø
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

➡ La taille de plus grand Fichier  Ø

Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1K Ø
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

➡ La taille de plus grand Fichier

$= (10 + 256 + 256^2 + 256^3) * 1$ KØ

≈ 16 GØ

Cas de Linux

❖ UFS (Unix File System)

La taille d'un bloc est	1K Ø
La taille d'i-node est	64 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	10 adresses directes

Cas de Linux

- **Ext2 (second extended filesystem - Linux)**

La taille d'un bloc est	1K Ø
La taille d'i-node est	128 Ø
La taille d'une adresse est	4 Ø
L'i-node contient	12 adresses directes

Cas de Windows

❖ Notion de Cluster

Le cluster est un ensemble de blocs adjacents

Cas de Windows

- **Organisation**

Disque dur

Cas de Windows

❖ Organisation

FAT1 **table qui conserve le chaînage entre les clusters de chaque Fichier.**

Cas de Windows

❖ Organisation

FAT2 **copie de la FAT1 et ce pour des mesures de sécurité.**

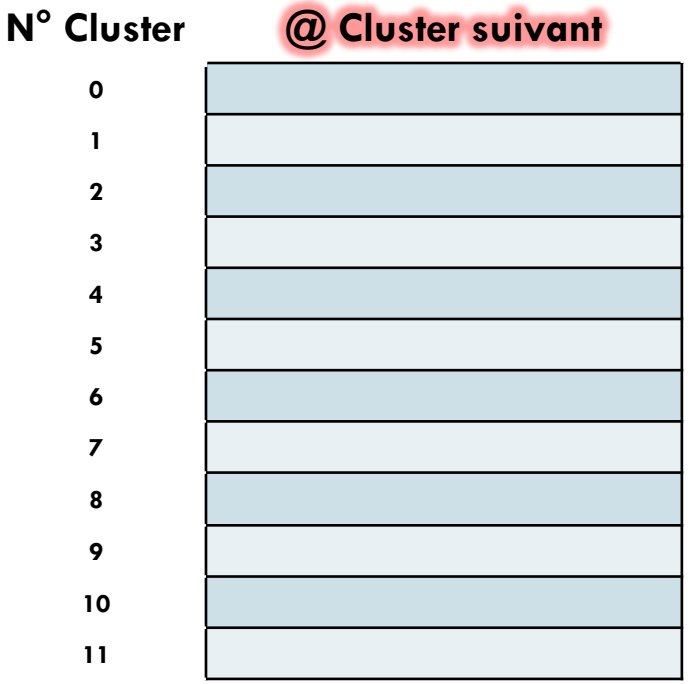
Cas de Windows

❖ Organisation

Racine **concerne le répertoire racine**

Cas de Windows

- **FAT**



Cas de Windows

❖ FAT

FAT 16 : L'adresse d'un cluster est sur 16 bits = 2 Ø (SE < Windows 95)

FAT 32 : L'adresse d'un cluster est sur 32 bits = 4 Ø (Windows 98)

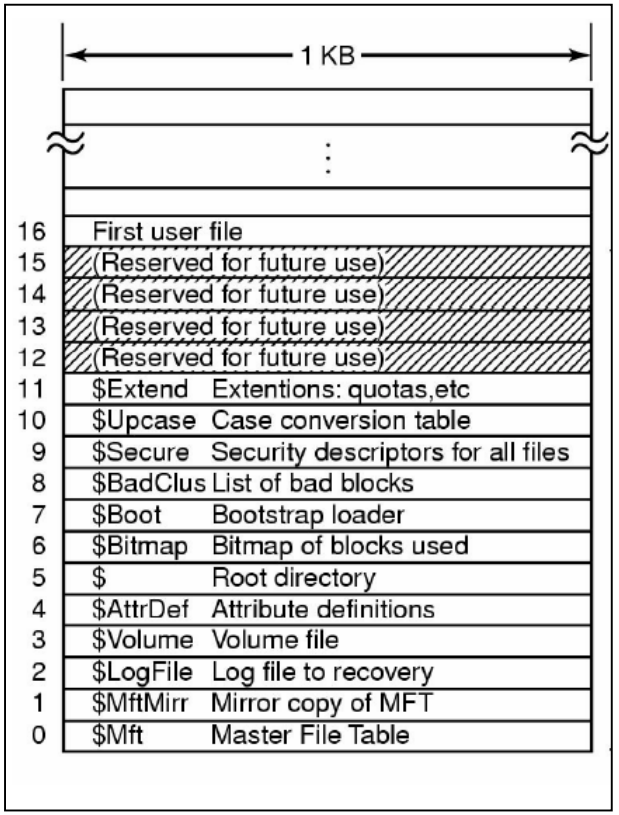
Cas de Windows

❖ NTFS (New Technologie File System)

Utilise la table MFT (Master File Table) depuis Windows NT

Chaque ligne est un enregistrement :

- **Attributs**
- **Description**



Cas de Windows

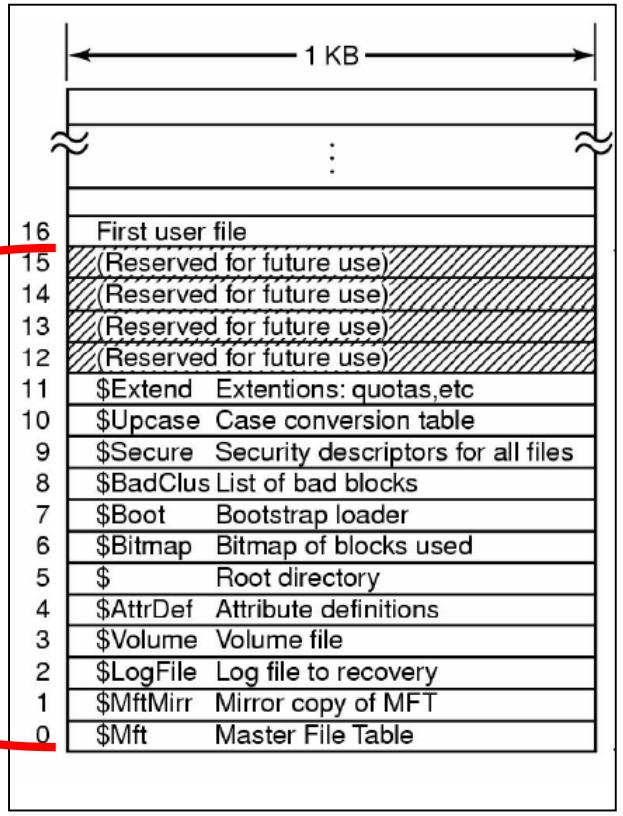
❖ NTFS (New Technologie File System)

Utilise la table MFT (Master File Table) depuis Windows NT

Contient au maximum 2^{48} enregistrements

Les 16 premières lignes

sont réservées pour le SE



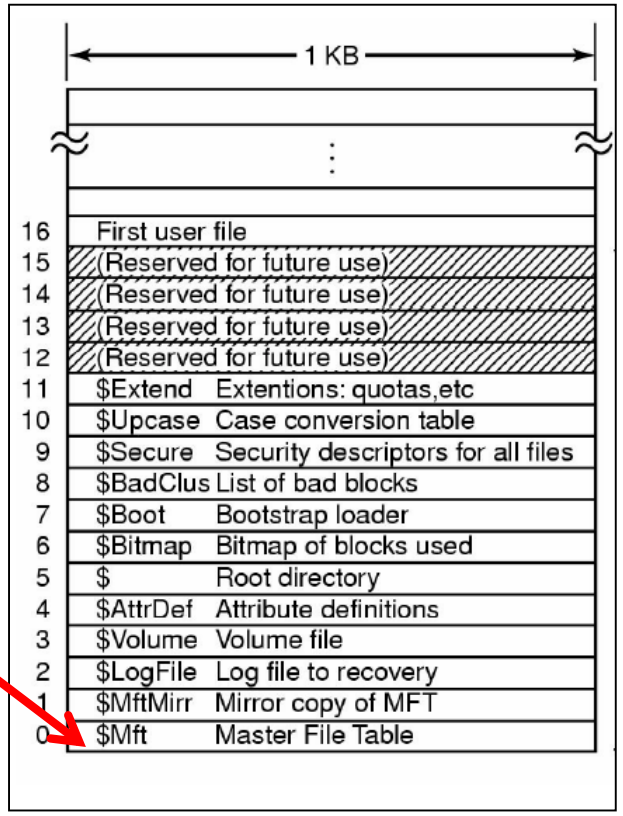
Cas de Windows

❖ NTFS (New Technologie File System)

Utilise la table MFT (Master File Table) depuis Windows NT

Le premier enregistrement décrit

l'emplacement les blocs du Fichier

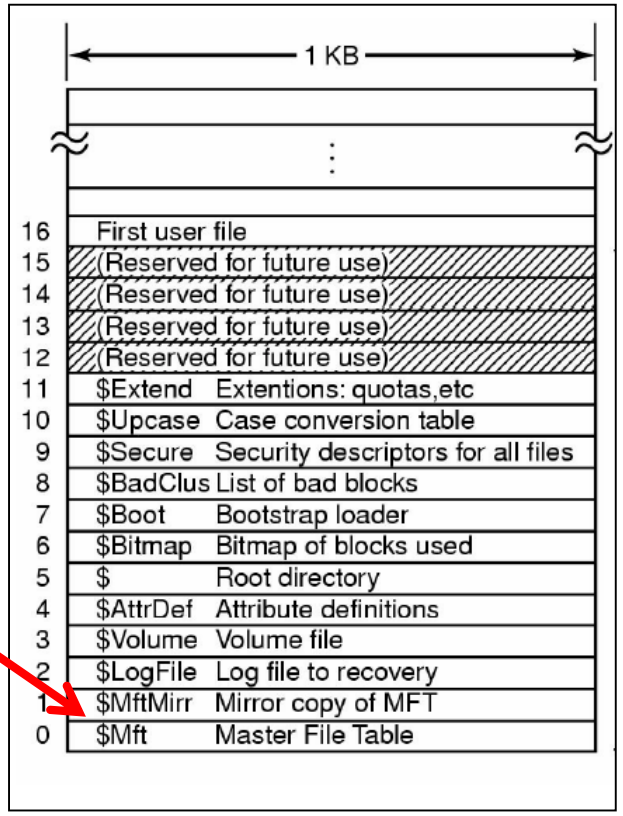


Cas de Windows

❖ NTFS (New Technologie File System)

Utilise la table MFT (Master File Table) depuis Windows NT

Le deuxième enregistrement est une copie du précédent.

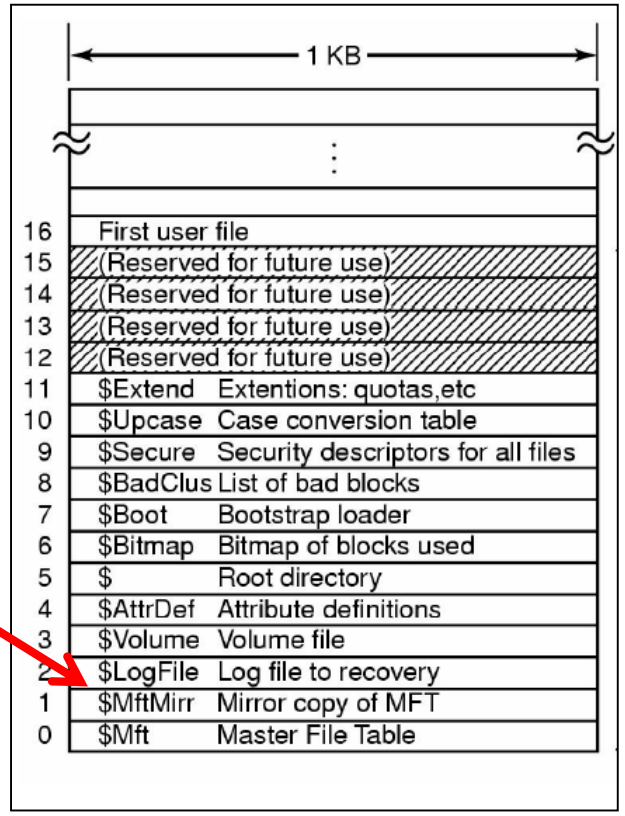


Cas de Windows

❖ NTFS (New Technologie File System)

Utilise la table MFT (Master File Table) depuis Windows NT

Le fichier Log (journal du SE) enregistre toutes les manipulations faites sur les Fichiers



FIN

Madame Khaoula ElBedoui-Maktouf

2ème année Ingénieur Informatique