Fonctionnement d'un Système de Gestion de Fichiers (SGF) - les disques

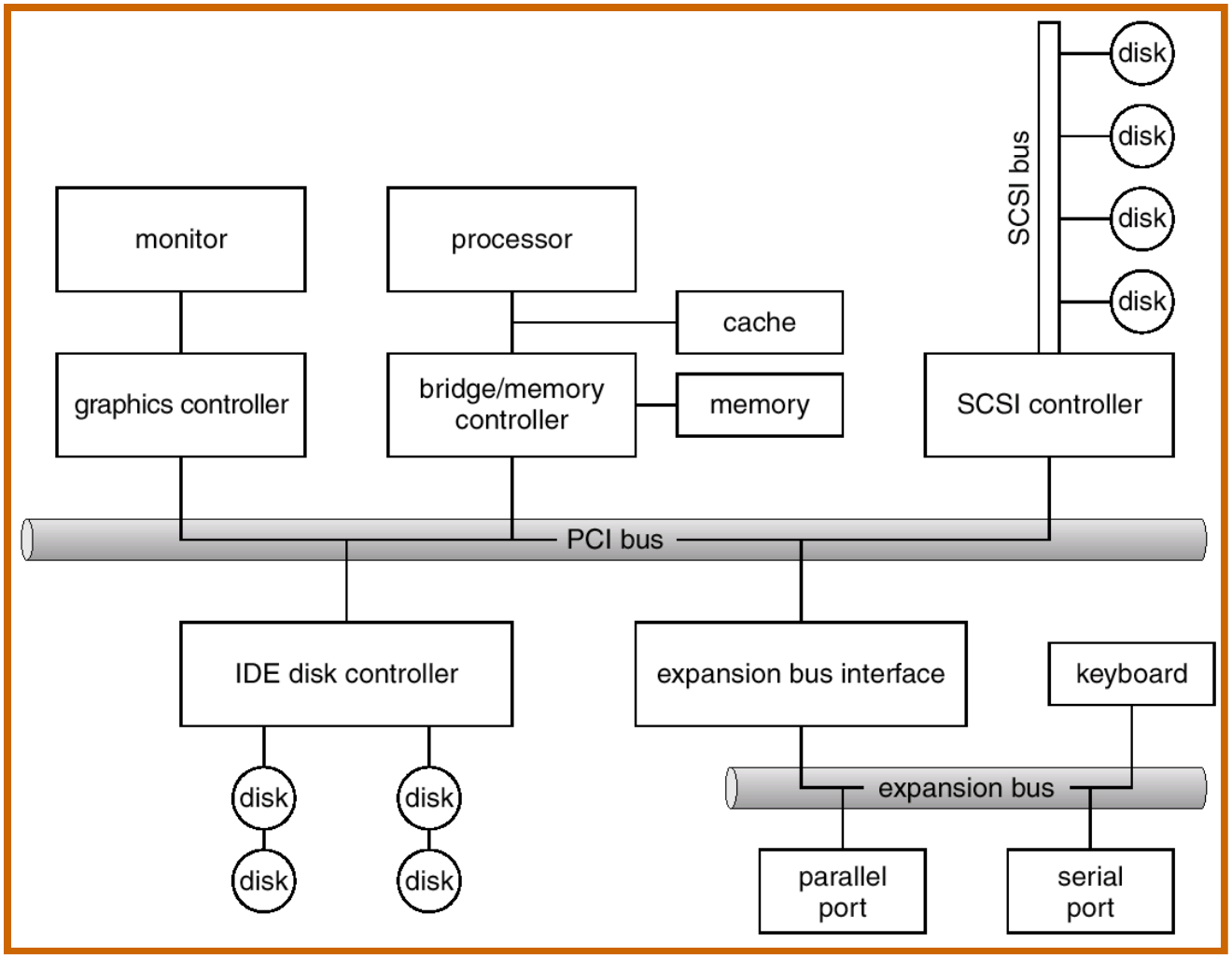
christophe.cerin@iutv.univ-paris13.fr

27 décembre 2012

Systèmes de fichiers

- Propriétés attendues :
  - **Stocker de très grandes quantités d'information**
  - **L'information doit survivre à la fin du processus qui l'a créée**
  - **Plusieurs processus doivent pouvoir y accéder concurremment**
- Vue abstraite: tableau dont les cases (blocs) sont de taille fixe :
  - **Comment trouver ?**
  - **Comment s'assurer qu'un utilisateur accède uniquement aux données d'un fichier donné ?**
  - **Comment savoir quels sont les blocs libres ?**

I- Organisation d'un PC (ancien)



Le chemin des données

- Remarques technologiques : S-ATA, PCI Express et PCI-X (sont des normes récentes concernant les I/O)
- L'envoi d'octets depuis un disque vers la carte réseaux passe par la RAM et le bus mémoire (2 fois)
- Ceci est dû à la méthode d'accès (les choix pour sauvegarder et gérer les données entre composants différents)

Structures logiques de fichiers

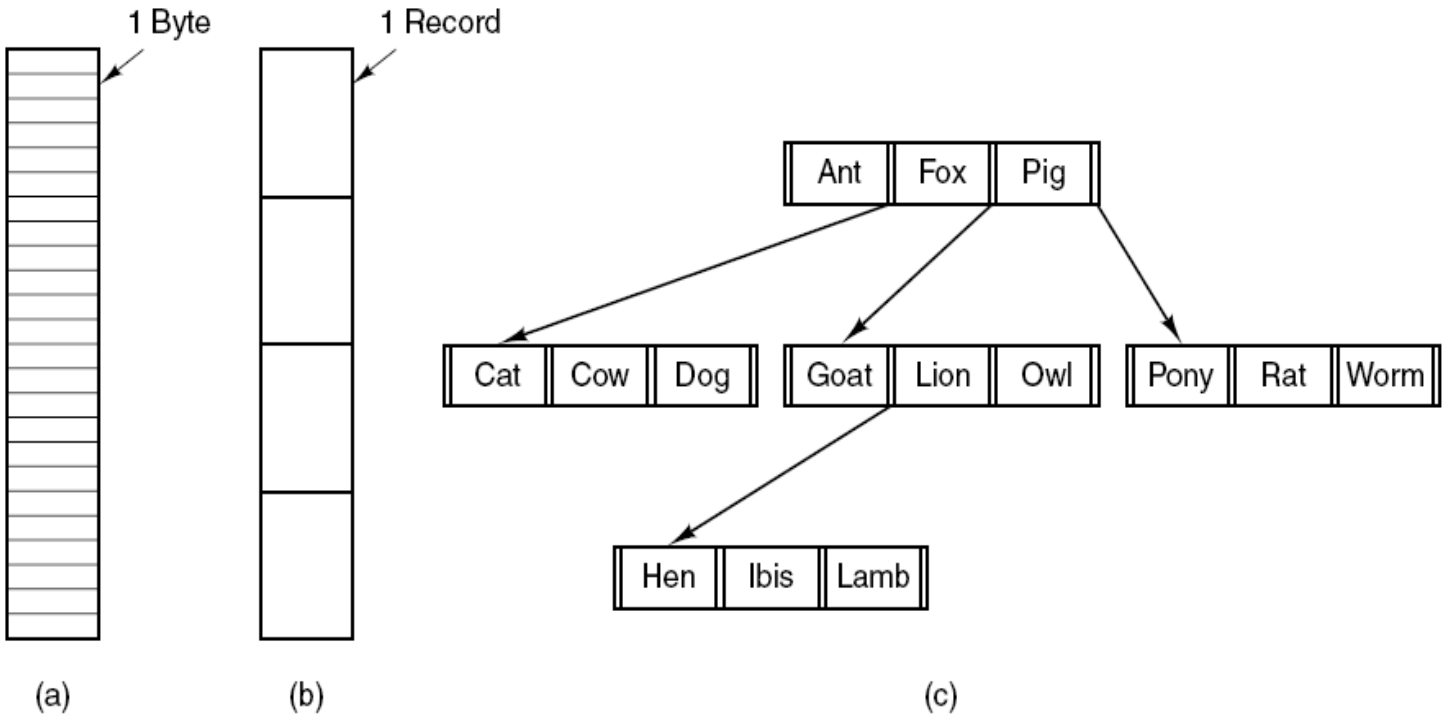


Figure 1. Three kinds of files. (a) Byte sequence. (b) Record sequence. (c) Tree.

Organisation d'un fichier

- Le fichier séquentiel :

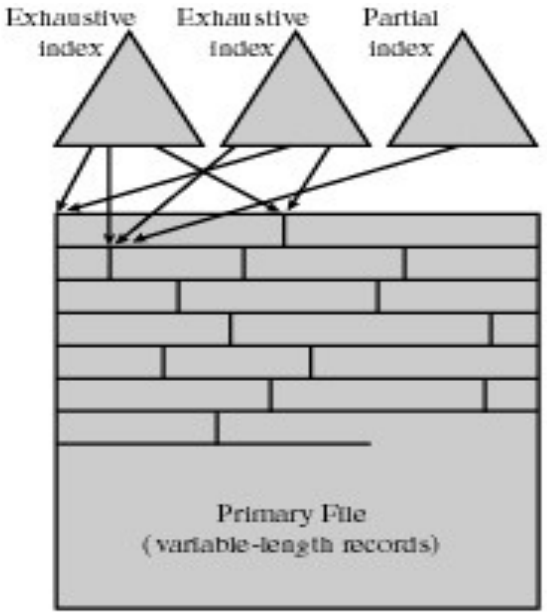
1. Un format fixé est utilisé pour les enregistrements
2. Les enregistrements sont de même longueur
3. Tous les champs sont les mêmes
4. Un champs est appelé la clé… pour identifier l'enregistrement
5. Nouveaux enregistrements sont traités par lots pour être fusionnés

Organisation d'un fichier

- Le fichier (séquentiel) indexé :

1. Une table (un indexe) est utilisé pour retrouver rapidement un enregistrement
2. Le reste est inchangé
3. Exemple d'indexe : (clé, ensemble des numéros des blocs ou apparaissent la clé). Si l'ensemble est trié, on peut faire une recherche dichotomique de coût O(log n). Discussion : coût associé pour un ajout/suppression ? On peut avoir plusieurs indexes pour une même clé : compliqué !!!

Fichier indexé



(d) Indexed File

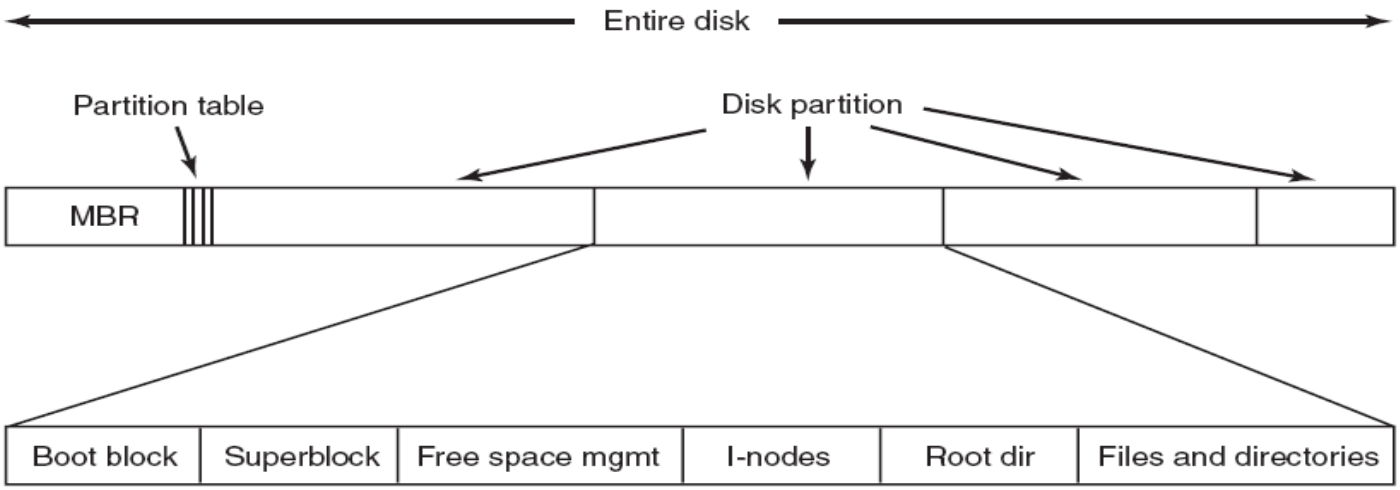
Figure 12.3 Common File Organizations

Ici on a plusieurs indexes et des records de taille variable
RQ : les indexes sont aussi rangés sur les disques

Fichier à accès direct (hash)

- Objectif : accéder avec un coût constant (en une seule opération d'I/O : idéal) et pas fonction de la taille du fichier ;
- H(key_field) => adresse du bloc
- Problème des collisions

Organisation physique d'un système de gestion de données



Allocation contiguë

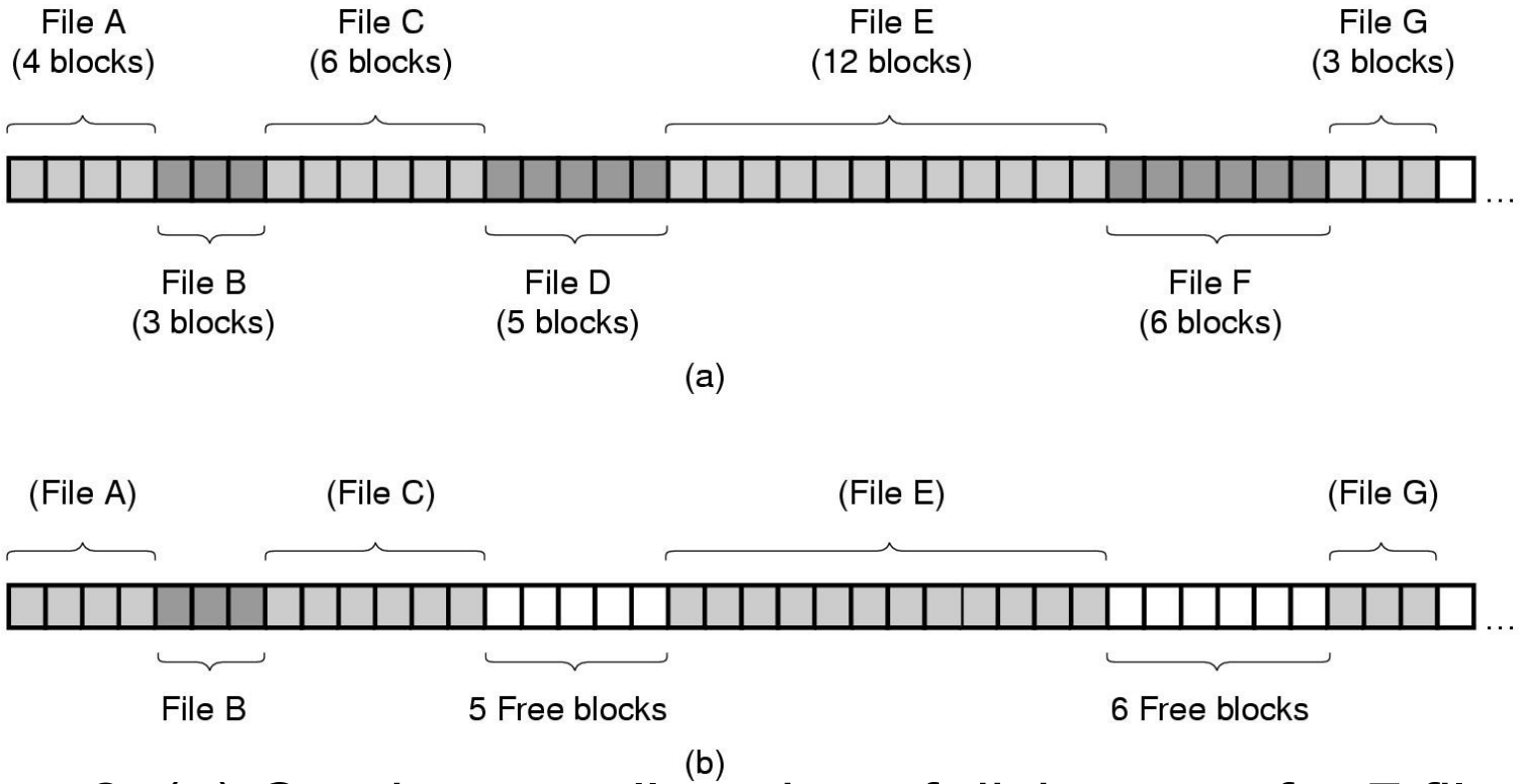
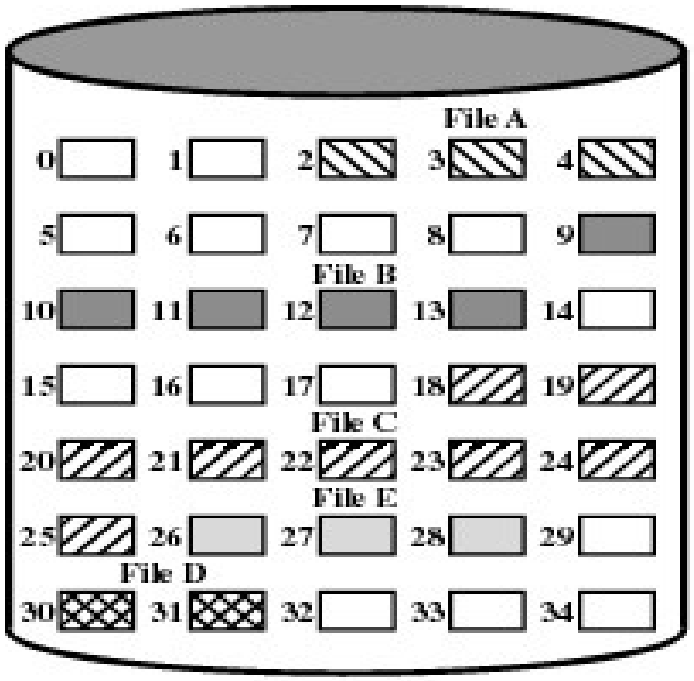


Figure 2. (a) Contiguous allocation of disk space for 7 files. (b) The state of the disk after files D and F have been removed.

Méthodes d'allocation des blocs sur le disque

- Allocation contiguë permet de ne garder comme information pour l'acces qu'une seule adresse et le nombre de blocs.
- C'est bon pour les performances... mais quand on supprime, on crée des « trous » d'où la nécessité de faire autrement.

Allocation contiguë

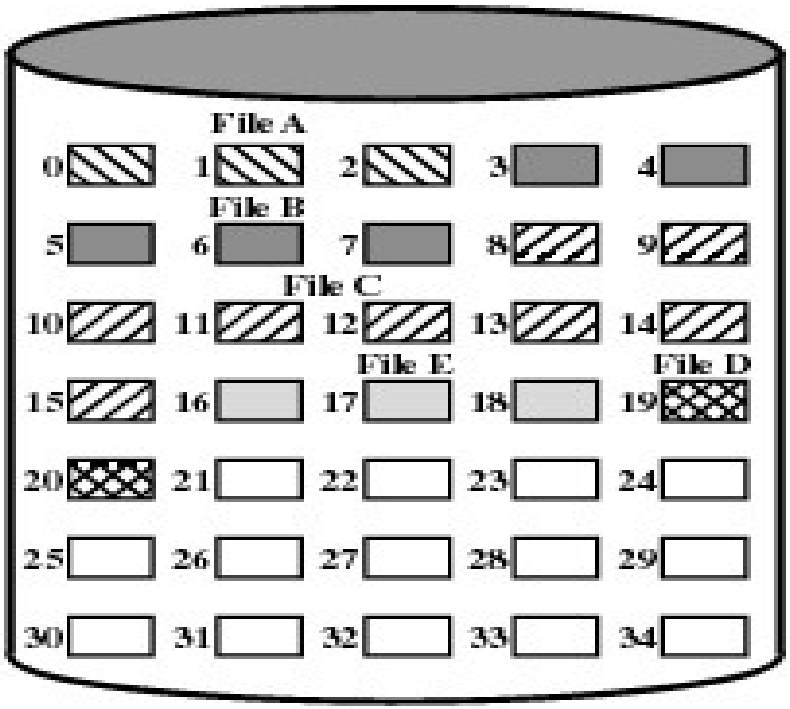


File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

Figure 12.7 Contiguous File Allocation

Allocation contiguë et compactage

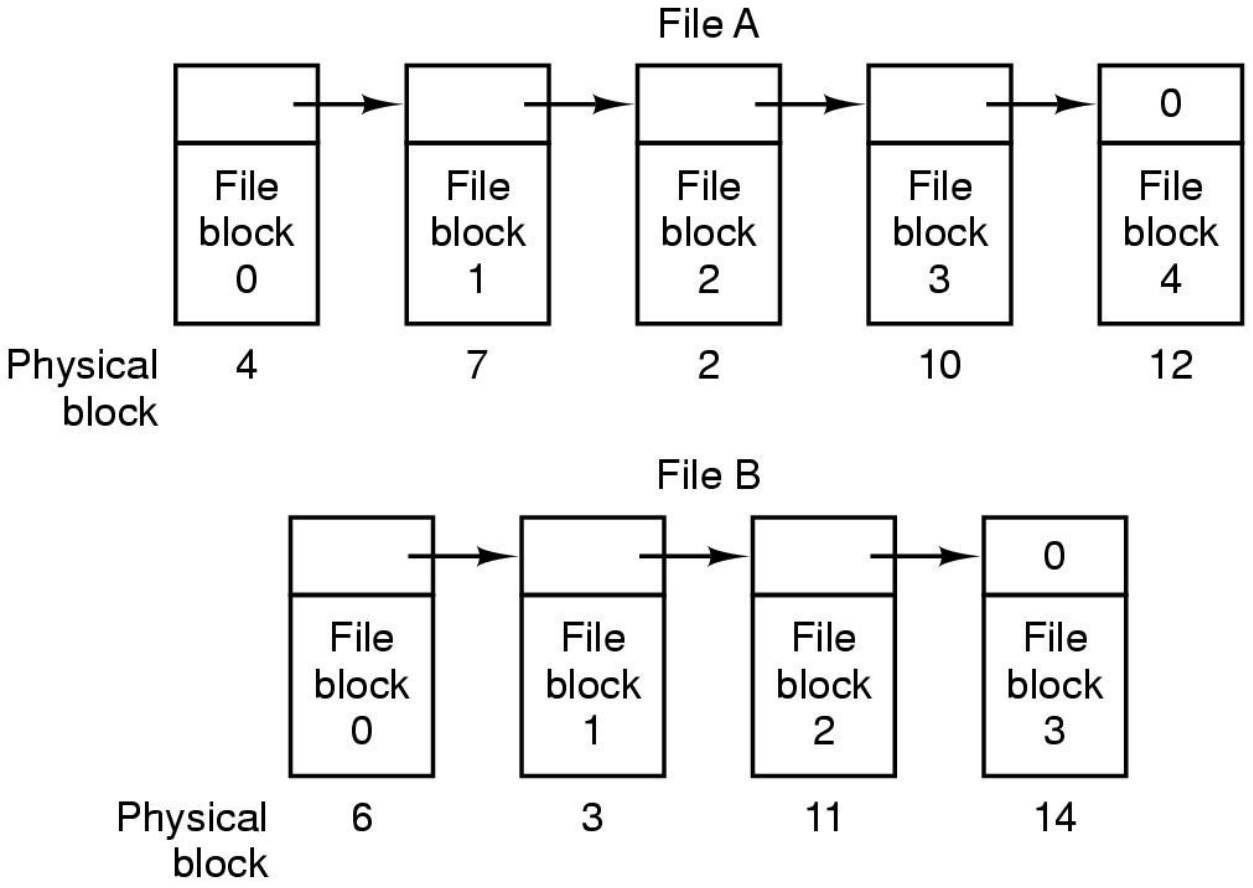


File Allocation Table

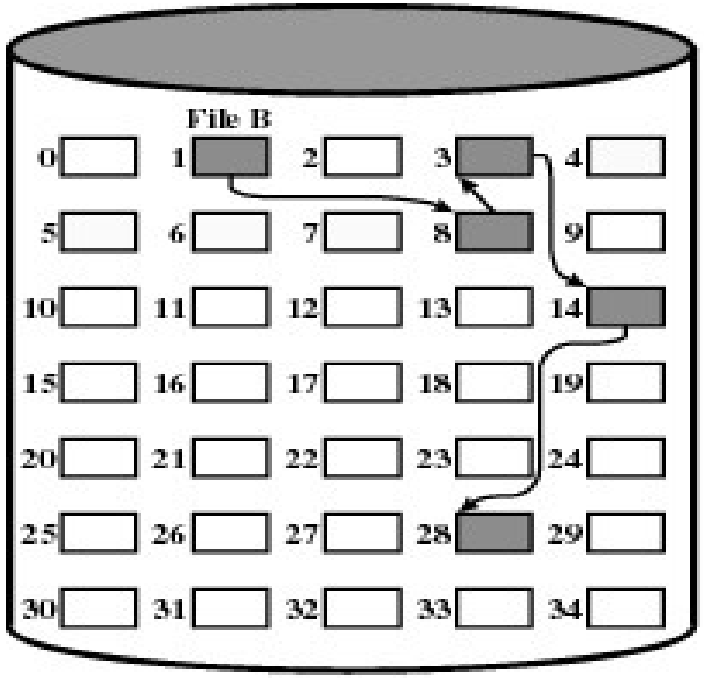
File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

Figure 12.8 Contiguous File Allocation (After Compaction

Allocation chainée



Allocation chainée



File Allocation Table

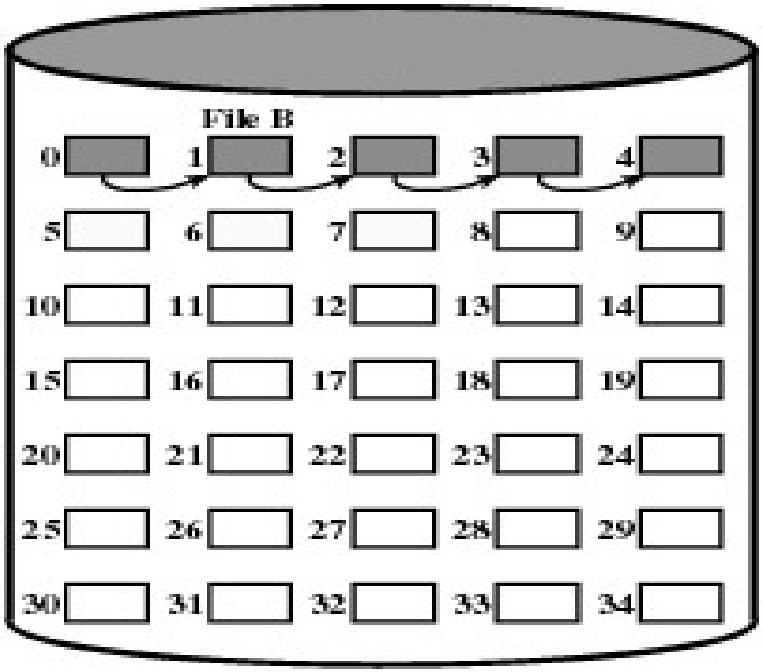
File Name	Start Block	Length
• • •	• • •	• • •
File B	1	5
• • •	• • •	• • •

Il suffit de connaître le bloc de début et la longueur (nobre de blocs)

Le principe de localité n'est pas respecté

Figure 12.9 Chained Allocation

Allocation chainée et consolidation

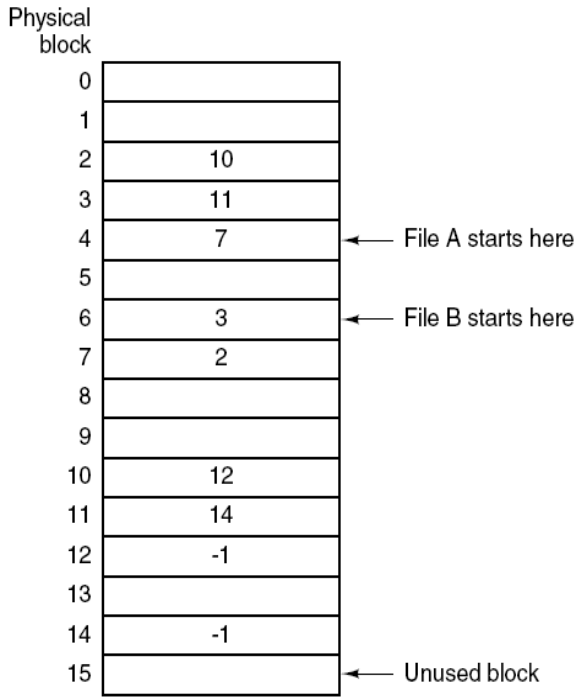


File Allocation Table

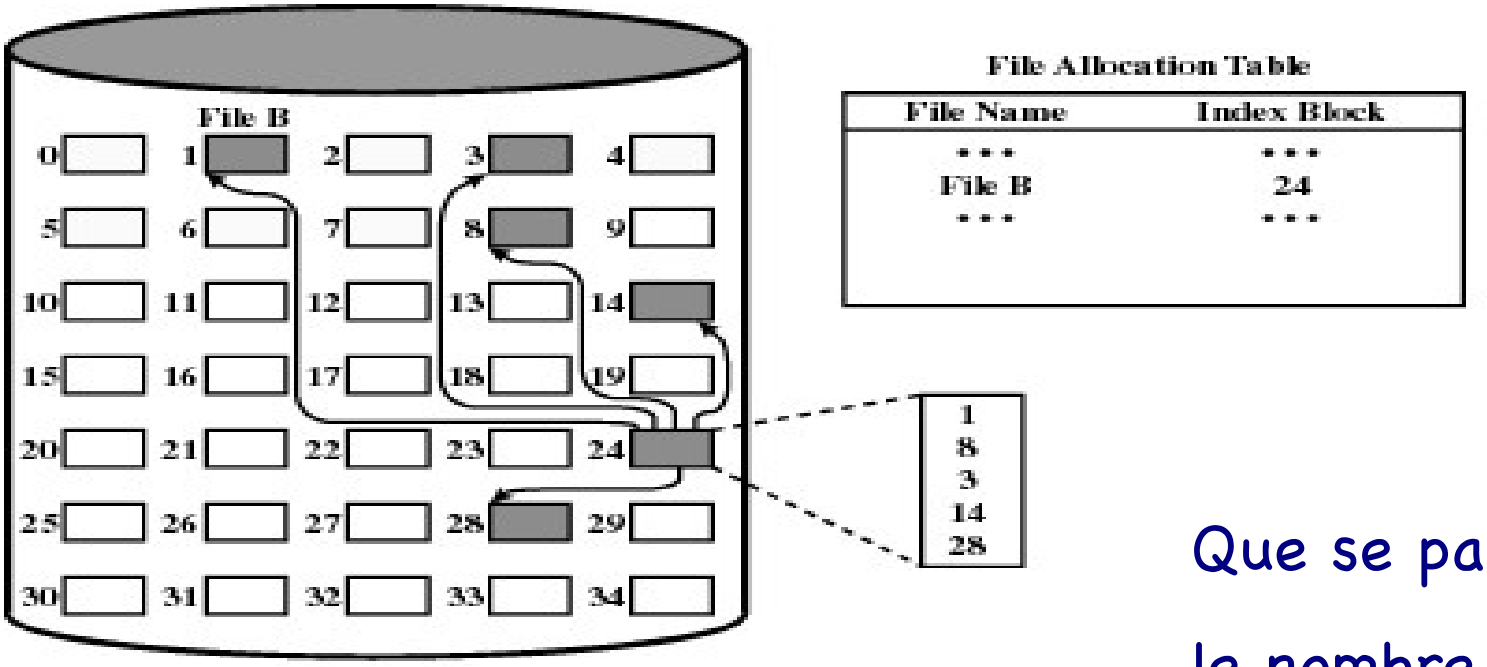
File Name	Start Block	Length
* * *	* * *	* * *
File B	0	5
* * *	* * *	* * *

Figure 12.10 Chained Allocation (after consolidation)

Allocation par liste chainée utilisant une table en mémoire



Allocation des indexes



Que se passe t'il si le nombre de blocs est grand ?

Figure 12.11 Indexed Allocation with Block Portions

Allocation des indexes

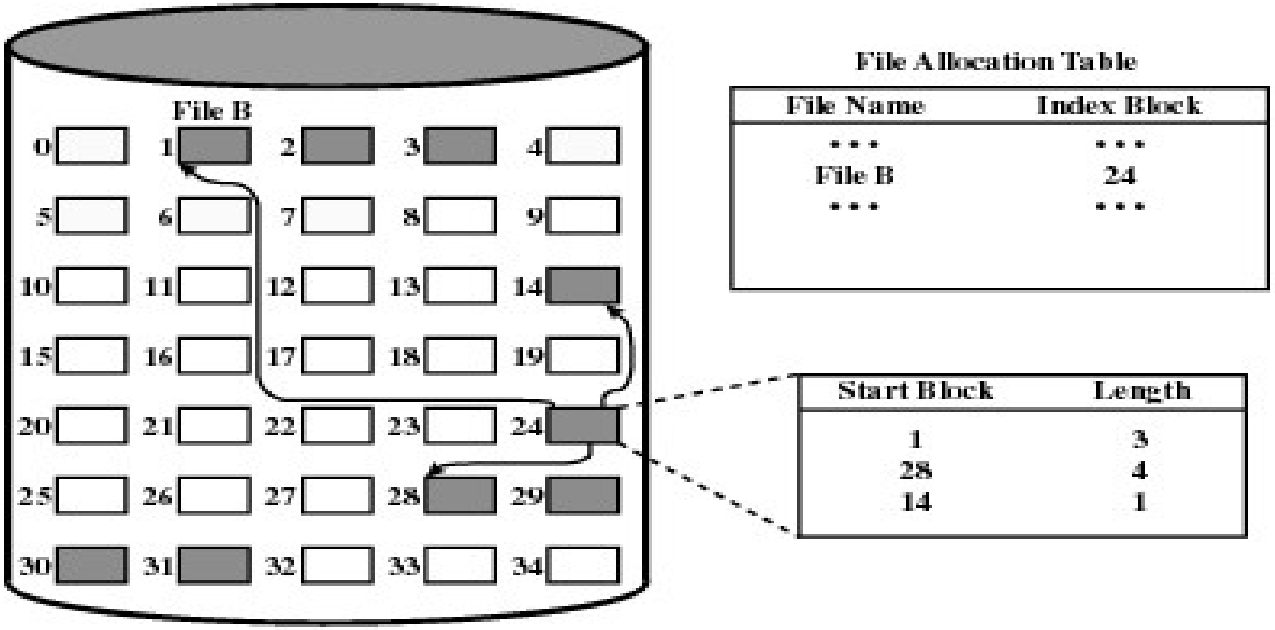
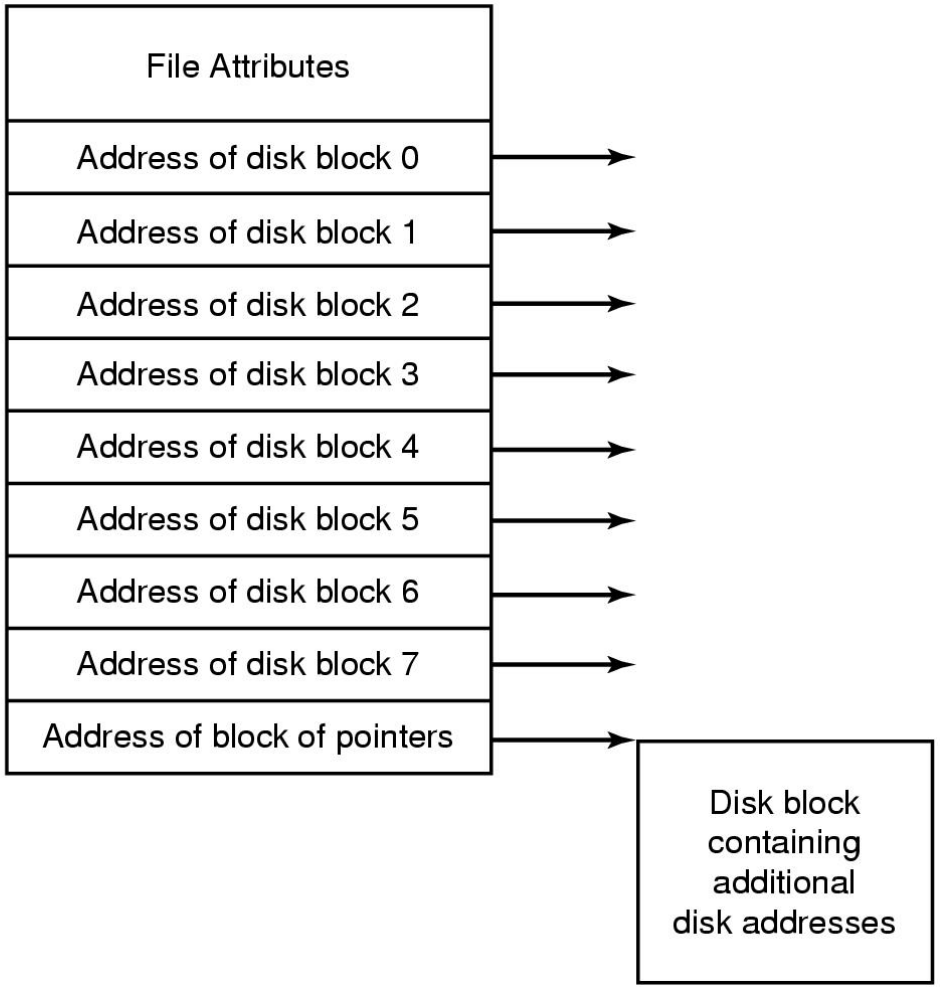
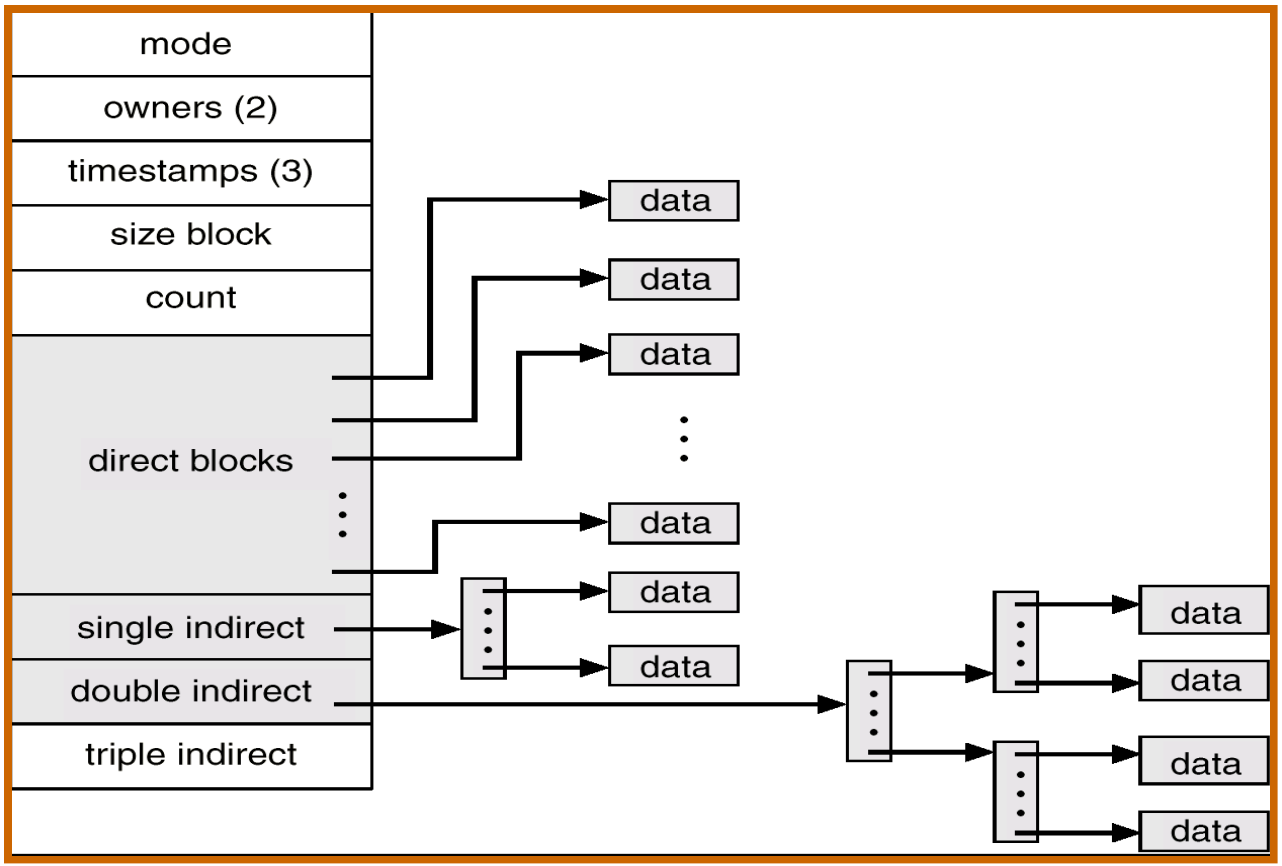


Figure 12.12 Indexed Allocation with Variable-Length Portions

I-nodes (terme du monde Unix)



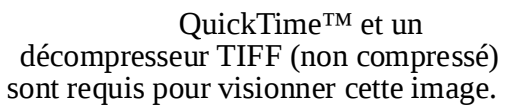
Le cas d'Unix


mode
owners (2)
timestamps (3)
size block
count
direct blocks
single indirect
double indirect
triple indirect
data
data
data
data
data
data
data
data
data
data

II- Structure d'un disque

- Un disque peut être vu comme un tableau à 1 dimension de blocs logiques ;
- Le bloc est la plus petite unité de transfert (512 octets ?) ;
- Le tableau à 1 dimension est « mappé » dans les secteurs physiques
- Le secteur 0 est mappé avec le premier secteur de la première piste du cyclindre extérieur

Organisation physique



Cylindres
Pistes
Secteurs
Plateaux
Têtes de lecture

Exemple : pour un disque de 1024 cylindres, 16 têtes et 63 secteurs (de 512 k), on aura donc une capacité totale de disque dur de 1024 * 16 *63 * 512 = 528 482 304 octets soit environ 504 Mo (**(528 482 304 / 1024) / 1024**).

Organisation physique

```
CFDISK(8)                                   Linux Programmer's Manual

NAME
       cfdisk - Curses based disk partition table manipulator
                for Linux

SYNOPSIS
       cfdisk  [  -agvz  ]  [ -c cylinders ] [ -h heads ]
               [ -s sectors-per-track ] [ -P opt ] [ device ]
```

Les commandes de gestion dépendent de la version utilisée d'Unix !!!

Organisation physique

```
     cfdisk 0.8i                                Disk Drive: /dev/hda
  Heads: 64   Sectors per Track: 63   Cylinders: 528Name
Flags         Part Type      FS Type                  Size (MB)
------------------------------------------------------------------------
/dev/hda1   Boot          Primary        DOS FAT16 [NO NAME    ]     127.97
/dev/hda2                 Primary        Linux Swap                   33.47
/dev/hda3                 Primary        Linux                       878.07


[Bootable]  [ Delete ]  [  Help  ]  [Maximize]  [ Print  ] [  Quit  ]
[  Type  ]  [ Units  ]  [ Write  ]
Toggle bootable flag of the current partition
```

Système de fichier consistant

- Imaginons une panne avant que toutes les modifications (écritures) sur un bloc aient eu lieu
- Unix fsck (réponse pratique pour Linuxien)
- Définitions :
  - **Table « Blocks in use » : combien de fois un block est présent dans un fichier ;**
  - **Table « Free blocks » : combien de fois chaque bloc est présent dans la liste des blocs disponibles**

- Trois cas :
  - **(b) bloc manquant. Free block passe à 1**
  - **(c) 4 apparaît deux fois. 2 → 1**
  - **(d) le bloc 5 est présent dans 2 fichiers : allouer un nouveau bloc et copier le contenu de 5 dedans**

Système de fichier consistant

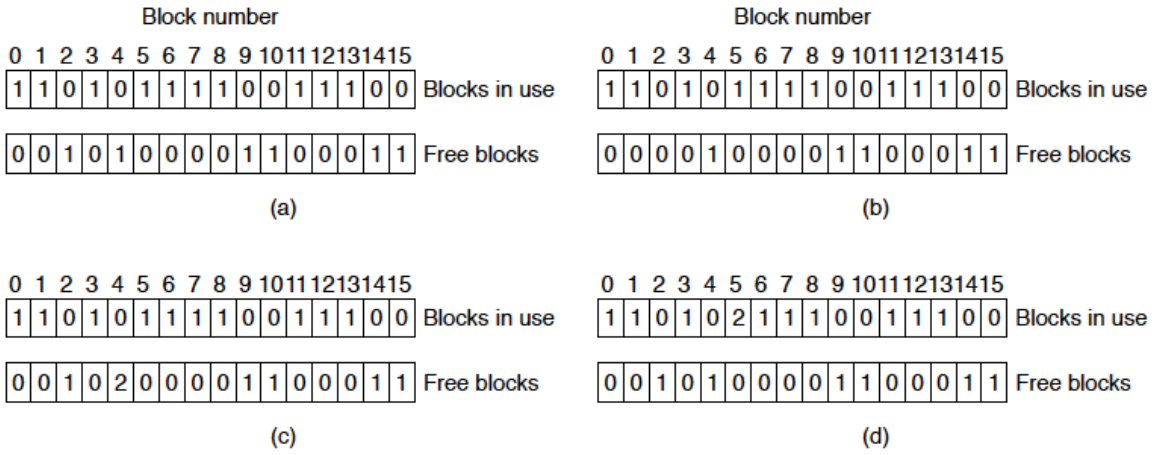
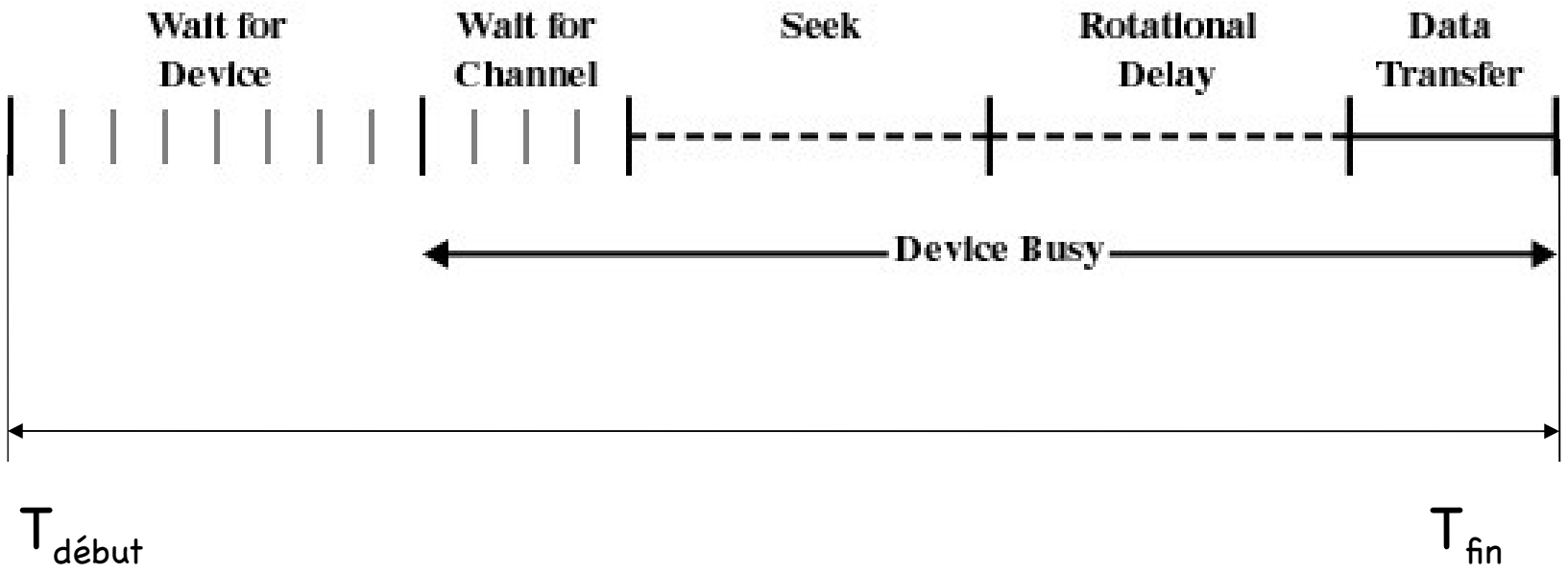


Figure 3. File system states. (a) Consistent. (b) Missing block. (c) Duplicate block in free list. (d) Duplicate data block.

Débit, performance d'un disque

- Le temps d'accès à un disque est modélisé par la somme du seek time et rotational latency :
- Seek time: temps pour positionner les têtes sur le bon cylindre ;
- Rotational latency: temps pour que la tête se positionne sur le bon secteur ;
- Disk bandwidth: nombre total d'octets transférés / ($T_{fin} - T_{début}$)

Transfert



data transfert << wait+seek+rotational => poor bandwidth

On a donc besoin d'un... cache

Calcul plus précis

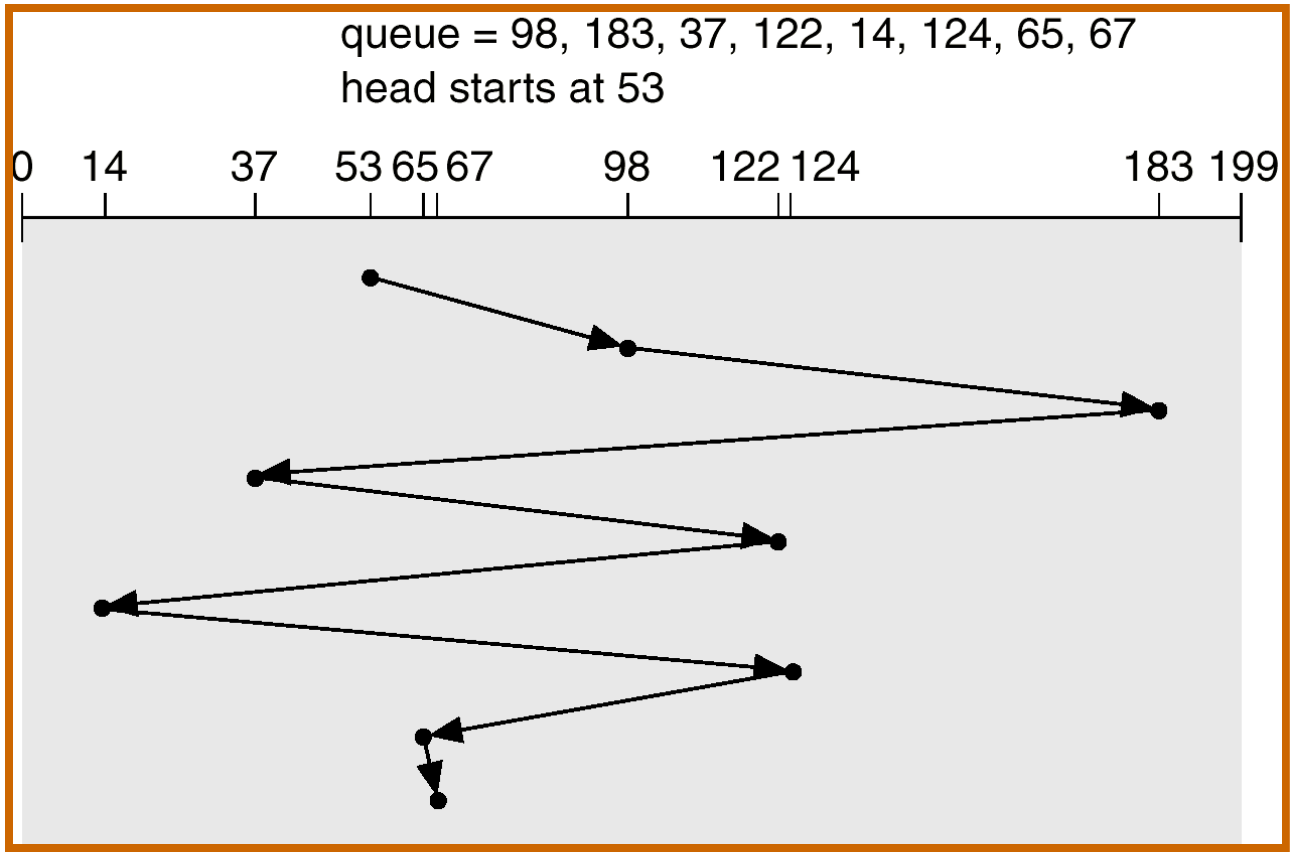
- Estimated seek time T_s = n*m+s (n=nombre de pistes traversées, m=cte propre au disque, s=startup time)
- Rotational delay : s'exprime en RPM (7200, 5400, 4200)
- Transfer Time T=b/(r*N) avec b=nombre d'octets à transférer, N=nombre d'octets par piste, r=vitesse de rotation en rpm)
- Total average seek time T_a = T_s+1/2r+b/(r*N)

Ordonnancement des accès au disque

- Si les données d'un même fichier sont un « peu partout » sur le disque alors la somme des $T_{accès}$ est grande !
- Plusieurs algorithmes existent pour ordonnancer les requêtes d'I/O
- Imaginons qu'une file de requêtes contienne les blocs suivants : 98, 183, 37, 122, 14, 124, 65, 67 et que la tête soit positionnée sur le bloc 53.

Politique First Come Fist Serve

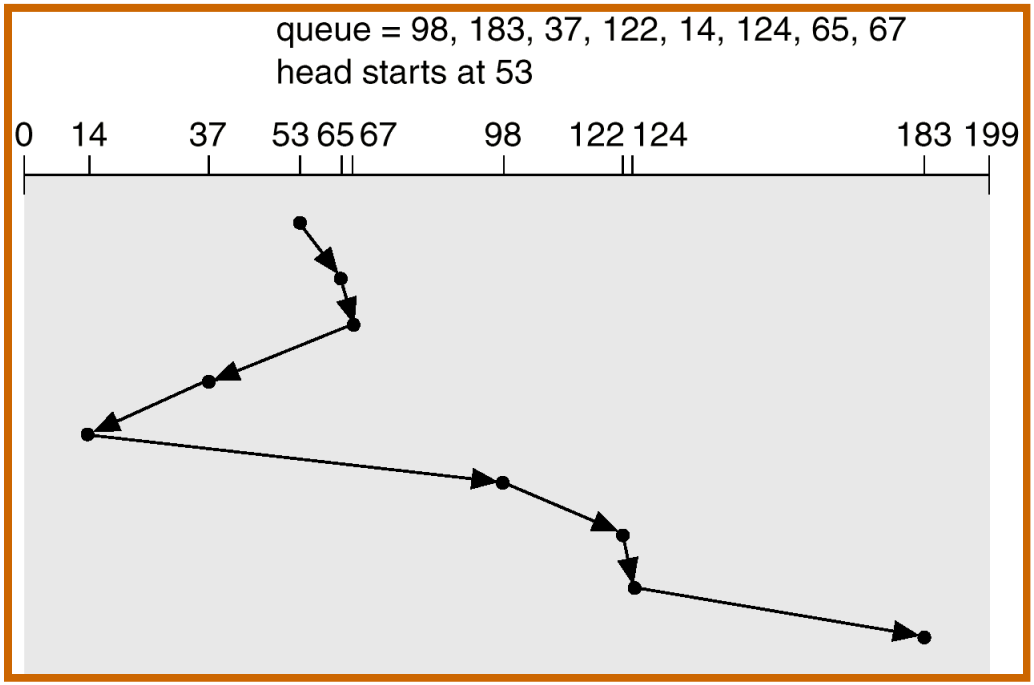
Illustration shows total head movement of 640 cylinders.



FCFS

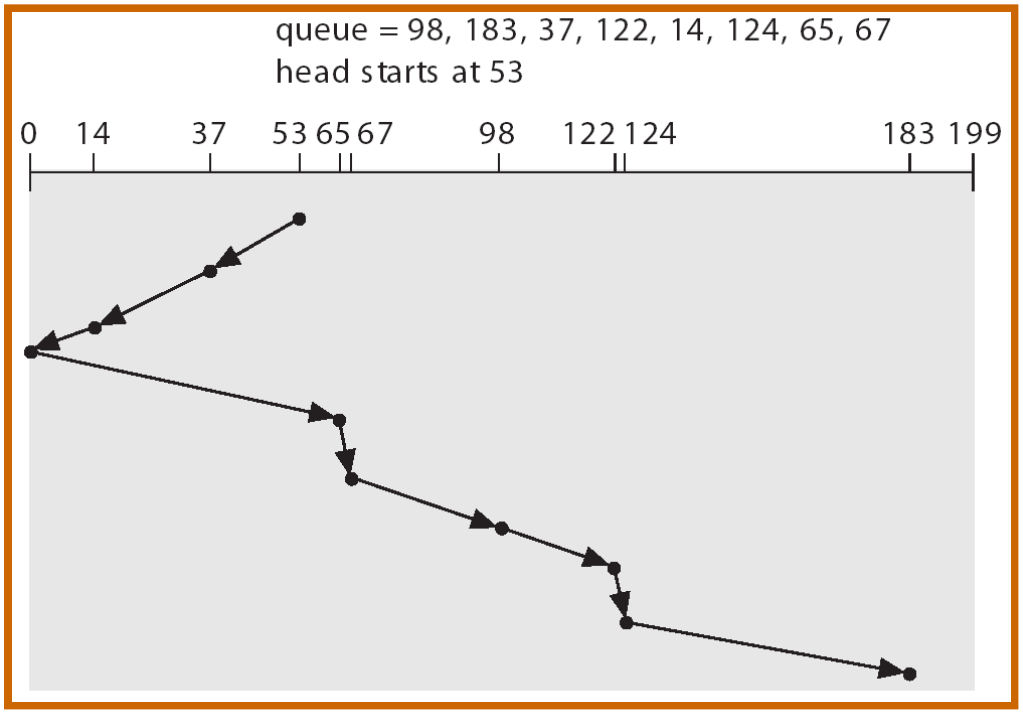
Shortest Service Time First (SSTF)

- Select requête avec le minimum seek time à partir de la position courrante de la tête
- Il peut y avoir famine (si on réalimente la file)
- Ici : 236 movements



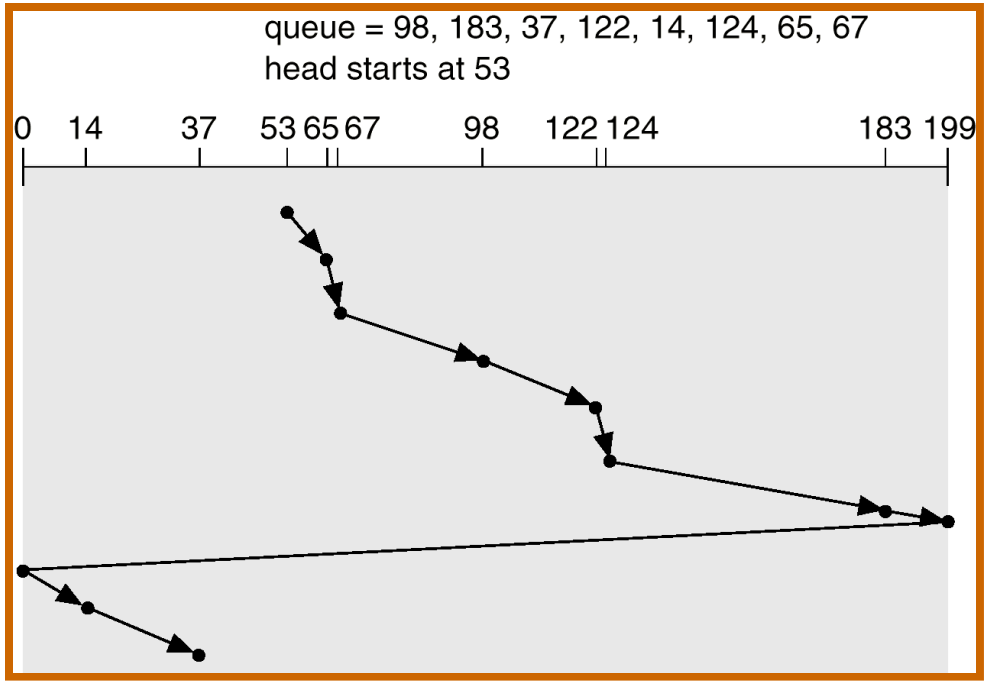
Politique SCAN (balayage)

- Le bras de la tête va « au début » du disque puis « à la fin »
- Principe de l'ascensseur
- Ici : 208 mouvements



Politique C-SCAN (circular)

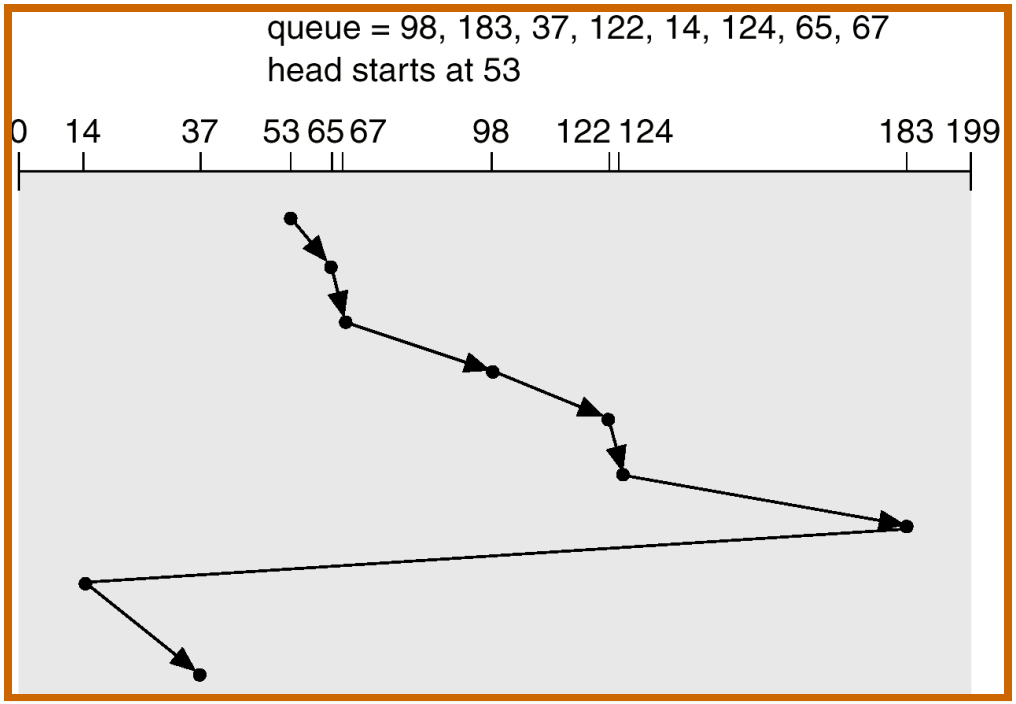
- Fournit un temps d'attente plus uniforme que SCAN
- Gestion par liste circulaire de cylindres
- Quand on « atteint » la fin du disque on revient au début sans rien faire
- Balayage dans un sens unique



Politique C-LOOK

- Version de C-SCAN

- Quand on « atteint » la dernière position la tête revient à la première position sans rien faire
- Balayage dans un sens unique



Politique N-step-SCAN & FSCAN

- L'ordonnanceur gère des files de taille N
- Au moment d'un ajout, on dépose dans une file qui n'est pas en cours de traitement
- Chaque file a une politique SCAN

- Deux files
- Une file vide pour recevoir les nouvelles requêtes

Quel choix pour des performance ?

- Le meilleur algorithme est à choisir en fonction du motif d'accès... mais il est inconnu à priori
- Le système de fichier devrait tous les implémenter et être capable de commuter de l'un à l'autre pendant l'exécution
- SSTF ou LOOK sont choisis par défaut

Exemple de système de fichiers : ext3

ext3 alloue les blocs libres juste à côté des autres blocs utilisés par le fichier, ce qui a pour effet de minimiser l'espace physique entre les blocs.

ext3 est néanmoins par définition fragmenté, c'est pourquoi son successeur ext4 inclut un utilitaire de défragmentation natif travaillant au niveau des bits et gérant la défragmentation à chaud.

Un système de fichiers ext3 est créé dans une partition or le format de table de partition géré par fdisk6 ne permet pas de créer des partitions de plus de 2,2 Tio (2^32*512)7. Utilisez la commande parted et le format de table de partitions GPT8 pour s'affranchir de cette limite de 2,2 Tio.

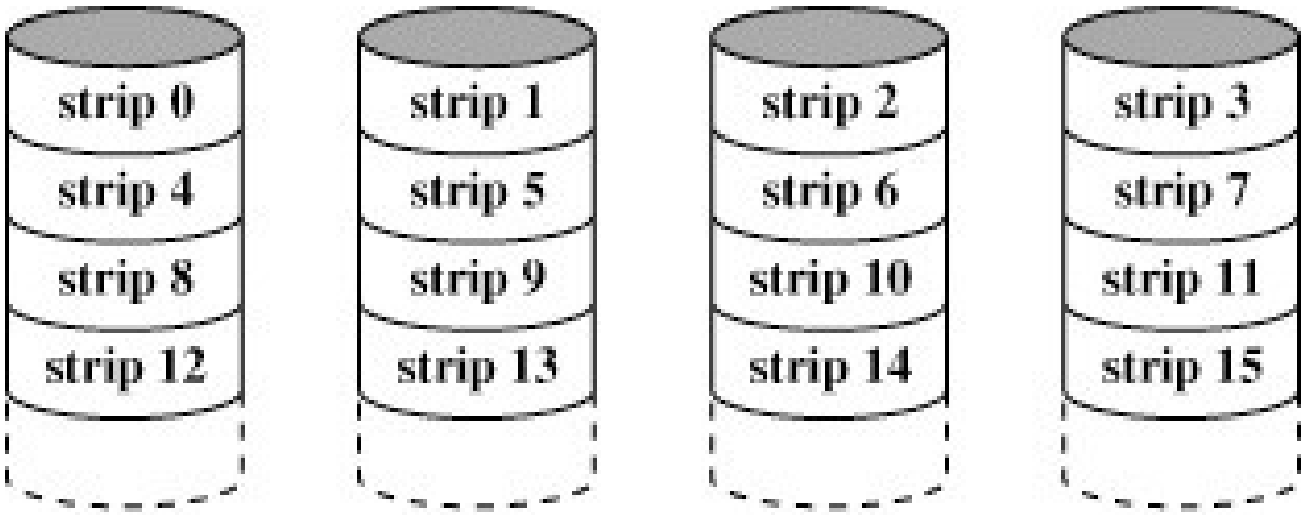
ext3	
Développeur	Stephen Tweedie
Nom anglais	Extended file system 3
Introduction	novembre 2001 (Linux 2.4.15)
Identificateur de partition	0x83 (MBR)
Structure	
Contenu des répertoires	tableau, arbre h avec index de répertoires
Allocation de fichiers	champs de bits (espace libre), table (metadata)
Mauvais blocs	table
Limitations	
Taille maximale de fichier	16 Gio – 2 Tio
Nombre maximal de fichiers	variable[1]
Taille maximale du nom de fichiers	255 octets (soit 255 caractères en ASCII).
Taille maximale de volume	2 Tio – 32 Tio
Caractères autorisés dans les noms de fichiers	tous les caractères Unicode sauf NUL et '/'

Technique du RAID pour augmenter les performances… et la sureté (pérénité des données)

- Principe : un contrôleur qui supervise plusieurs disques en autorisant des requêtes simultanées + sureté via la redondance
- Il y a différents niveaux de RAID

RAID=Redundant Array of Inexpensive Disks

RAID-0 (non redondant)



(a) RAID 0 (non-redundant)

C'est bon pour les perfs, pas pour la pérénité !

RAID-0

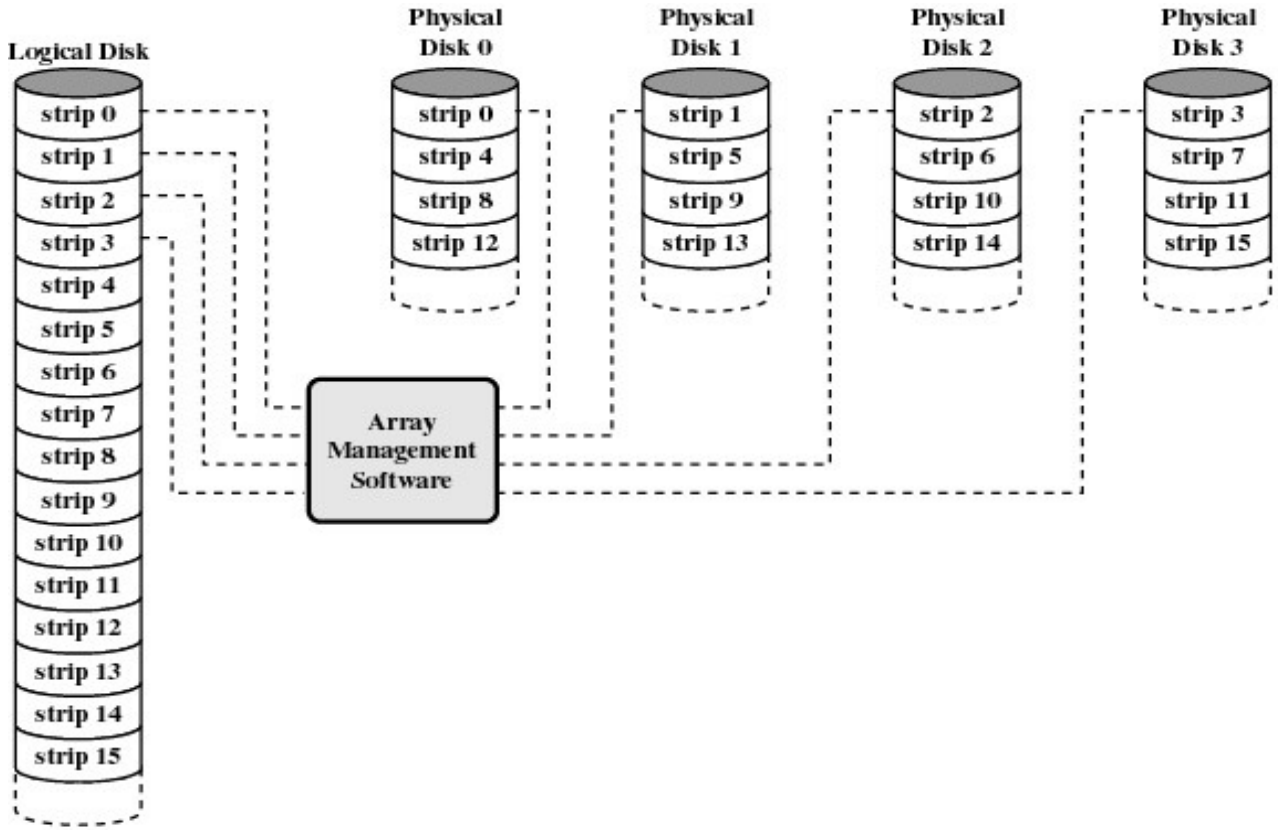
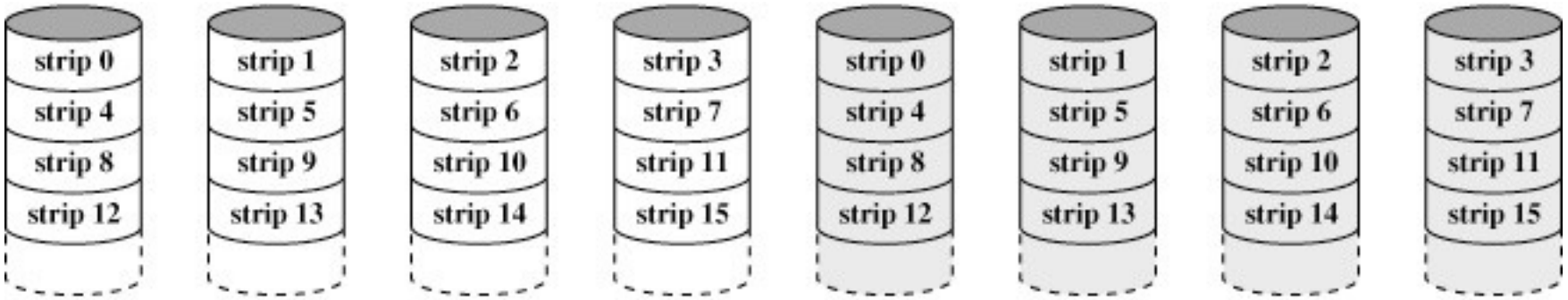


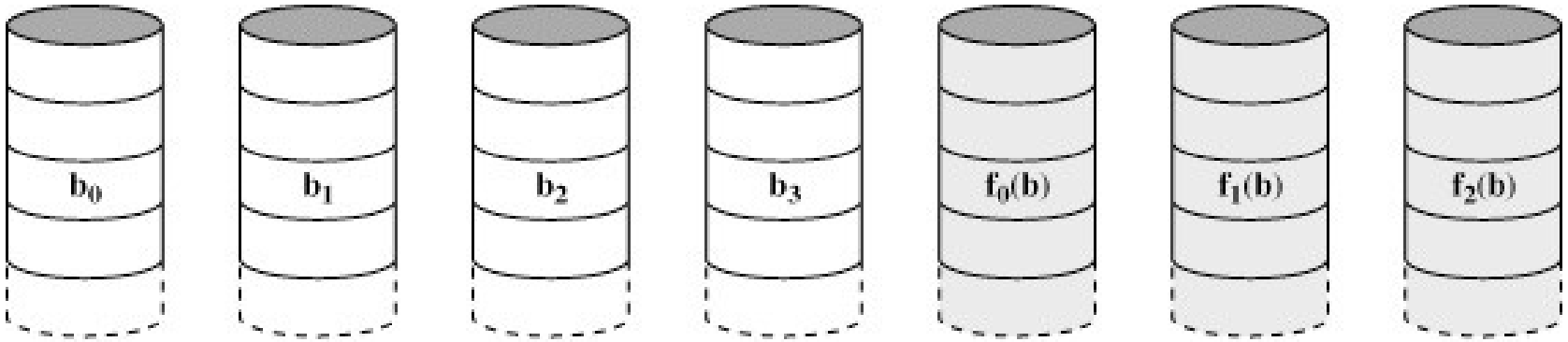
Figure 11.10 Data Mapping for a RAID Level 0 Array [MASS97]

RAID-1 (mirroir)



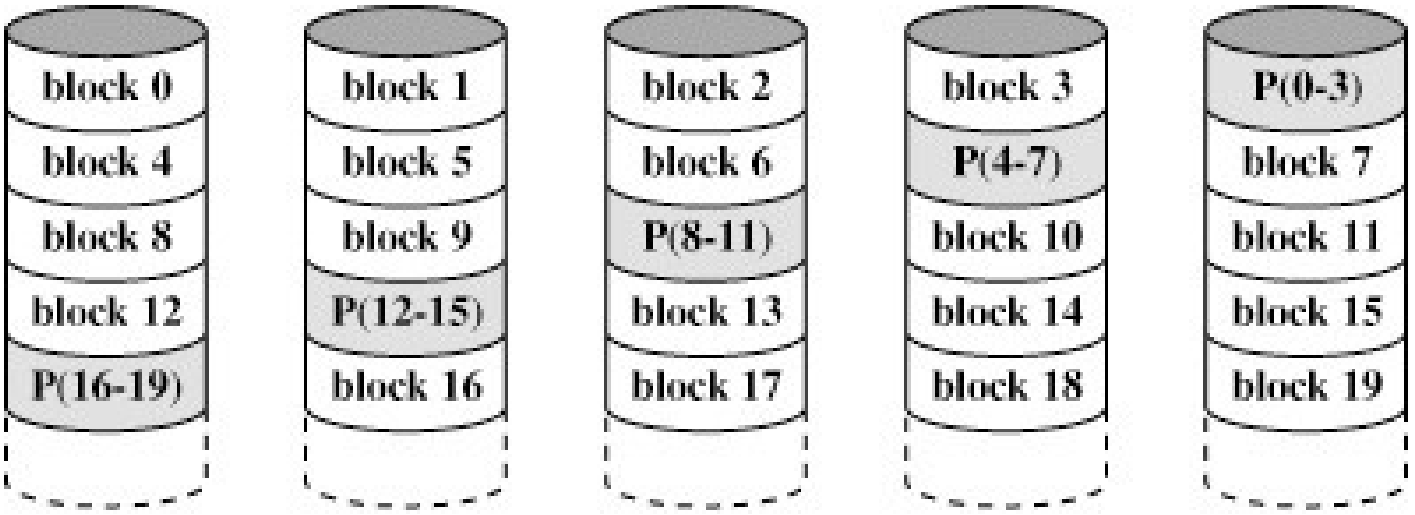
(b) RAID 1 (mirrored)

RAID-2 (redondance via code de Hamming)



(c) RAID 2 (redundancy through Hamming code)

RAID-5 (calcul parité+distribution de la comme de contrôle)



(f) RAID 5 (block-level distributed parity)

Et si un seul disque tombe en panne ?

Conclusion

http://en.wikipedia.org/wiki/Comparison_of_file_systems

… et les systèmes de fichiers pour lecteurs de CD, disques SSD, lecteurs de bandes... le noyau du système d'exploitation doit aussi pouvoir les gérer !