

Séance 8

Stockage de masse et système de fichiers



Ce(tte) œuvre est mise à disposition selon les termes de la Licence Creative Commons Attribution – Pas d'Utilisation Commerciale – Pas de Modification 4.0 International.

Rappels

- Utilisation du principe de **mémoire virtuelle**
 - Découpe de la mémoire logique en pages
 - Mécanisme de pagination à la demande et défaut de page
 - Comparaison d'algorithmes de remplacement de pages
- Méthodes d'**allocations des cadres** physiques aux processus
 - Méthodes d'allocation des cadres et comparaison
 - Écoulement et modèle des localités

Objectifs

- Notion de **fichiers** et organisation en répertoires
 - Propriétés et opérations sur les fichiers et répertoires
 - Structure des fichiers sur le disque
- Principes du **système de fichiers**
 - Structure, types et montage d'un système de fichiers
 - Implémentation d'un système de fichier et allocation disque
- **Stockage de masse** et structure du disque

Algorithmes d'ordonnancement et formatage de disques

A close-up, shallow depth-of-field photograph of a stack of colorful, textured files or papers. The colors transition from dark red at the top to yellow, green, blue, and finally white at the bottom. The texture of the paper is visible, showing creases and slight variations in color.

Fichier

Concept de fichier

- Stockage d'informations sur divers médias

Disque/bande magnétique, disque optique...

- Information abstraite en une vue uniforme et logique par l'OS

Unité de stockage logique mappé sur les dispositifs physiques

- Stockage permanent de fichiers

- Collection nommée d'informations sur stockage secondaire
- Plus petite unité de stockage pour l'utilisateur

Type de données

- Quatre **types d'informations** dans un fichier de données
 - Numérique
 - Alphabétique
 - Alphanumérique
 - Binaire
- **Structure** des fichiers libre ou rigide
 - Séquence de bits ou d'octets
 - Séquence de lignes
 - Séquence d'enregistrements

Type de fichiers

- Un fichier peut stocker **différents types d'information**

Donnée texte/numérique, photo, musique, image, vidéo...

- La **structure d'un fichier** dépend de son type

- Un **fichier texte** est une séquence de caractères

Organisation complémentaire en lignes et pages

- Un **fichier source** est une séquence de fonctions

Avec ensemble de déclarations et instructions exécutables

- Un **fichier exécutable** contient des sections de code

Le loader peut les charger en mémoire et les exécuter

Attribut des fichiers

- Fichier possède **un nom** permettant de l'identifier

Manière pratique pour l'utilisateur de se référer à un fichier

- Plusieurs **attributs communs** aux différents OS

- **Identificateur** unique non lisible par l'utilisateur
- **Type** lorsque plusieurs types supportés
- **Localisation** sur le périphérique
- **Taille** en octets, mots ou blocs et limite éventuelle
- **Droits** en lecture/écriture/exécution
- **Date+Heure et utilisateur** pour création/accès/modification

Infos sur ECAM-OS4MEOIN-2017-Cours8... 7,1 Mo

Modifié : aujourd'hui 11:56

Ajouter des tags...

▼ Général :

Type : Portable Document Format (PDF)
Taille : 7.082.609 octets (7,1 Mo sur disque)
Emplacement : /Users/combefis/Documents/Work/ECAM/
2017-2018/OS4MEOIN/Cours 8/Slides
Création : dimanche 13 septembre 2015 17:53
Modifié : aujourd'hui 11:56
 Modèle
 Verrouillé

▼ Plus d'infos :

Version : 1.5
Pages : 74
Résolution : 362 x 272
Sécurité : None
Créateur du contenu : LaTeX with Beamer class
Logiciel d'encodage : pdfTeX-1.40.17

▼ Nom et extension :

ECAM-OS4MEOIN-2017-Cours8-Slides.pdf

Masquer l'extension

► Commentaires :

► Ouvrir avec :

► Aperçu :

▼ Partage et permissions :

Lecture et écriture autorisées

Nom	Privilège
combefis (Moi)	↳ Lecture et écriture
staff	↳ Lecture seulement
everyone	↳ Lecture seulement

+ - ⚙ 🔒

Structure de répertoire

- Informations sur tous les fichiers dans le **répertoire**
Se trouve également sur le stockage secondaire
- Chaque **entrée** garde le nom et l'identificateur du fichier
L'identificateur donne ensuite accès aux attributs du fichier
- Peut occuper beaucoup de **place en mémoire**
Facilement plus d'un Kio par fichier

Opération de base sur les fichiers

- **Création** d'un nouveau fichier

Stocker le fichier et nouvelle entrée dans le répertoire

- **Écriture et lecture** du contenu d'un fichier (read/write)

Pointeur de lecture/écriture localise le lieu de l'opération

- **Déplacement** à l'intérieur du fichier (seek)

Ne nécessite pas d'opération disque

- **Suppression et troncature** d'un fichier

Suppression du fichier ou seulement de son contenu

Autres opérations

- Deux autres opérations de base possibles
 - Ajout à la fin d'un fichier (append)
 - Renommage d'un fichier (rename)
- Construction par combinaison d'opérations primitives

append = seek + write

- Récupération des attributs d'un fichier

Obtenir sa taille, changer le propriétaire...

Ouverture et fermeture d'un fichier

- Chaque opération doit savoir **sur quel fichier opérer**

Recherche des informations dans le répertoire (long !)

- L'OS maintient une **table de fichiers ouverts**

Un pointeur de fichier est l'indice dans cette table

- **Deux appels systèmes** pour pouvoir manipuler un fichier

- `open` ouvre le fichier avant de faire des opérations
 - `close` ferme le fichier une fois les opérations terminées

Table des fichiers ouverts (1)

- Stocke des **informations** sur tous les fichiers ouverts

Mise à jour lors de chaque appel système d'ouverture/fermeture

- **Deux niveaux** de tables pour gérer plusieurs processus

- Une table **au niveau du système**

Localisation sur disque, dates d'accès, taille...

- Une table **par processus**

Pointeur, statistiques, droits d'accès...

- L'appel open renvoie un **pointeur** vers une entrée de la table

La table par processus point sur la table du système

Table des fichiers ouverts (2)

- **Liens** entre les tables des fichiers ouverts

Celle par processus point sur celle au niveau du système

- Utilisation d'un **compteur d'ouverture** des fichiers

- Incrémenté à chaque open

Initialisé lors de l'ajout de l'entrée dans la table niveau système

- Décrémenté à chaque close

Suppression de la table niveau système lorsque compteur atteint 0

Lecture d'un fichier

```
1 #include <fcntl.h>
2 #include <unistd.h>
3
4 #define BUFSIZE 5
5
6 int main()
7 {
8     int file;
9     if ((file = open ("data.txt", O_RDONLY)) != -1)
10    {
11        char buffer[BUFSIZE];
12        ssize_t s;
13        while ((s = read (file, buffer, BUFSIZE)) != 0)
14        {
15            write (STDOUT_FILENO, buffer, s);
16        }
17        close (file);
18    }
19
20    return 0;
21 }
```

Verrous de fichiers

- Empêcher un fichier d'être ouvert par plusieurs processus

Verrou partagé ou exclusif, mandatory ou advisory

- Gestion de l'**acquisition des verrous**

- Acquisition concurrente par des processus avec **verrou partagé**
- Un seul processus à la fois peut l'acquérir avec **verrou exclusif**

- Deux types de **mécanisme de verrouillage**

- L'OS interdit l'accès dans le cas d'un **verrou mandatory**
- Le programmeur gère les vérifications avec un **verrou advisory**

Appel système flock

- Obtention d'un verrou avec l'**appel système flock**

Permet d'obtenir et de vérifier si un verrou est placé sur un fichier

- **Choix** entre un verrou partagé (LOCK_SH) ou exclusif (LOCK_EX)

Sous Linux, les verrous de fichier sont de type advisory

```
1 if (flock (file, LOCK_EX) == 0)
2 {
3     // ...
4 }
```

Type de fichiers

- Support possible pour différents **types de fichier**

L'OS peut offrir des services particuliers pour des types de fichier

- Utilisation d'une **extension ou d'un nombre magique**

Type de fichier	Extension usuelle	Fonction
Exécutable	exe, com, bin	Programme exécutable en langage machine
Objet	obj, o	Langage machine compilé, non lié
Code source	c, cc, java, perl, asm	Code source dans divers langages de programmation
Batch	bat, sh	Commandes pour l'interpréteur de commandes
Balisage	xml, html, tex	Données textuelles
Traitement de texte	xml, rtf, docx	Formats de traitements de texte
Bibliothèque	lib, a, so, dll	Librairies pour les programmeurs
Imprimer ou visualiser	gif, pdf, jpg	Fichier texte ou binaire pour afficher ou imprimer
Archive	rat, zip, tar	Fichiers regroupés en un seul fichier
Multimédia	mpeg, mov, mp3, mp4, avi	Fichier binaire audio ou vidéo

Méthodes d'accès

- Accès aux informations d'un fichier et placement en mémoire

Totalité ou uniquement partie bien définie du fichier

- Plusieurs méthodes d'accès existantes

- **Séquentiel** lit le fichier du début à la fin (`read next`)
- **Direct** lit un enregistrement précis (`read n`)
- Accès à un bloc en fonction d'un index du fichier



Structure sur disque

Structure du disque (1)

- Disque utilisé intégralement ou **partitionné**

Chaque partition peut avoir un système de fichier différent

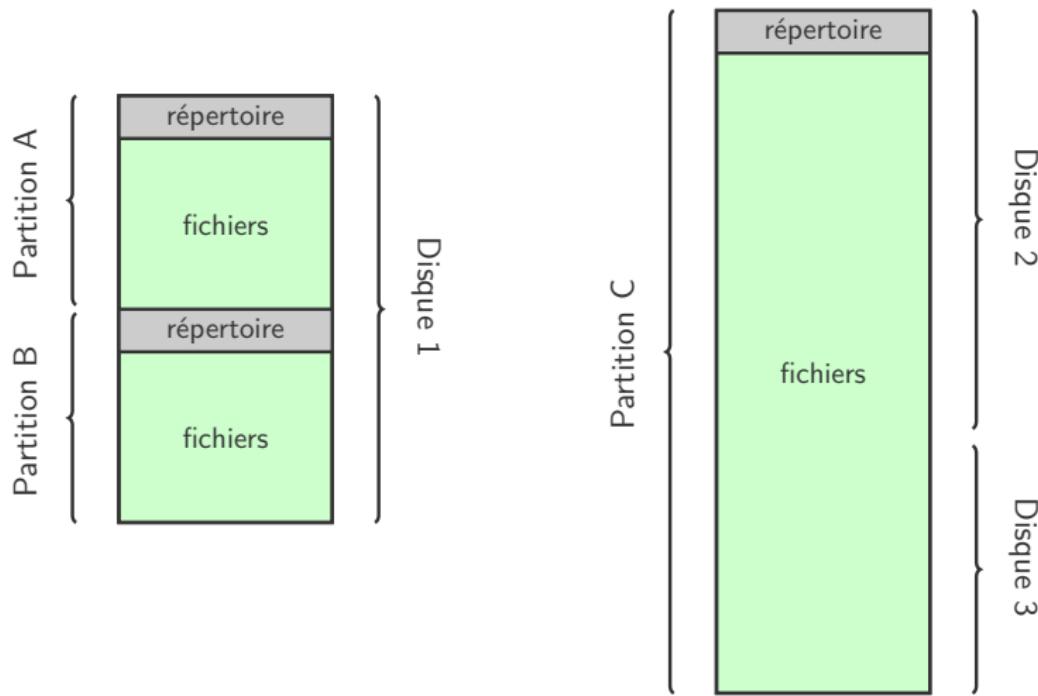
- Combinaison de plusieurs disques dans un **système RAID**

Construction d'un gros disque logique avec plusieurs physiques

- Un **volume** est une entité ayant un système de fichier

- Partie d'un disque, un disque, plusieurs disques...
- Peut être vu comme un disque virtuel
- Table des matières du volume (répertoire du périphérique)

Structure du disque (2)



Système de fichiers

- Une machine peut avoir **plusieurs systèmes de fichiers**

Focus sur les systèmes de fichiers d'usage général

- Exemple de systèmes de fichiers sur **Solaris**
 - **tmpfs**, temporaire, stocké en mémoire volatile
 - **objfs**, virtuel, interface vers le kernel pour les débuggeurs
 - **procfs**, virtuel, informations sur les processus
 - **ufs, zfs**, systèmes de fichiers d'usage général
 - ...

Opération sur les répertoires

- Un répertoire est une **table de symboles**

Mappe un nom de fichier vers l'entrée du répertoire

- Plusieurs opérations doivent être possibles sur un répertoire
 - **Chercher** un fichier dans un répertoire
 - **Créer ou supprimer** un fichier depuis un répertoire
 - **Lister** les fichiers faisant partie d'un répertoire
 - **Renommer** un fichier dans un répertoire
 - **Traverser** complètement la structure d'un répertoire

Structure des répertoires

- Plusieurs **choix de design** pour structurer les répertoires

Doit permettre une série d'opérations de base

- Plusieurs **critères** pour comparer les structures

- **Efficacité** des opérations (localiser un fichier rapidement)
- **Noms** des fichiers (plusieurs avec le même)
- **Groupement** logique des fichiers (jeux, programmes, photos...)

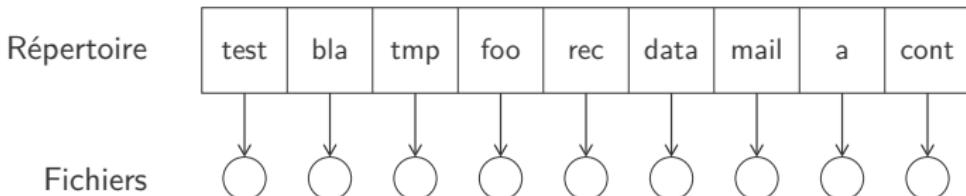
Répertoire à un seul niveau

- Tous les fichiers dans **le même répertoire**

Facile à supporter et à comprendre

- Plusieurs **contraintes** et limitations

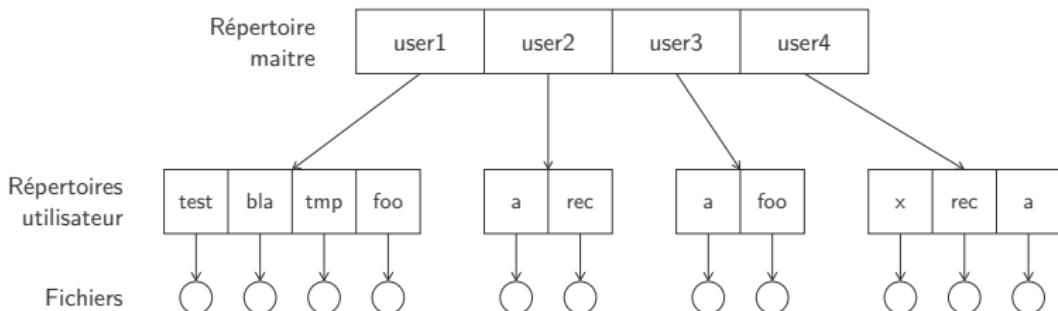
- Noms doivent être uniques
- Pas pratique pour faire des groupements (jouer avec noms)



Répertoire à deux niveaux

- Un répertoire par utilisateur (*User File Directory*, UFD)
 - Un répertoire maître qui liste les utilisateurs (MFD)
 - Pas de groupements, pas de partage entre utilisateurs
- Apparition de la notion de **chemin**

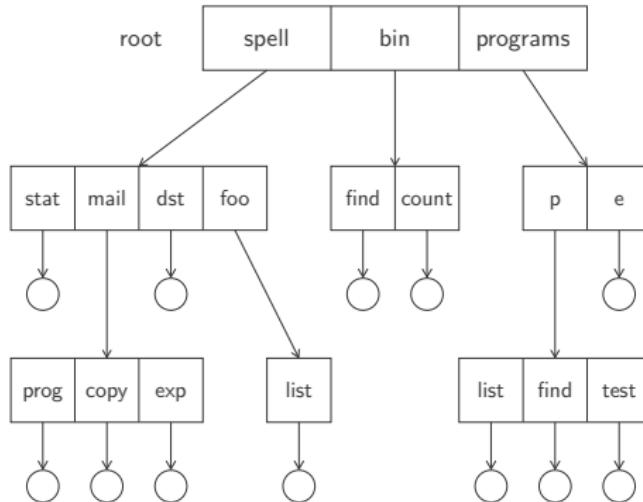
Difference entre /user2/rec et /user4/rec



Répertoire en arbre (1)

- Généralisation pour supporter **plusieurs niveaux**

Construction d'une structure d'arbre d'hauteur arbitraire



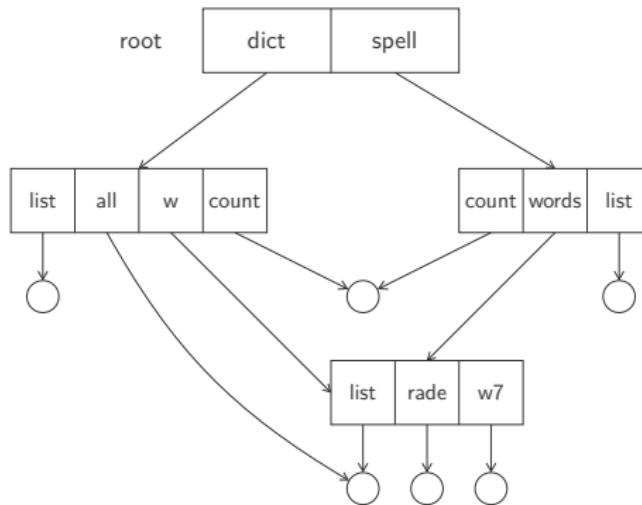
Répertoire en arbre (2)

- Répertoire **racine** de l'arbre (/)
 - Chemin unique pour chaque fichier
 - Un répertoire peut contenir des sous-répertoires
Identifié par un bit (0 pour fichier, 1 pour sous-répertoire)
- Chaque processus possède un **répertoire courant**
Appels systèmes pour se déplacer parmi les répertoires
- Chemins **relatifs et absolus** pour référencer un fichier/répertoire
Par rapport au répertoire courant ou à la racine

Répertoire en graphe acyclique (1)

- Graphe acyclique pour partager des fichiers entre répertoires

Pas de boucles pour éviter des parcours infinis



Répertoire en graphe acyclique (2)

- Structure suivant un **graphe sans cycles**
 - Plusieurs chemins pour certains fichiers (*alias*)
 - Possibilité de partager un répertoire
 - Sous Unix, lien symbolique est un pointeur vers un autre fichier
- **Complexité** de certaines opérations

Suppression, parcours... et éviter de visiter deux fois un partagé
- Maintenir le graphe **acyclique**

Interdire des répertoires ou des liens partagés

Système de fichiers



Montage d'un système de fichiers

- Il faut **monter** un système de fichiers avant utilisation

La structure du répertoire peut venir de plusieurs volumes

- Système de fichier monté sur un **point de montage**

Souvent un répertoire vide qui devient racine du système monté

- **Driver OS** pour lire et comprendre le système monté

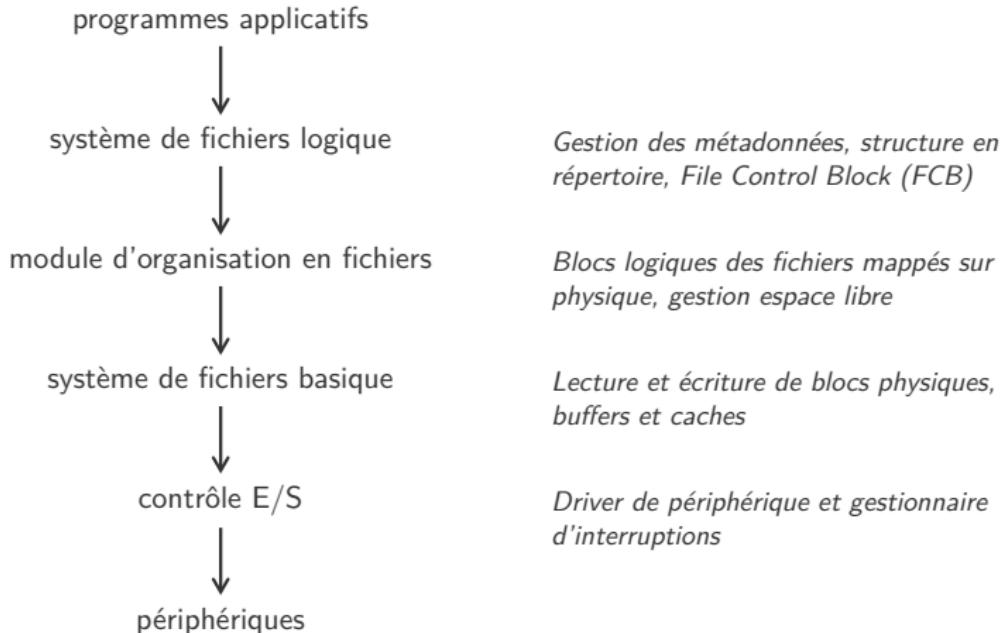
Lecture seule ou lecture/écriture selon le système monté

Structure des systèmes de fichiers

- Avantages de stocker systèmes de fichier sur **disques**
 - Peut être réécrit sur place
On peut lire un bloc, le modifier, puis le réécrire à la même place
 - Accès direct à n'importe quel bloc d'informations sur le disque
- Transfert d'informations du disque à la mémoire **par blocs**
Amélioration des performances d'E/S
- Un bloc est composé de plusieurs **secteurs**
Taille d'un secteur entre 32 et 4096 octets (usuelle 512)

Organisation en couches

- Mapper vue logique utilisateur au stockage physique (disque)



Systèmes de fichiers

- La structure en couches permet de **réutiliser du code**

Plusieurs systèmes de fichiers différents par dessus contrôle E/S

- Exemples de **systèmes de fichiers** répandus

- **CD-ROMs** : ISO 9660
- **Unix** : UFS (*Unix File System*)
- **Windows** : FAT, FAT32, NTFS (*Windows NT File System*)
- **Linux** : ext2, ext3, ext4 (*Extended File System*)
- **macOS** : HFS (*Hierarchical File System*)

Implémentation (1)

- Structure **sur disque** similaire sur les différents OS

- **Boot control block** (par volume)

Démarrage de l'OS, souvent premier bloc d'un volume

- **Volume control block** (par volume)

Nombre de blocs, taille des blocs, blocs libres (nombre/pointeurs)

- **Structure de répertoire** (par système de fichiers)

Organise les fichiers

- **File control block** (par fichier)

Détails sur le fichier et identificateur unique

Implémentation (2)

- Structure **en mémoire** pour un système de fichier monté
 - **Mount table**
Informations sur volumes montés
 - **Cache de structure de répertoire**
Informations sur les répertoires récemment accédés
 - **Table des fichiers ouverts** (système)
Copie des FCB de chaque fichier ouvert
 - **Table des fichiers ouverts** (processus)
Pointeur vers la table des fichiers ouverts
- **Mémoire tampon** pour les blocs lus depuis le disque

File Control Block (FCB)

- **Création d'un nouveau fichier** par système de fichiers logique

Allocation d'un nouveau FCB, mise à jour du répertoire

- **Répertoire** est un fichier spécial sous Unix

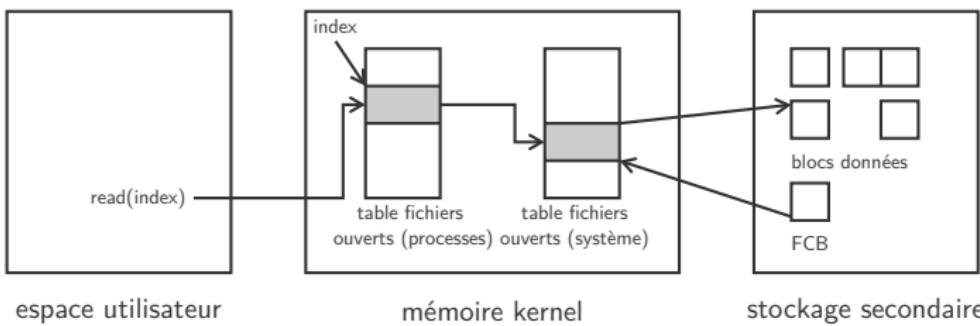
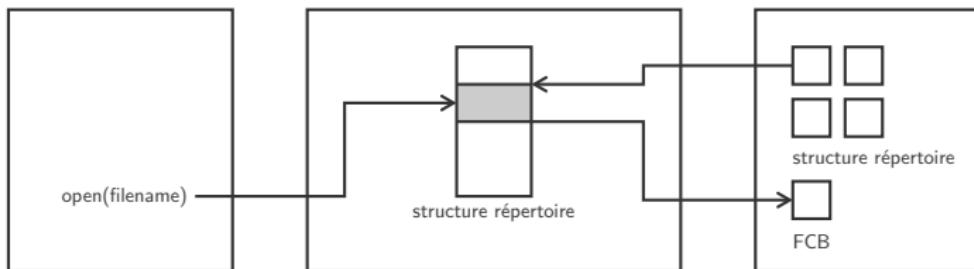
Appels systèmes différents sous Windows

permissions
dates de création, accès, modification
propriétaire, group, ACL
taille du fichier
blocs de données ou pointeurs vers de tels blocs

Ouverture de fichier

- Plusieurs étapes lors de l'**appel système open**
 - 1 Nom du fichier passé au système de fichier logique
 - 2 Recherche dans la *table des fichiers ouverts* (système)
Si trouvé, aller directement en 5
 - 3 Recherche dans structure de répertoire
 - 4 Copie du FCB dans la table des fichiers ouverts (système)
 - 5 Création pointeur dans table fichiers ouverts (processus)
 - 6 Renvoi d'un pointeur vers cette dernière table (file descriptor)
- **Autres informations** stockées dans tables des fichiers ouverts
Compteur de processus, mode d'accès, position dans le fichier...

Ouverture/lecture d'un fichier



Fermeture de fichier

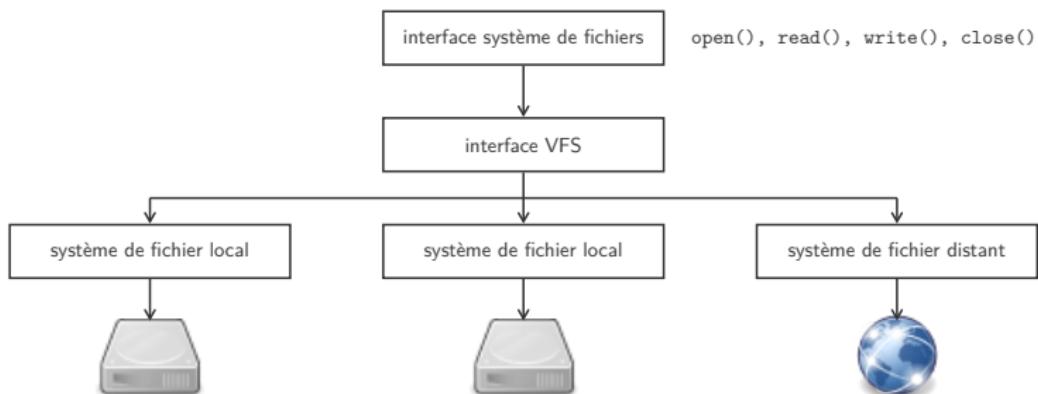
- Plusieurs étapes lors de l'**appel système close**
 - 1 Suppression entrée dans table fichiers ouverts (processus)
 - 2 Décrémentation compteur dans table fichiers ouverts (système)
Si compteur > 0, fin de la procédure
 - 3 Copie métadonnées modifiées vers disque (structure répertoire)
Car traitement en cache de la structure de répertoire
 - 4 Suppression entrée dans table fichiers ouverts (système)
- Autre **fichiers spéciaux**
Sous UFS, connexion réseaux gérées comme les fichiers...

Partitions et montage

- Partition “raw” (sans système de fichiers) ou “cooked” (avec)
La partition swap de Linux, par exemple, est « raw »
- Partition spéciale pour les informations de démarrage (boot)
Système au chargement pas capable interpréter système fichiers
- Partition **root** contient le kernel de l'OS monté au démarrage
Autres partitions montées au démarrage ou manuellement
- **Montage** d'une partition
 - **Windows** : espace de nommage séparé (F :, E :...)
 - **Unix** : montage sur n'importe quel répertoire

Système de fichiers virtuel (1)

- Support de **plusieurs systèmes de fichiers** différents
 - À intégrer dans la structure de répertoire
- Utilisation d'une interface **Virtual File System** (VFS)



Système de fichiers virtuel (2)

- **Séparation** opérations génériques/implémentations

Pour les opérations sur systèmes de fichiers

- Représenter de manière unique un **fichier sur le réseau**

*Utilisation des **vnode** sous Unix*

- **Redirection transparente** des appels systèmes

*Vers opérations spécifiques ou le réseau avec **NFS***



Méthode d'allocation

Problème de l'allocation

- Plusieurs fichiers stockés sur le **même disque**

Espace alloué pour ces différents fichiers

- **Critères** de qualité d'une bonne méthode d'allocation

- Utilisation optimale de l'espace disque
- Accès rapide aux fichiers

- **Trois méthodes** d'allocation différentes

- Allocation contigüe, liée ou indexée
- Support des trois, mais un seul au sein d'un système de fichiers

Allocation contigüe

- Un fichier occupe un **ensemble de blocs contigus** sur le disque

Car les adresses sur le disque définissent une ordre linéaire

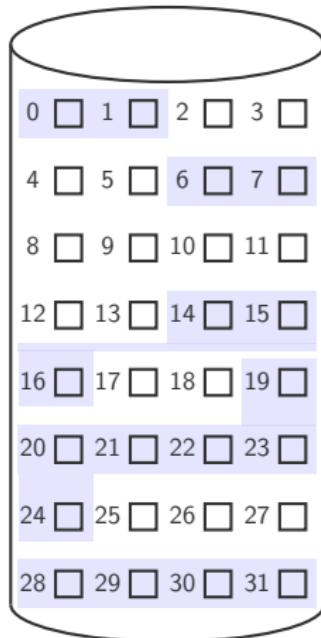
- Allocation définie par **adresse disque b** et **longueur n**

$b, b + 1, \dots, b + n - 1$ (avec n un nombre de blocs)

- **Accès** séquentiel ou direct à un fichier possible

Mémorisation du dernier bloc accédé, ou accès direct en $b + i$

Exemple de l'allocation contigüe



file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Problèmes de l'allocation contigüe

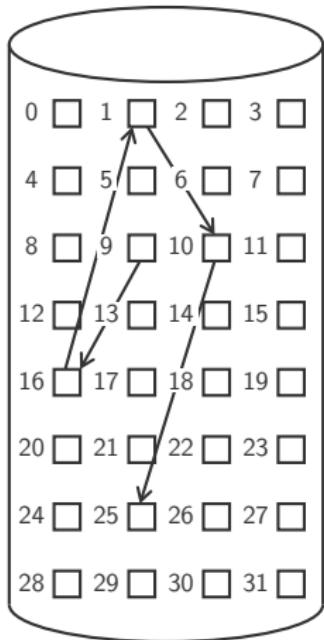
- Trouver de l'espace libre pour un **nouveau fichier**
 - Toujours plus lent que pour les autres méthodes d'allocation
 - Stratégies *first-fit* et *best-fit* toujours mieux que *worst-fit*
- Apparition de **fragmentation externe**

Espace disque découpé en petits morceaux ⇒ compactage
- Déterminer la **taille du fichier** pour prévoir assez de blocs
 - Difficile pour création nouveau fichier (déplacement)
 - Ou *extent* en stockant un lien vers le prochain bloc libre

Allocation liée

- Un fichier est une **liste chaînée de blocs** sur le disque
Un fichier peut être éparpillé sur le disque
- Allocation définie par le **premier et dernier** bloc
Chaque bloc contient un pointeur vers le bloc suivant
- Lecture et écriture de fichier se fait en **suivant les liens**
Pointeur null pour les fichiers vides

Exemple de l'allocation liée



file	start	end
jeep	9	25

Problèmes de l'allocation liée

- Résolution de la **fragmentation externe**

N'importe quel bloc libre peut être utilisé pour l'extension

- Accès **séquentiel** au fichier en suivant les liens

Accès direct pas possible sans suivre les liens

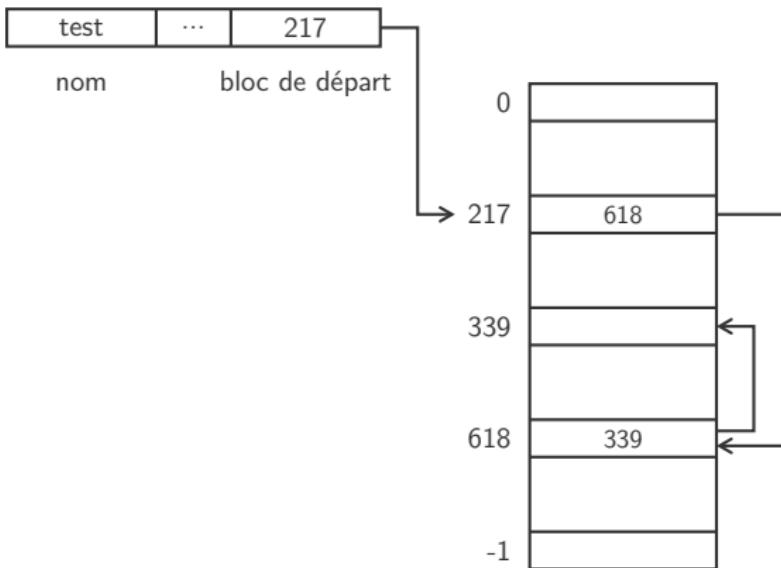
- Espace nécessaire pour **stocker les pointeurs**

- 4 o sur des blocs de 512 o représentent 0.78% du disque
 - Découpe du disque et allocations de clusters de blocs

- **Diminution de la fiabilité** en cas de perte de pointeurs

Utilisation d'une File-Allocation Table (FAT), à cacher

File-Allocation Table (FAT)



Allocation indexée

- Tous les pointeurs sont stockés dans l'**index block**

Un index block pour chaque fichier

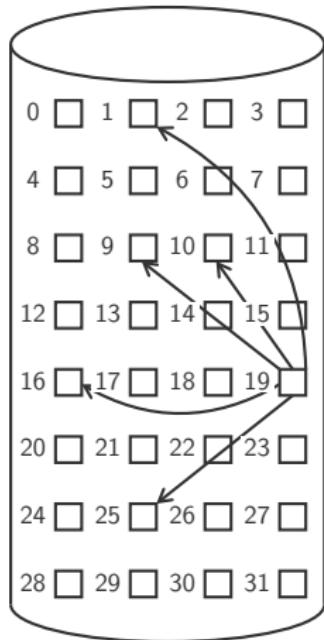
- Similaire au système de **pagination** avec sa table

Limitations dues à la taille d'un bloc

- Tous les pointeurs fixés à **null** lors de la création du fichier

Obtention d'un bloc libre sur le disque et ajout dans l'index block

Exemple de l'allocation indexée



file	index block
jeep	19

index block 19

9
16
1
10
25
-1
-1
-1

Problèmes de l'allocation indexée

- **Gaspillage** d'espace disque

Plus d'espace perdu pour les pointeurs que l'allocation liée

- **Taille** des index blocks limite la taille des fichiers

- **Linked scheme**

Un index block peut contenir un pointeur vers un autre

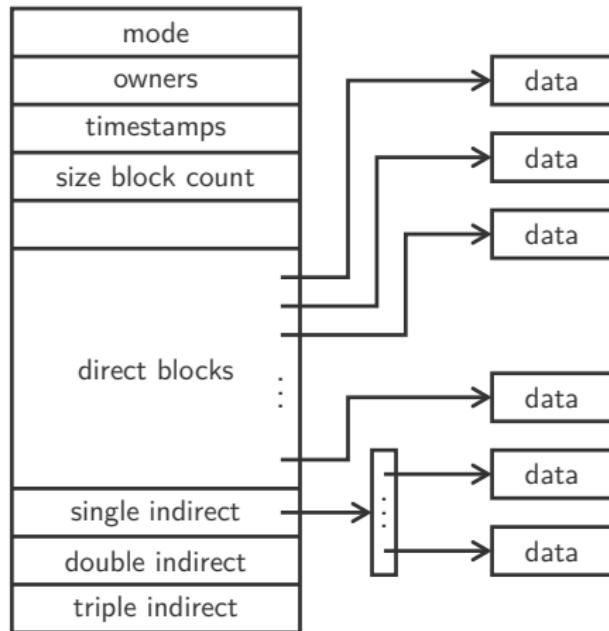
- **Multilevel index**

Indirection, l'index block pointe vers des sous-index blocks

- **Combined scheme**

*Combinaison des deux méthodes, exemple **inode** Unix*

Unix inode



Problème de la gestion de l'espace libre

- Réutiliser l'espace libéré suite à des suppressions

Si le disque n'est pas en lecture seule

- Utilisation d'une liste des espaces libres

Liste où de l'espace sera cherché pour les nouveaux fichiers

- Plusieurs implantations pour stocker les espaces libres

Bit vector, liste chaînée, space maps...

Stockage de masse



Disque magnétique (1)

- **Disque magnétique** utilisé comme stockage secondaire
- **Plateaux** circulaires recouverts d'un matériau magnétique

Sur les deux faces de chaque plateau

- **Têtes** de lecture/écriture flottent sur les plateaux (μm)

Toutes les têtes bougent ensemble comme une seule unité

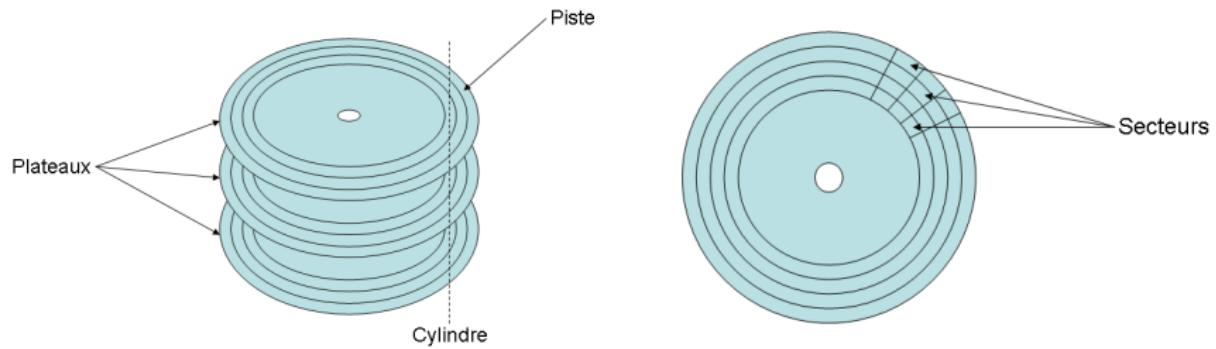
- Découpe des plateaux en **pistes** coupées en **secteurs**

L'ensemble des pistes des plateaux est appelé cylindre

Disque magnétique (2)



Disque magnétique (3)



Disque magnétique (4)

- Les disques **tournent** pendant que les têtes restent fixes

Vitesses de 5400, 7200, 10000 ou 15000 tours/minute

- **Deux vitesses** à prendre en compte

- Vitesse de transfert entre le disque et l'ordinateur (*Mio/s*)
- Positionnement des têtes et d'alignement du secteur (*ms*)

- **Crash** des têtes (déplacement, secousse, redémarrage...)

Ne peut normalement pas être réparé, disque à remplacer

- Connecté à la machine par un ensemble de fils appelé **bus E/S**

ATA, SATA, eSATA, USB, FC... gérés par un contrôleur

Solid-State Disk (SSD)

- Mémoire **non-volatile** utilisée comme un disque dur
- Plusieurs **technologies** utilisables
DRAM + batterie ou mémoire flash (SLC, MLC)
- Peuvent être plus **fiables et rapides**, moins de puissance
 - Pas de parties mobiles et accès aléatoire direct
 - Connexion à un bus plus rapide (voire le bus système PCI)
- Coutent **plus chers**, capacités plus faibles, vies plus courtes
Mais performance, légèreté, faible consommation d'énergie

Bandes magnétiques

- Plus longue durée de vie et grandes capacités de stockage

Accès aux données très lent, et accès aléatoire pas possible

- Utilisation pour des backups

Stockage d'information peu utilisée, transfert entre systèmes

- Bande rangée dans une bobine

Déroulée et rembobinée sous une tête de lecture/écriture

Structure du disque

- Addressage comme une séquence de **blocs logiques**

Le bloc logique est la plus petite unité de transfert

- Taille usuelle d'un bloc de **512 octets**

- Centaine de secteurs par piste sur le cylindre extérieur
 - Dizaine de milliers de cylindres par disque

- **Mapping** vers un numéro de cylindre, de piste et de secteur

Secteurs défectueux et nombre de secteurs par piste pas constant

- **Numérotation des blocs** du cylindre extérieur vers l'intérieur

Premier bloc sur secteur 0 (première piste cylindre extérieur)

Vitesse de rotation

- Vitesse linéaire constante (*Constant Linear Velocity, CLV*)
 - Densité de bits par piste uniforme
 - Plus grand nombre de secteurs à l'extérieur
 - La vitesse de rotation varie selon le cylindre lu
- Vitesse angulaire constante (*Constant Angular Velocity, CAV*)
 - Densité de bits par piste variable
 - Plus petit nombre de bits à l'extérieur
 - La vitesse de rotation est constante

Ordonnancement du disque



Ordonnancement du disque (1)

- L'OS doit utiliser le disque de **manière efficace**

Temps d'accès rapide et grande bande passante

- Deux composantes majeures au **temps d'accès** à un disque

- **Temps de recherche** pour se déplacer sur le bon cylindre
 - **Temps de latence** pour attendre le secteur voulu

- **Bandé passante** du disque mesure capacités de transfert

- Temps total entre première requête et complétion du transfert
 - Nombre total de bits transférés sur temps total

Ordonnancement du disque (2)

- Un processus fait un **appel système** pour accéder au disque
 - Opération d'entrée ou de sortie
 - Adresse sur le disque pour le transfert
 - Adresse mémoire pour le transfert
 - Le numéro des secteurs à transférer
- Requête satisfaite directement ou placée dans une **file**
Algorithme d'ordonnancement pour choisir les requêtes

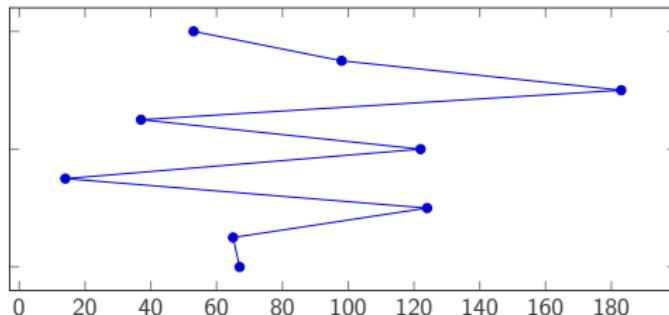
FCFS

- First-Come First-Served (FCFS) est équitable

Mais ne fournit pas le service le plus rapide

- Demande de blocs sur les cylindres suivants (tête en 53)

98, 183, 37, 122, 14, 124, 65, 67 (*déplacement de 640 cylindres*)



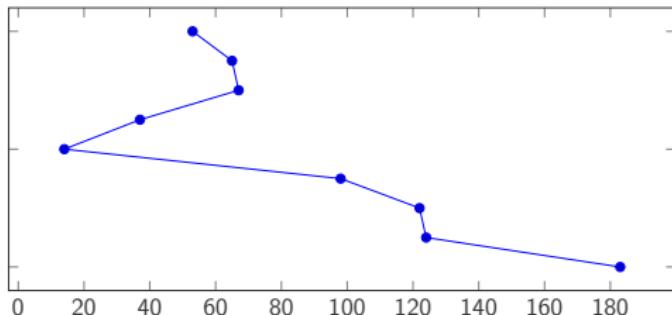
SSTF (1)

- Shortest-Seek-Time-First (SSTF)

Choisit le bloc le plus proche de la tête

- Demande de blocs sur les cylindres suivants (tête en 53)

98, 183, 37, 122, 14, 124, 65, 67 (*déplacement de 236 cylindres*)



SSTF (2)

- Fait beaucoup mieux que l'**algorithme FCFS**

Diminution globale du déplacement de la tête

- Similaire à l'**algorithme SJF**

- Même risque de famine possible avec SSTF
- Certaines requêtes pas servies rapidement

- **Pas de garantie d'optimalité** avec l'algorithme SSTF

53 → 37 → 14 puis 65, 67... (déplacement de 208 cylindres)

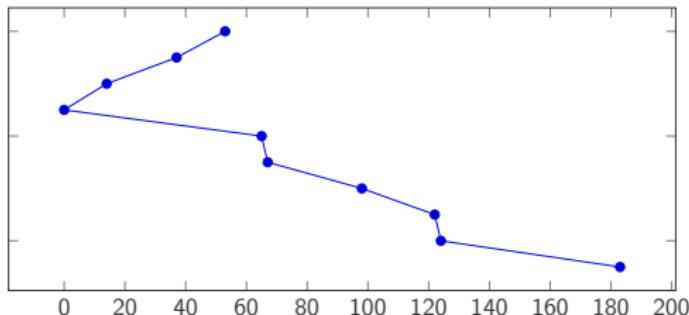
SCAN (1)

- La tête fait des **aller-retour** entre le début et la fin

Les blocs sont servis quand la tête passe dessus (escalateur)

- **Demande de blocs** sur les cylindres suivants (tête en 53 ↘)

98, 183, 37, 122, 14, 124, 65, 67 (*déplacement de 236 cylindres*)

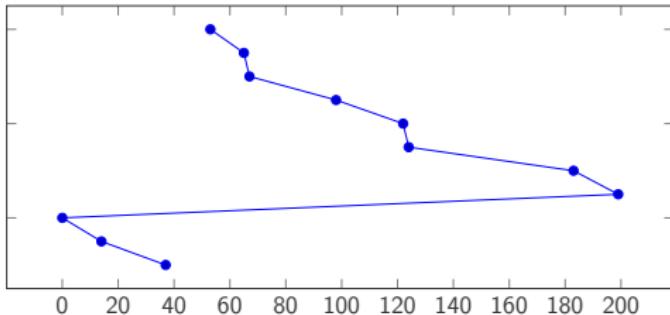


SCAN (2)

- Une **nouvelle requête** peut être vite servie ou non
 - Devant la tête, sera servie quasi immédiatement
 - Derrière la tête, devra attendre son retour
- Soit une **distribution uniforme** des requêtes sur les cylindres
 - Lors d'un retour, densité de requête de faible à grande
 - Les requêtes près de la tête ont été récemment servies

C-SCAN

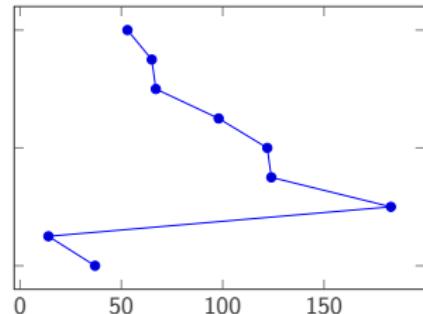
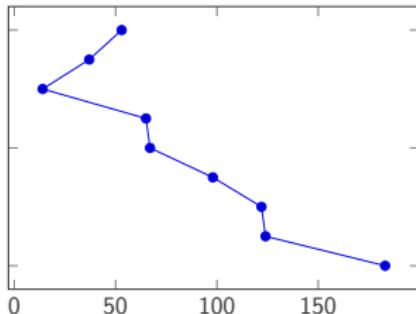
- Circular SCAN ne sert aucune requête au moment du retour
Car la file d'attente est souvent de l'autre côté du disque
- Demande de blocs sur les cylindres suivants (tête en 53 ↗)
98, 183, 37, 122, 14, 124, 65, 67 (*déplacement de 382 cylindres*)



LOOK et C-LOOK

- SCAN et C-SCAN bougent la tête sur la **largeur du disque**
Aucun algorithme ne fait cela en pratique
- Les variantes **LOOK et C-LOOK** ne vont pas jusqu'au bout

Déplacements respectifs de 208 et 322 cylindres



Choisir son algorithme

- **SSTF** est le plus commun

Donne de meilleures performances que FCFS

- **SCAN et C-SCAN** réduisent le risque de famine

Très utilisé sur système très dépendants du disque

- Performances dépendent du **nombre et des types** de requêtes

Une seule requête dans la file en moyenne, algos pareils

- Dépend aussi fortement de la méthode d'**allocation des fichiers**

SSTF et LOOK sont les deux algorithmes par défaut



Gestion de l'espace

Formatage de bas niveau

- Formatage de bas niveau (ou physique)

Division du disque en secteurs pour le contrôleur disque

- Disque rempli avec structure de données spéciale par secteur

- Entête (*header*), données (512 octets) et fin (*trailer*)
 - Entête contient numéro de secteur et ECC

- Formatage de base niveau souvent réalisé en usine

- Mapping entre numéro de blocs logique et secteurs sans défaut
 - Choix de la taille des données (256, 512 ou 1024)

Formatage logique

- Partition du disque en groupes de cylindres par l'OS

Chaque partition est vue comme un disque séparé

- Formatage logique crée le système de fichiers

Stockage de structures (dossier vide, liste des zones libres...)

- Regroupement de blocs en clusters

E/S disque par blocs et E/S système de fichiers par clusters

- Accès au disque comme un tableau de blocs

Raw I/O, par exemple utilisé par les bases de données

Bloc de démarrage

- Séquence de démarrage (bootstrap) est un programme simple
Initialise le système (registre, contrôleurs, mémoire centrale...)
- Démarrage de l'OS à partir du code du kernel en mémoire
Place le kernel en RAM et lance son exécution
- Programme de chargement du bootstrap se trouve en ROM
Bootstrap se trouve dans un bloc de démarrage sur le disque

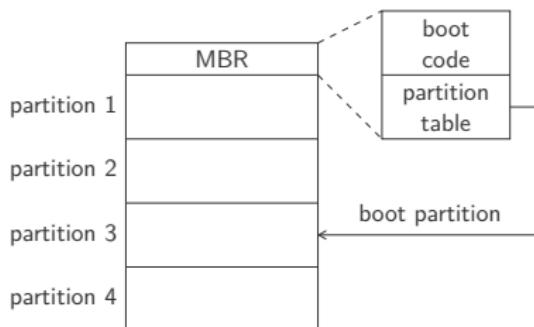
Exemple sous Windows

- Une **partition de démarrage**

Contient l'OS et les pilotes des périphériques

- Code de démarrage placé sur **master boot record** (MBR)

Premier secteur du disque dur



Bloc défectueux

- Disques **sujets à des défauts**

À cause des parties mobiles et faible tolérance aux pannes

- Défaillance totale ou seulement quelques **secteurs défectueux**

Déjà des blocs défectueux en sortie d'usine

- Gestion **manuelle** (contrôleur IDE)

Scan du disque et flag des blocs défectueux (badblocks sur Linux)

- Gestion **intelligente** par le contrôleur

Maintient d'une liste des blocs défectueux

Sector forwarding

- Analyse des secteurs pendant le formatage de bas niveau

Liste des blocs défectueux et de blocs de réserve

- Détection d'un secteur défectueux

- 1 L'OS demande à lire le bloc logique 87
- 2 Le contrôleur calcule l'ECC et signale un bloc défectueux
- 3 Au prochain reboot, le contrôleur remplace le bloc défectueux
- 4 Requête pour bloc logique 87 mappée sur un bloc de réserve

- Problème pour les algorithmes d'ordonnancement du disque

- 1 Secteur de réserve un peu partout pour chaque cylindre
- 2 Autre solution : le sector slipping déplace tous les blocs

Gestion de l'espace de swap

- Disque comme **extension** de la mémoire principale

Offrir le meilleur débit possible

- **Utilisation** de l'espace de swap

- Stocker tout un processus ou simplement une page
- De quelques Mio à plusieurs Gio

- **Localisation** de l'espace de swap

- Dans le système de fichiers, comme un gros fichier
- “*Raw partition*” gérée par un swap-space storage manager

Crédits

- https://www.flickr.com/photos/ted_major/4238539781
- <https://www.flickr.com/photos/flyingblogspot/15361704293>
- <https://www.flickr.com/photos/143304142@N07/27063135253>
- <https://openclipart.org/detail/34537/tango-drive-hard-disk-by-warszawianka>
- <https://openclipart.org/detail/35389/tango-applications-internet-by-warszawianka>
- https://www.flickr.com/photos/jon_scally/24120633319
- <https://www.flickr.com/photos/wwarby/11682625885>
- https://fr.wikipedia.org/wiki/Fichier:IBM_old_hdd_mod.jpg
- https://fr.wikipedia.org/wiki/Fichier:SEAGATE_36_go.jpg
- <https://fr.wikipedia.org/wiki/Fichier:Dd1.png>
- <https://fr.wikipedia.org/wiki/Fichier:Dd2.png>
- <https://www.flickr.com/photos/40563526@N04/6154179542>
- <https://www.flickr.com/photos/kvh/3599678171>