Module 9

Systèmes de fichiers – Chapitres 10 et 11 (Silberchatz)

Concepts importants du chapitre

- Systèmes fichiers
- Méthodes d'accès
- Structures Répertoires
- Protection
- Structures de systèmes fichiers
- Méthodes d'allocation
- Gestion de l'espace libre
- Implémentation de répertoires
- Questions d'efficacité

Que c'est qu'un fichier

- **Une collection d'informations apparentées, enregistrées dans un stockage secondaire**
  - De nature permanente
- **Similairement données qui se trouvent dans un stockage secondaires doivent être dans un fichier**
- **Sous différents types:**
  - Données (binaire, numérique, caractères….)
  - Programmes

Structures de fichiers

- Aucune – séquences d'octets…
- Texte: Lignes, pages, documents formatés
- Source: classes, méthodes, procédures…
- Etc.

Attributs d'un fichier

- **Représentent les propriétés du fichiers et sont stockés dans un fichier spécial appelé répertoire (directory). Exemples d'attributs:**
  - Nom:
    - Permet aux personnes d'accéder au fichier
  - Identificateur:
    - Un nombre permettant au SE d'identifier le fichier
  - Type:
    - Ex: binaire, ou texte; lorsque le SE supporte cela
  - Position:
    - Indique le disque et l'adresse du fichier sur disque
  - Taille:
    - En octets ou en blocs
  - Protection:
    - Détermine qui peut écrire, lire, exécuter…
  - Date:
    - pour la dernière modification, ou dernière utilisation
  - Autres…

Opérations sur les fichiers: de base

- **Création**
- **Écriture**
  - Pointeur d'écriture qui donne la position d'écriture
- **Lecture**
  - Pointeur de lecture
- **Positionnement dans un fichier (temps de recherche)**
- **Suppression d'un fichier**
  - Libération d'espace
- **Troncature: remise de la taille à zéro tout en conservant les attributs**

Autres opérations

- **Addition d'informations**
- **Renommage**
- **Copiage**
  - peut être fait en le renommant: deux noms pour un seul fichier
- **Ouverture d'un fichier: le fichier devient associé à un processus qui en garde les attributs, positions, etc.**
- **Fermeture**
- **Ouverture et fermeture peuvent être explicites (opératins: *open, close*)**
- **ou implicites**

Informations reliées à un fichier ouvert

- **Pointeurs de fichier**
  - Pour l'accéder séquentiellement
    - ex. pour read, write
- **Compteur d'ouvertures**
- **Emplacement**

Types de fichiers

- **Certains SE utilisent l'extension du nom du fichier pour identifier le type.**
  - Microsoft: Un fichier exécutable doit avoir l'extension .EXE, .COM, ou .BAT (sinon, le SE refusera de l'exécuter)
- **Le type n'est pas défini pour certains SE**
  - Unix: l'extension est utilisée (et reconnue) seulement par les applications
- **Pour certains SE le type est un attribut**
  - MAC-OS: le fichier a un attribut qui contient le nom du programme qui l'a généré (ex: document Word Perfect)

Types de fichiers

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm	binary file containing audio or A/V information

Structure logique des fichiers

- **Le type d'un fichier spécifie sa structure**
  - Le SE peut alors supporter les différentes structures correspondant aux types de fichiers
    - Cela complexifie le SE mais simplifie les applications
- **Généralement, un fichier est un ensemble d'enregistrements (records)**
  - Chaque enregistrement est constitué d'un ensemble de champs (fields)
    - Un champ peut être numérique ou chaîne de chars.
  - Les enregistrements sont de longueur fixe ou variable (tout dépendant du type du fichier)
- **Mais pour Unix, MS-DOS et autres, un fichier est simplement une suite d'octets « byte stream »**
  - Donc, 1 enregistrement = 1 octet
  - C'est l'application qui interprète le contenu et spécifie une structure
  - Plus versatile mais plus de travail pour le programmeur

Méthodes d'accès

Séquentiellement
Indexée Séquentiellement
Indexée
Directe

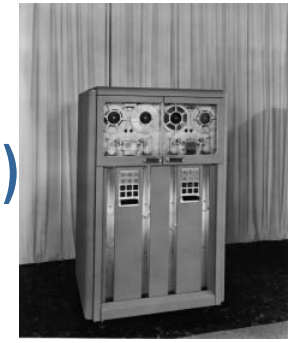
Méthodes d'accès: 4 de base

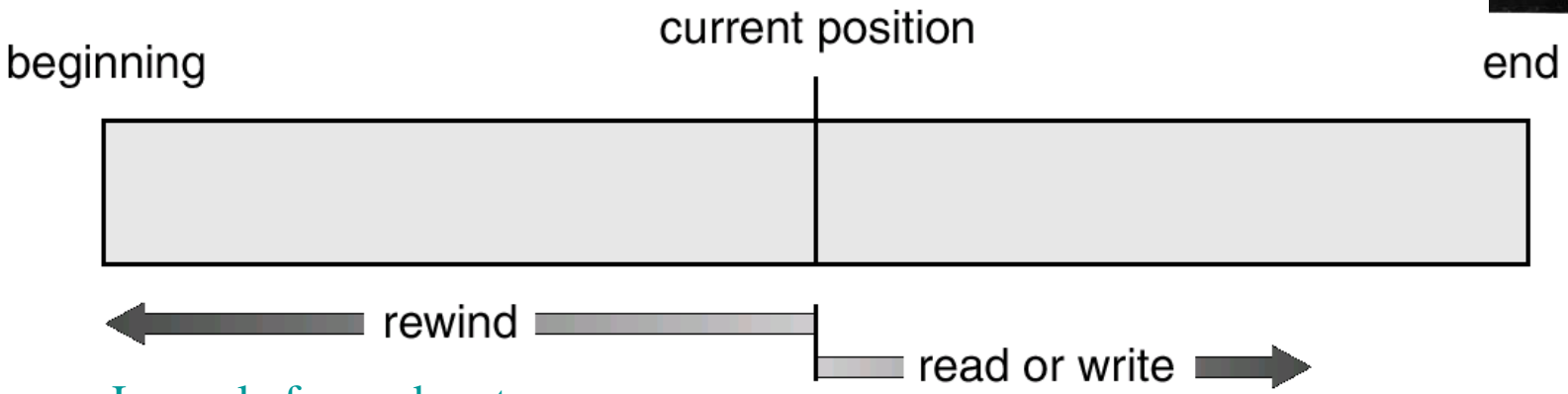
- **Séquentiellement (rubans ou disques): lecture ou écriture des enregistrements dans un ordre fixe**
- **Indexée séquentiellement (disques): accès séquentiel ou accès direct (aléatoire) par l'utilisation d'indexes**
- **Indexée: multiplicité d'indexes selon les besoins, accès direct par l'indexe**
- **Direct ou hachée: accès direct à travers un tableau d'hachage (une structure de données)**
- **Pas tous les SE supportent toutes les méthodes d'accès**
  - Quand le SE ne les supporte pas, c'est à l'application de les supporter

Méthodes d'accès aux fichiers

- **La structure logique d'un fichier détermine sa méthode d'accès**
- **Les SE sur les « mainframes » fournissent généralement plusieurs méthodes d'accès**
  - Car ils supportent plusieurs types de fichiers
- **Plusieurs SE modernes (Unix, Linux, MS-DOS…) fournissent une seule méthode d'accès (séquentielle) car les fichiers sont tous du même type (ex: séquence d'octets)**
  - Mais leur méthode d'allocation de fichiers (voir + loin) permet habituellement aux applications d'accéder aux fichiers de différentes manières
- **Ex: les systèmes de gestions de bases de données (DBMS) requièrent des méthodes d'accès plus efficaces que juste séquentielle**
  - Un DBMS sur un « mainframe » peut utiliser une structure fournie par le SE pour un accès efficace aux enregistrements.
  - Un DBMS sur un SE qui ne fournit qu'un accès séquentiel doit donc « ajouter » une structure aux fichiers de bases de données pour accès directs plus rapides.

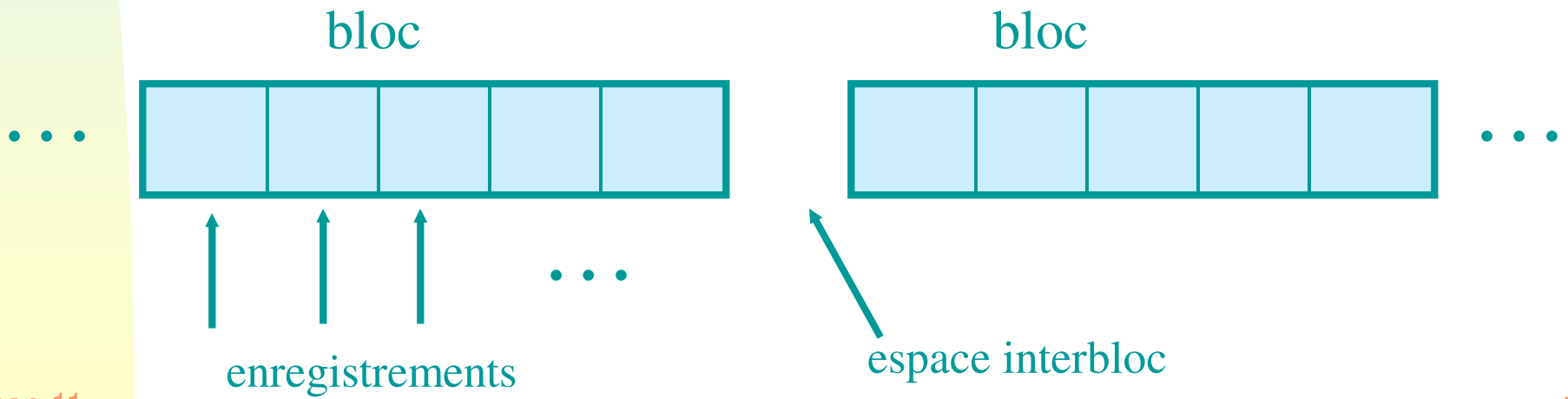
Fichiers à accès séquentiel (archétype: rubans)





La seule façon de retourner en arrière est de retourner au début (rébobiner, rewind)

En avant seulement, 1 seul enreg. à la fois



Lecture physique et lecture logique dans un fichier séquentiel

- **Un fichier séquentiel consiste en bloques d'octets enregistrés sur un support tel que ruban, disque…**
- **La dimension de ces bloques est dictée par les caractéristiques du support**
- **Ces blocs sont lus (lecture physique) dans un tampon en mémoire**
- **Un bloc contient un certain nombre d'enregistrements (records) qui sont des unités d'information logiques pour l'application (un étudiant, un client, un produit…)**
  - Souvent de longueur et contenu uniformes
  - Triés par une *clé,* normalement un code (code d'étudiant, numéro produit…)
- **Une lecture dans un programme lit le prochain enregistrement**
- **Cette lecture peut être réalisée par**
  - La simple mise à jour d'un pointeur si la lecture logique précédente ne s'est pas rendue à la fin du tampon
  - la lecture du prochain bloque (dans un tampon d'E/S en mémoire) si la lecture logique précédente s'est rendue à la fin du tampon
    - Dans ce cas le pointeur est remis à 0

Autres propriétés des fichiers séquentiels

- **Pour l'écriture, même idée: une instruction d'écriture dans un programme cause l'addition d'un enregistrement dans un tampon, quand le tampon est plein il y a une écriture de bloque**

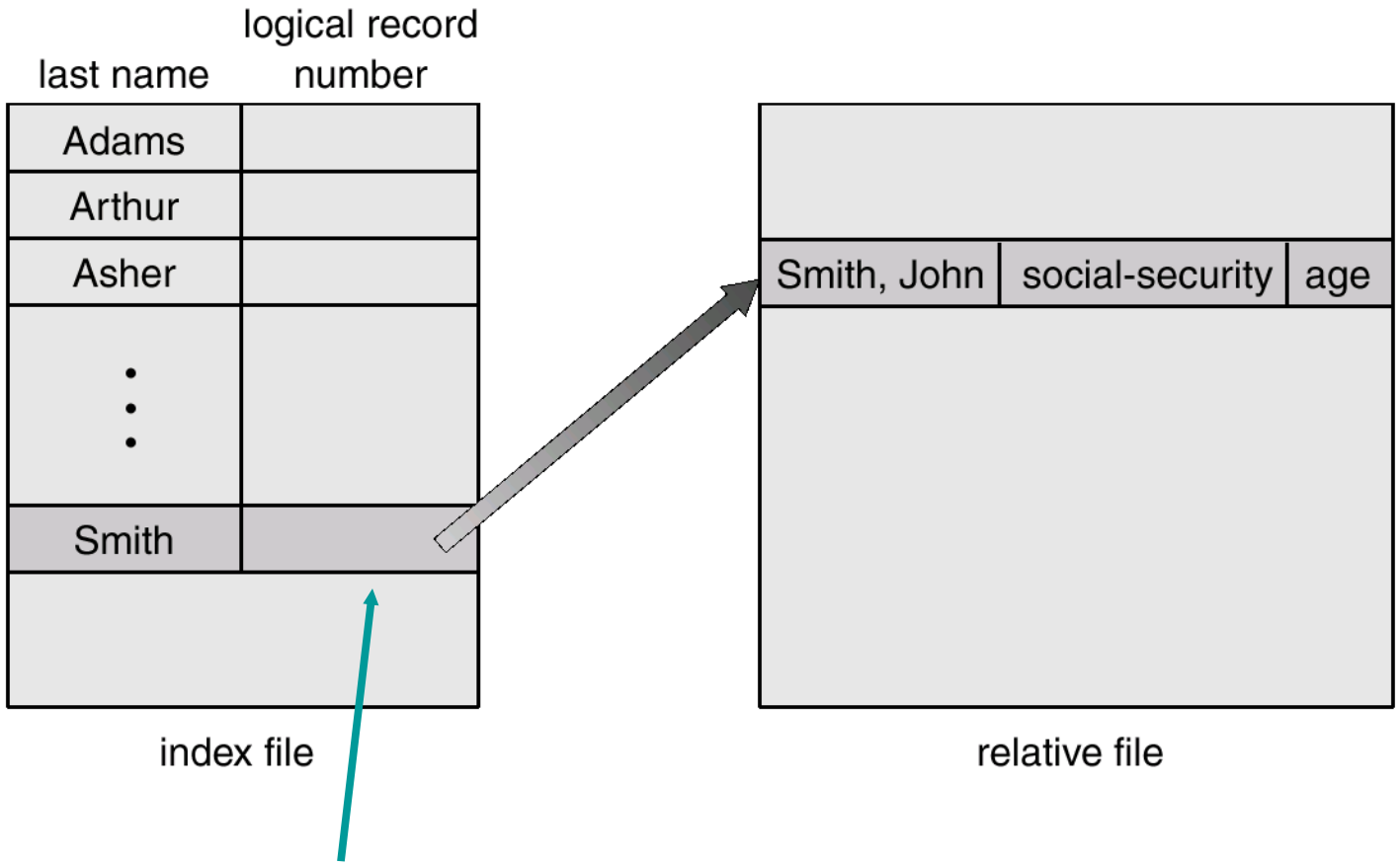
- **Un fichier séquentiel ouvert en mode de lecture ne peut pas être écrit et vice-versa (impossible de mélanger lectures et écritures)**

- **Les fichiers séquentiels ne peuvent être lus ou écrits qu'un enregistrement à la fois et seulement dans la direction 'en avant' (en aval)**

Adressage Indexé séquentiellement (index sequential)

- **Un index permet d'arriver directement à l'enregistrement désiré, ou en sa proximité**
  - Chaque enregistrement contient un champ clé
  - Un fichier indexe contient des repères (pointeurs) à certains points importants du fichier principal (ex. début de la lettre S, début des Lalande, début des matricules qui commencent par 8)
  - Le fichier indexe peut être organisé en niveaux (ex. dans l'indexe de S on trouve l'indexe de tous ceux qui commencent par S)
  - Le fichier indexe permet d'arriver à un point de repère dans le fichier principal, et la recherche est séquentielle

Exemples d'index et fichiers relatifs



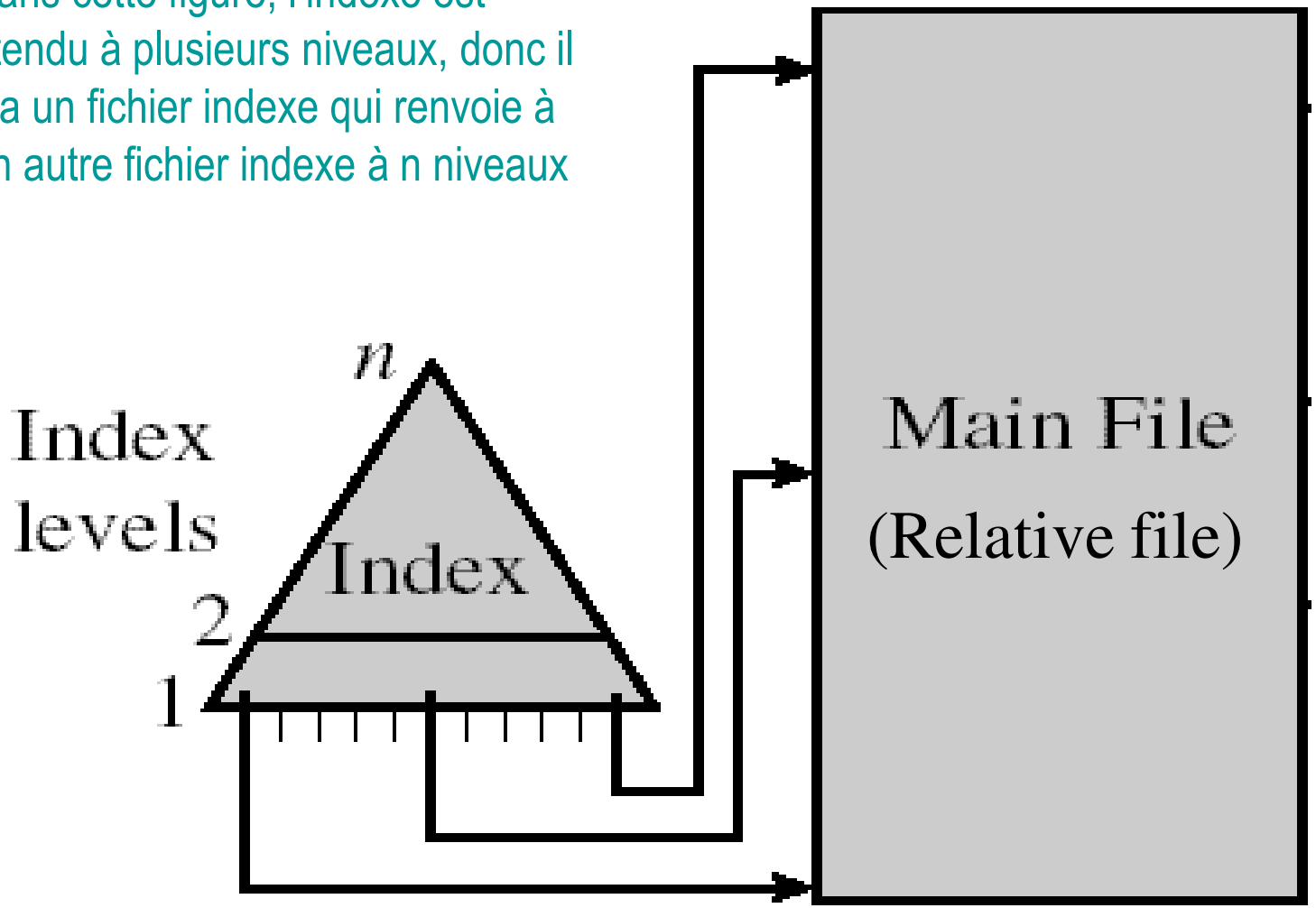
Pointe au début des Smiths (il y en aura plusieurs)

Le fichier indexe est à accès direct, le fichier relatif est à accès séquentiel

Accès direct: voir ci-dessous

Indexage et fichier principal (Stallings)

Dans cette figure, l'indexe est étendu à plusieurs niveaux, donc il y a un fichier indexe qui renvoie à un autre fichier indexe à n niveaux



Pourquoi plusieurs niveaux d'indexes

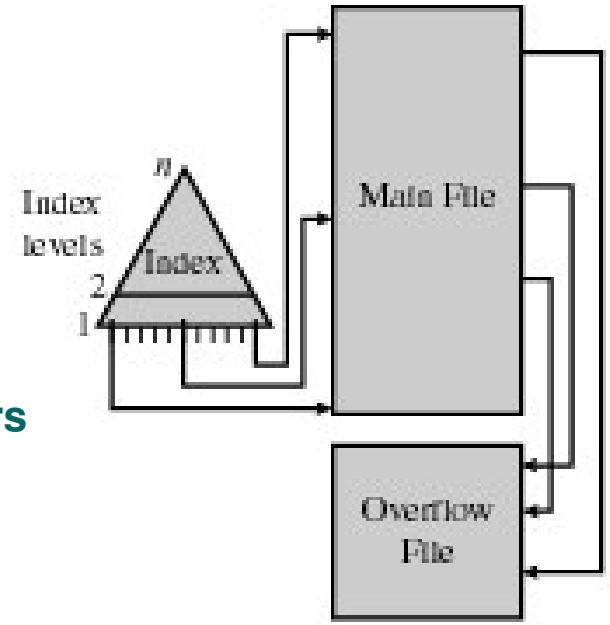
- Un premier niveau d'indexe pourrait conduire au début de la lettre S
- Un deuxième niveau au début des Smith, etc.
- Donc dans le cas de fichiers volumineux plusieurs niveaux pourraient être justifiés.

Séquentiel et indexage séquentiel: comparaison

- **Ex. Un fichier contient 1 million d'enregistrements**
- **En moyenne, 500, 000 accès sont nécessaires pour trouver un enregistrement si l'accès est séquentiel!**
- **Mais dans un indexage séquentiel , s'il y a un seul niveau d'indices avec 1000 entrées (et chaque entrée pointe donc à 1000 autres),**
- ***En moyenne*, ça prend 500 accès pour trouver le repère approprié dans le fichier index**
- **Puis 500 accès pour trouver séquentiellement le bon enregistrement dans le fichier relatif**

Mais… besoin de fichier débordement

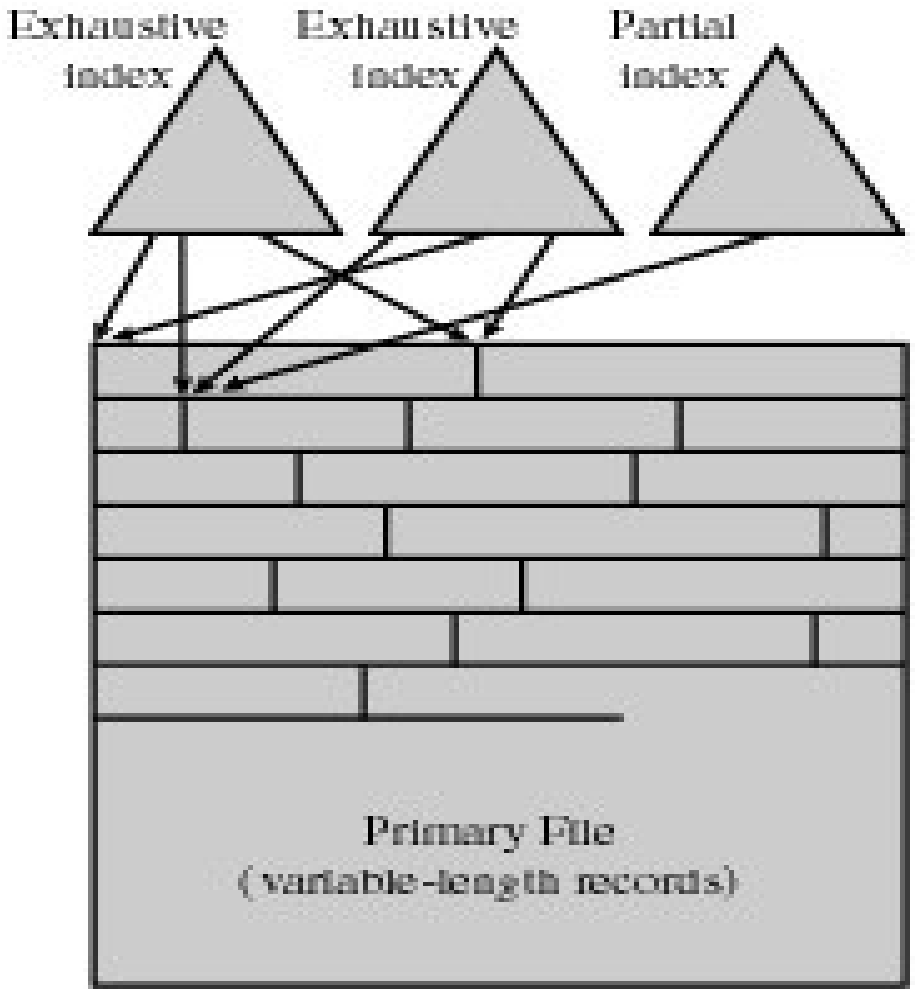
- Les nouveaux enregistrements seront ajoutés à un fichier de débordement
- Les enregistrements du fichier principal qui le précèdent dans l'ordre de triage seront mis à jour pour contenir un pointeur du nouvel enregistrement
  - Donc accès additionnels au fichiers de débordement
- Périodiquement, le fichier principal sera fusionné avec le fichier de débordement



Fichier indexé

- Utilise des indexe multiples pour différentes clés, selon les différents besoins de consultation
- Quelques uns pourraient être exhaustifs, quelques uns partiels, et sont organisés de différentes façons

Indexed File (Stallings)



Accès directe (ou relatif)

- **Le fichier est vu comme collection d'enregistrements logiques de grandeurs fixes**
  - Basé sur le modèle disque (composé de bloques)
  - Spécifie le numéro de bloque pour accéder les données
  - Numéro souvent relatif (du début du fichier)
- **Pas tous les SEs offrent les accès séquentiels et directes**
  - Facile de simuler l'accès séquentiel avec l'accès directe
    - Maintient un pointeur qui indique la position courante dans un fichier
  - L'inverse est très difficile

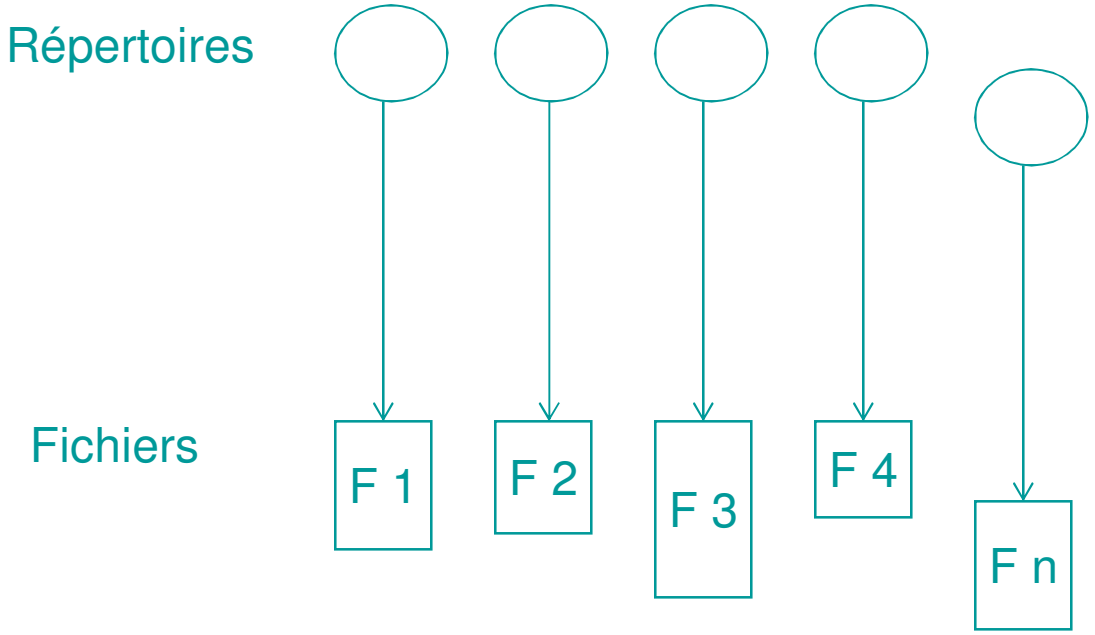
Utilisation des 4 méthodes

- Séquentiel (rubans ou disques): lecture ou écriture des enregistrements dans un ordre fixe
  - Pour travaux 'par lots': salaires, comptabilité périodique…
- Indexage séquentiel (disques): accès séquentiel ou accès direct par l'utilisation d'indexes
  - Pour les fichiers qui doivent être consultés parfois de façon séquentielle, parfois de façon directe (ex. par nom d'étudiant)
- Indexée: multiplicité d'indexes selon les besoins, accès direct par l'indexe
  - Pour les fichiers qui doivent être consultés de façon directe selon différents critères (ex. pouvoir accéder aux infos concernant les étudiants par la côte du cours auquel ils sont inscrits)
- Direct ou hachée: accès direct à travers tableau d'hachage
  - Pour fichiers qui doivent être consultés de façon directe par une clé uniforme (ex. accès aux information des étudiants par No. de matricule)

Répertoires

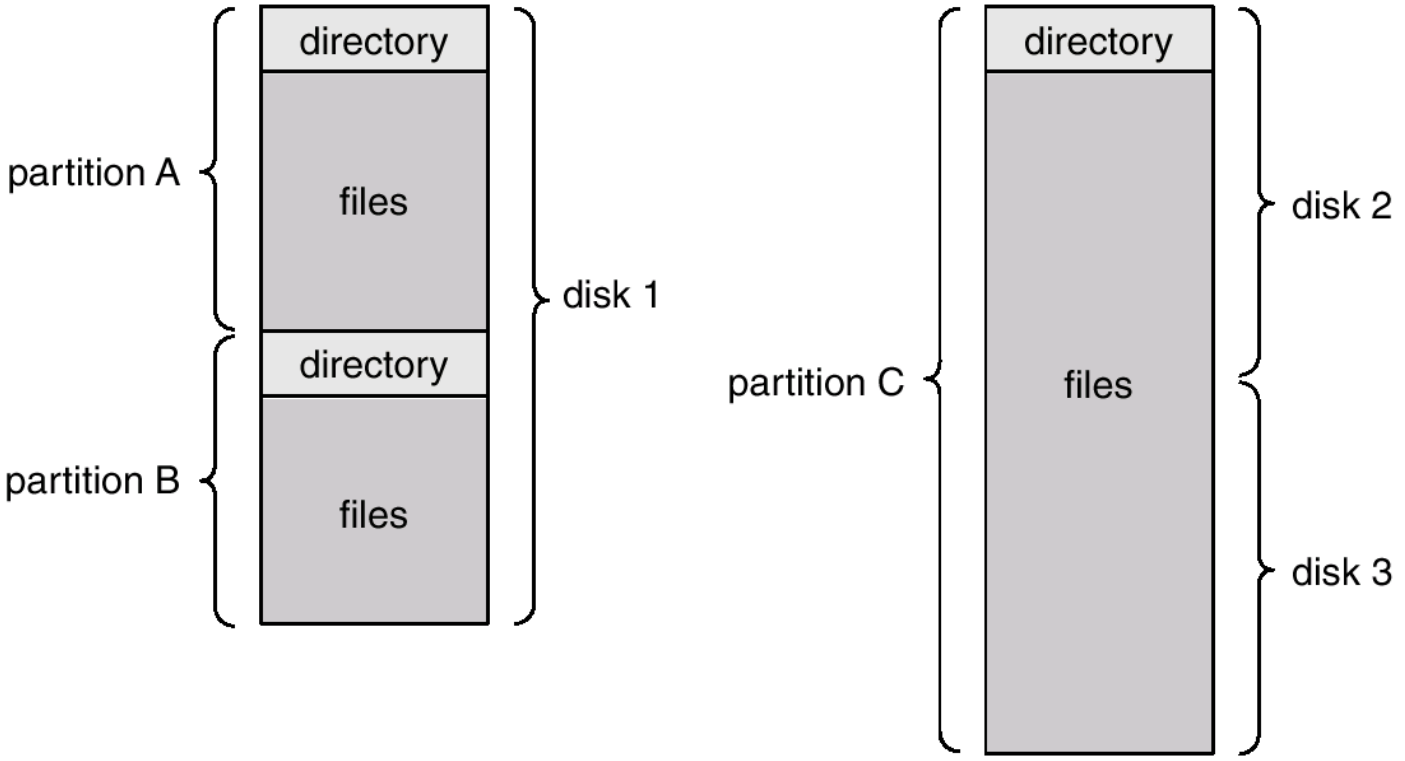
Structures de répertoires (directories)

- **Une collection de structures de données contenant des informations sur les fichiers.**



- **Les répertoires aussi bien que les fichiers, sont sur des disques**
- **À l'exception d'un répertoire racine en mémoire centrale**

Organisation typique de système de fichiers



Information dans un répertoire

- Nom du fichier
- Type
- Adresse sur disque, sur ruban...
- Longueur courante
- Longueur maximale
- Date de dernier accès
- Date de dernière mise à jour
- Propriétaire
- Protection

Opérations sur répertoires

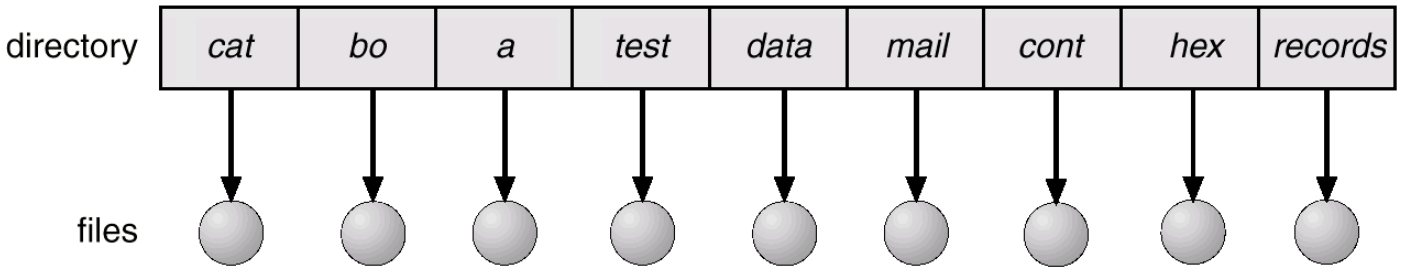
- Recherche de fichier
- Création de fichier
- Suppression de fichier
- Lister un répertoire
- Rénommer un fichier
- Traverser un système de fichier

Organisation de répertoires

- **Efficacité: arriver rapidement à un enregistrement**
- **Structure de noms: convenable pour usager**
  - deux usagers peuvent avoir le même noms pour des fichiers différents
  - Le même fichier peut avoir différents noms
- **Groupement de fichiers par type:**
  - tous les programmes Java
  - tous les programmes objet

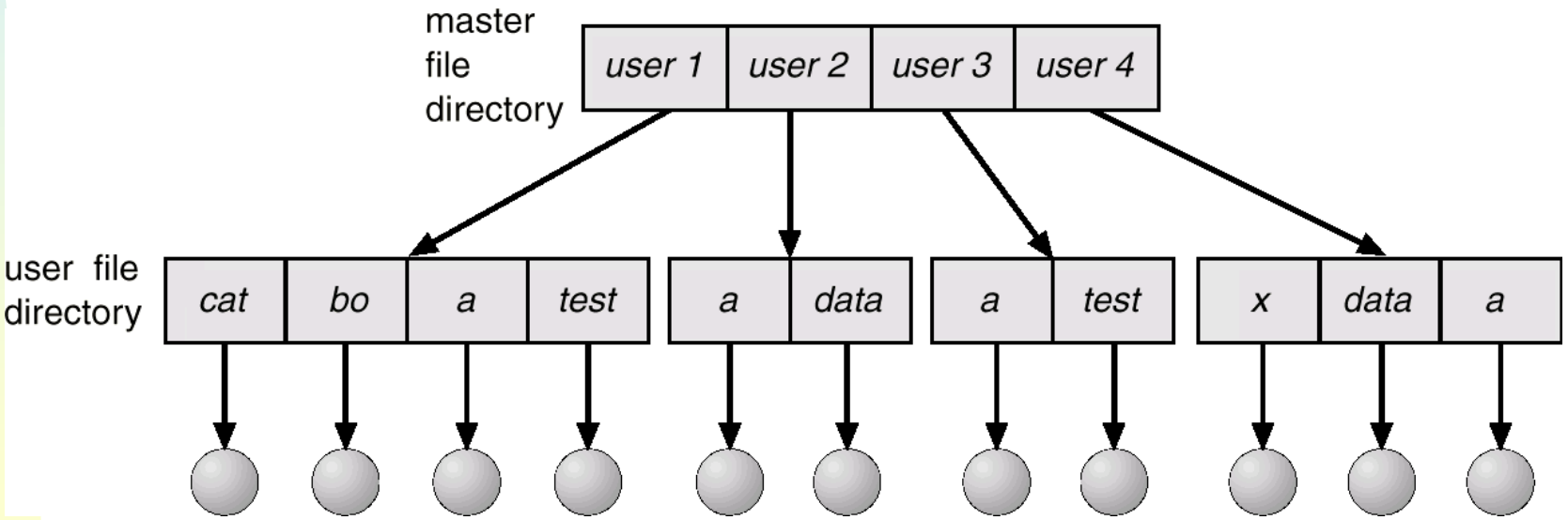
Structure à un niveau

- Un seul répertoire pour tous les usagers
- Ambiguïté de noms
- Problèmes de groupement
- Primitif, pas pratique

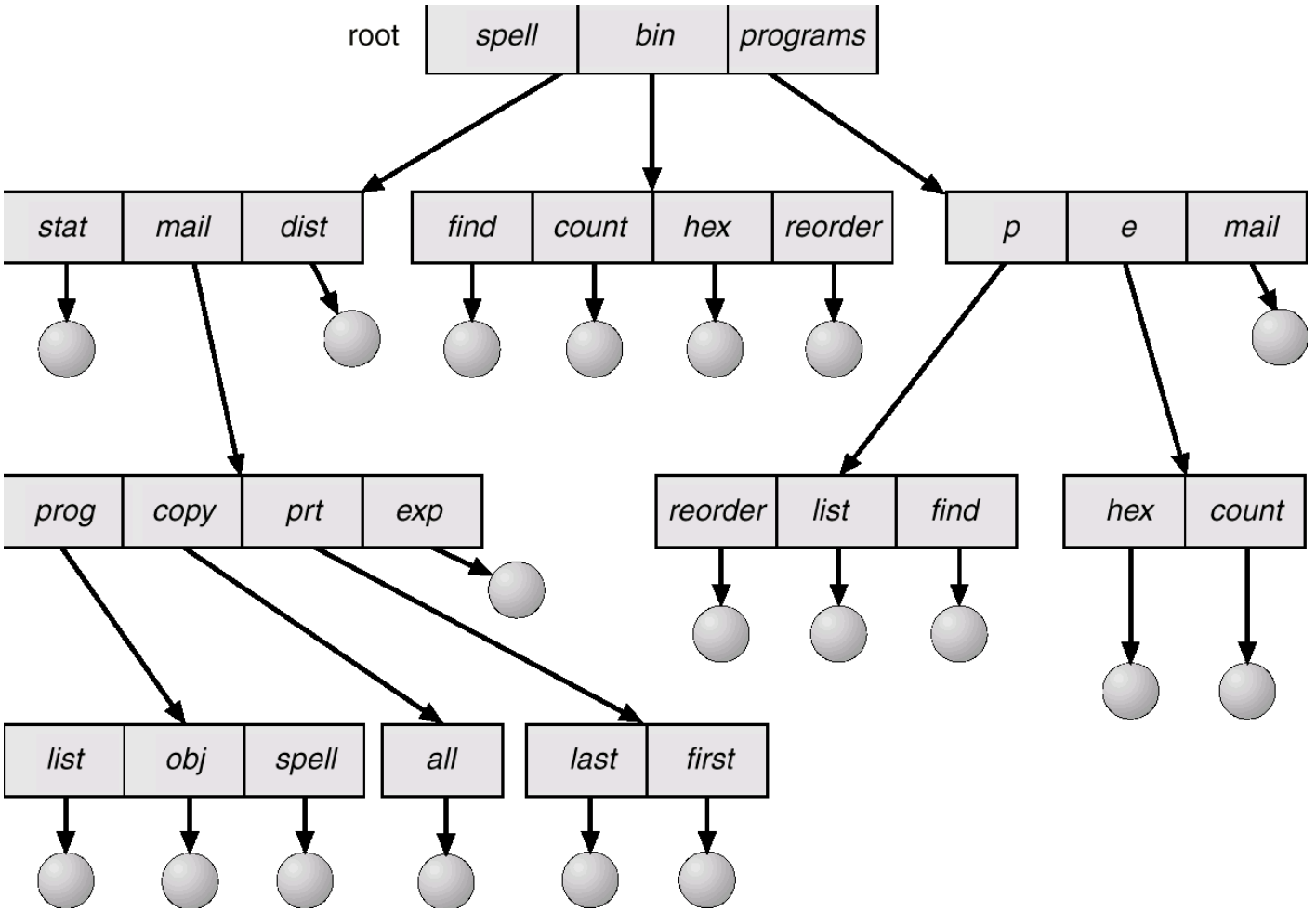


Répertoires à deux niveaux

- Répertoire séparé pour chaque usager
- `path name`, nom de chemin
- Le même nom de fichier pour différents usagers est permis
- Recherche efficace



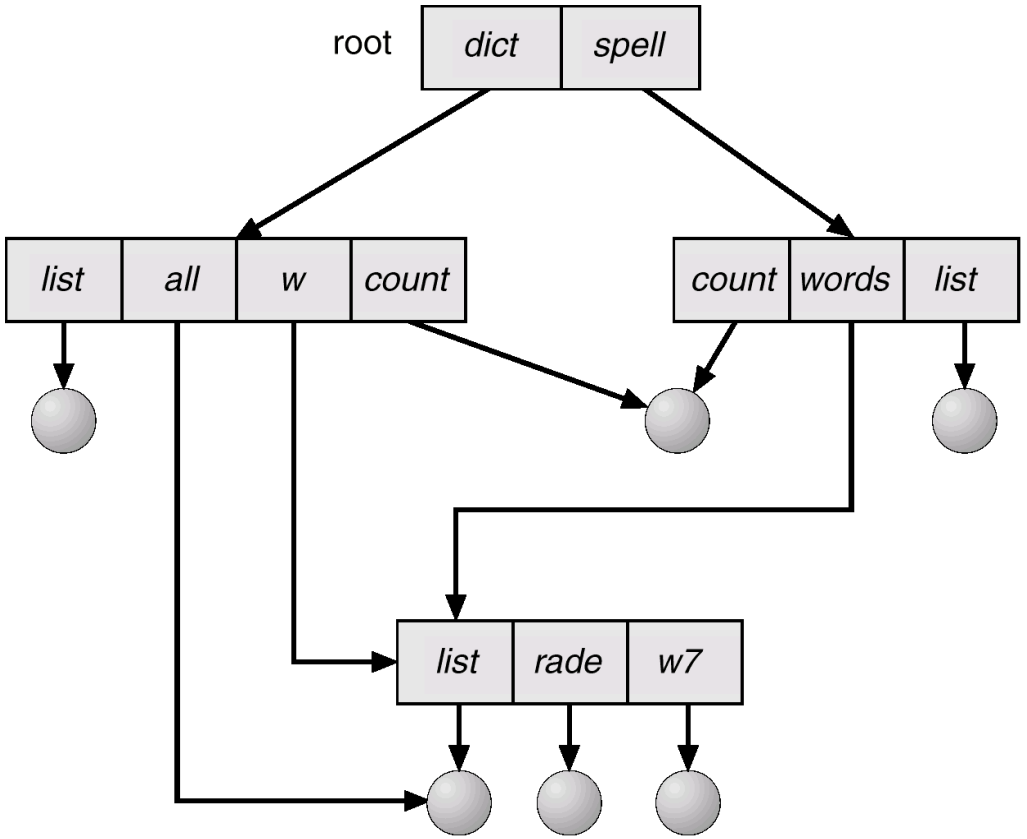
Répertoires à arbres (normal aujourd'hui)



Caractéristiques des répertoires à arbres

- **Recherche efficace**
- **Possibilité de grouper**
- **Repertoire courant (working directory)**
  - **cd** /spell/mail/prog

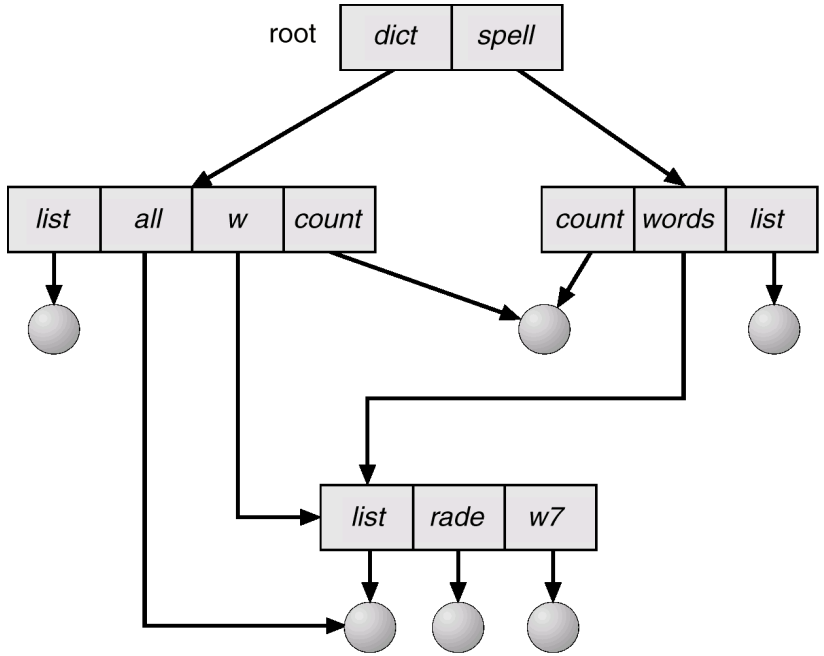
Graphes sans cycles: permettent de partager fichiers



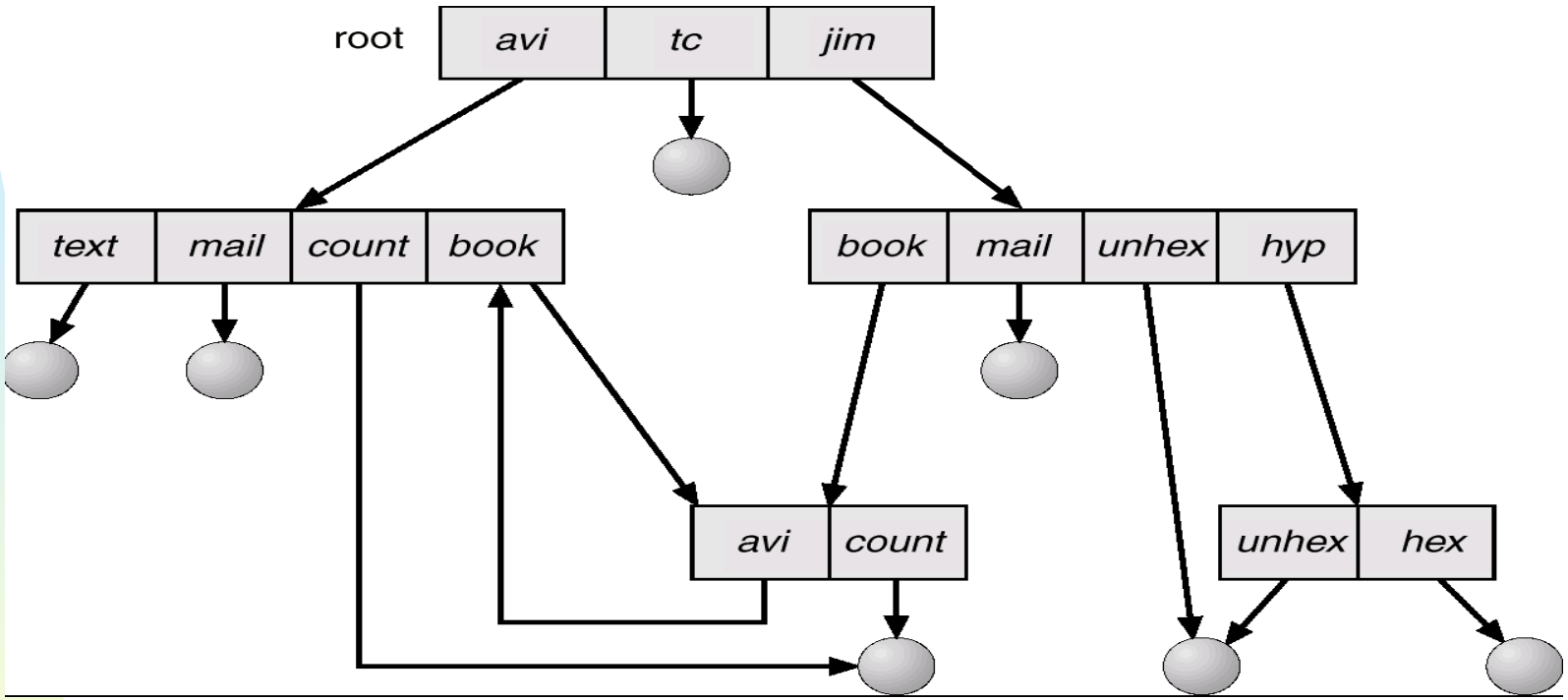
Unix: un *symbolic link* donne un chemin à un fichier ou sous-répertoire

Références multiples dans les graphes acycliques

- **Un nœud peut avoir deux noms différents**
- **Si dict supprime list, le pointeur vers fichier devient inexistant (dangling pointer). Solutions:**
  - Pointeurs rétractifs, effacent tous les pointeurs
  - Compteurs de références (s'il y a encore des refs au fichier, il ne sera pas effacé)
  - Ni Unix ni Microsoft n'implémentent ces politiques, donc messages d'erreur
  - Solutions impossibles à gérer dans un système fortement réparti (ex: www)



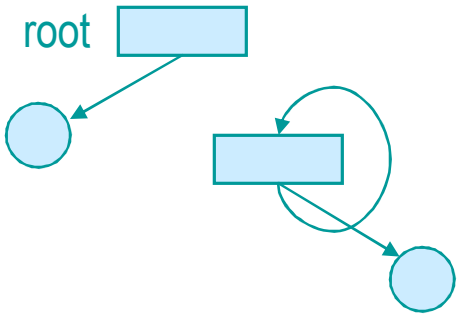
Graphes avec cycles (structure générale)



- Presque inévitables quand il est permis de pointer à un noeud arbitraire de la structure
- Pourraient être détectés avec des contrôles appropriés au moment de la création d 'un nouveau pointeur
- Contrôles qui ne sont pas faits dans les SE courants

Considérations dans le cas de cycles

- **En traversant le graphe, il est nécessaire de savoir si on retombe sur un nœud déjà visité**
- **Un nœud peut avoir un compteur de ref != 0 en se trouvant dans une boucle de nœuds qui n'est pas accessible!**
- **Des algorithmes existent pour permettre de traiter ces cas, cependant ils sont compliqués et ont des temps d'exécution non-négligeables, ce qui fait qu'ils ne sont pas souvent employés**
  - Ramasse-miettes = garbage collection



Un sous-arbre qui n'est pas accessible à partir de la racine mais qui ne peut pas être effacé en utilisant le critère ref=0 car il fait référence à lui-même!

Combiner plusieurs systèmes de fichier

Le système de fichier

- Répertoire qui réside dans une partition/appareil spécifique

Pourquoi combiner?

- Plusieurs partitions de disques rigides, disquettes, CDROM, disques réseau.
- Vision et accès uniforme

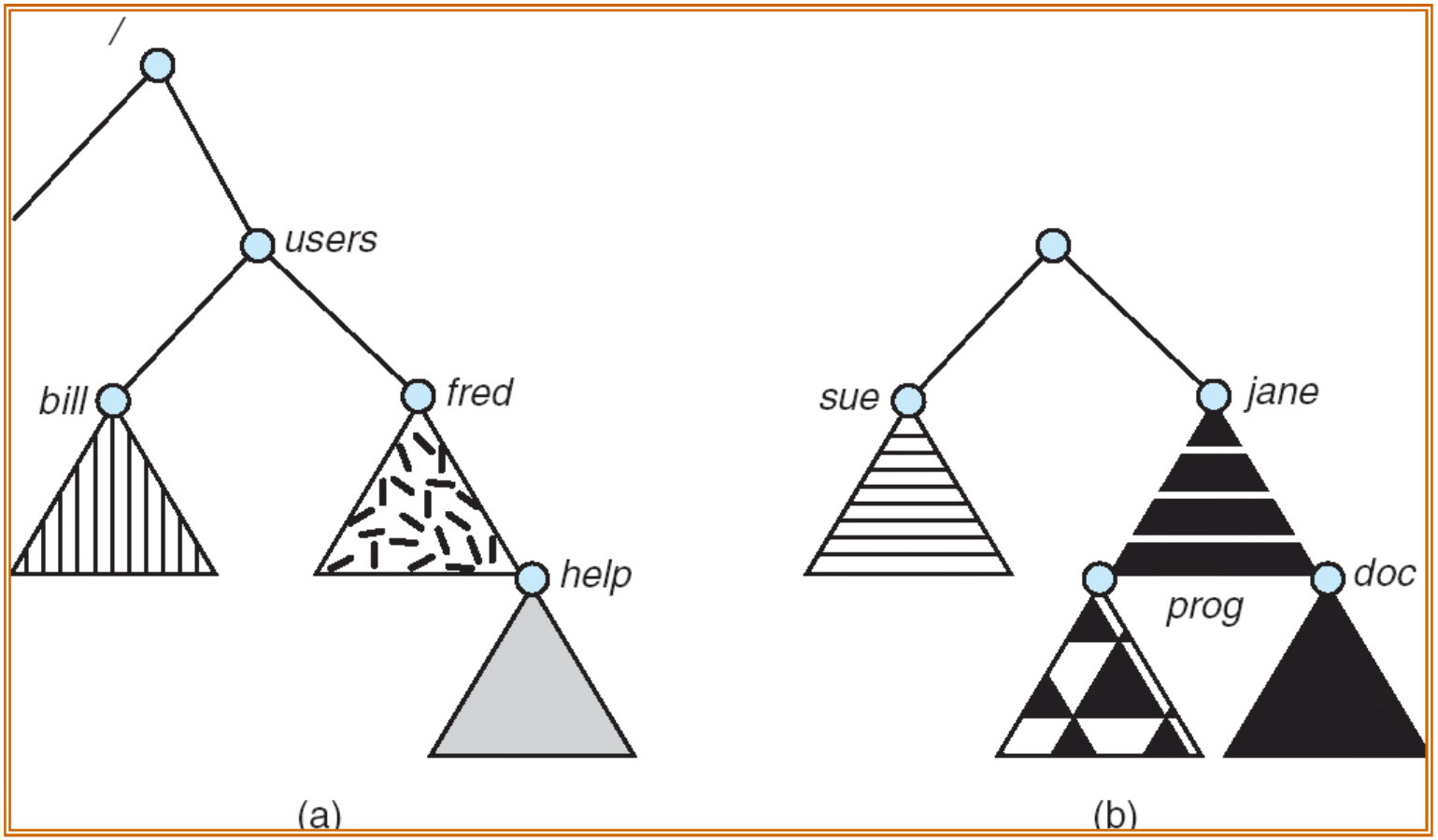
Comment?

- Attacher un système de fichiers à un nœud particulier dans la hiérarchie du répertoire.
- Windows: système à 2 niveaux – attaché à des lettres d'appareils
- Unix: opération d'attachement explicite (mount), peut attacher un système de fichier n'importe où dans le répertoire.

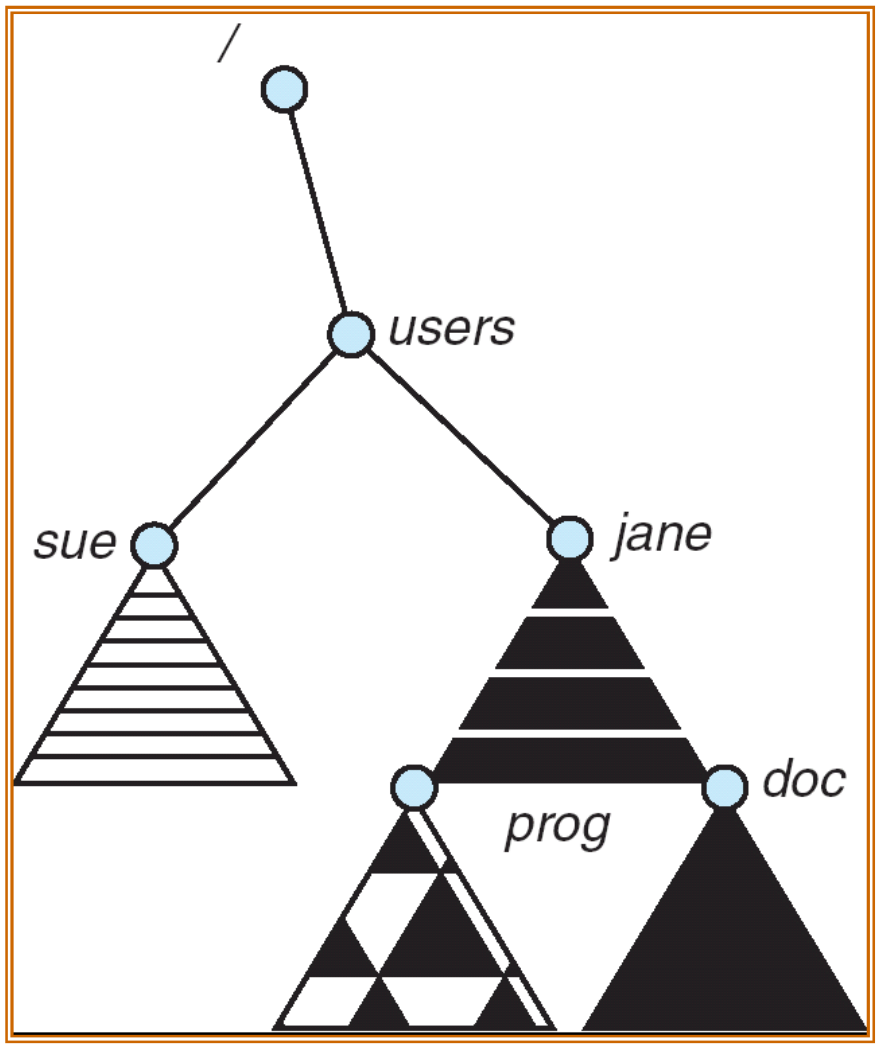
Attachement du système de fichier

- Un système de fichier doit être attaché (mounted) avant d'être accédé
- Un système de fichier est attaché à un point d'attachement (mount point) – voir Fig. 11-11(b).

(a) Existant. (b) Partition non-attachée



Point d'attachement (mount point)



Partage de fichiers

- **Désirable sur les réseaux**
- **Nécessite de la protection**

Protection

- **Types d'accès permis**
  - **lecture**
  - **écriture**
  - **exécution**
  - append (annexion)
  - effacement
  - listage: lister les noms et les attributs d 'un fichier
- **Par qui**

Listes et groupes d'accès - UNIX

- **Modes d'accès: R W E**
- **Trois classes d'usager:**
  - propriétaire
  - groupe
  - public
- **Demande à l'administrateur de créer un nouveau groupe avec un certain usager et un certain propriétaire**
- **Droit du propriétaire de régler les droits d'accès et d 'ajouter des nouveaux usagers**

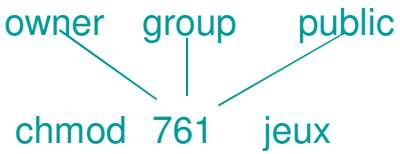
Listes et groupes d'accès

- **Mode d'accès: read, write, execute**
- **Trois catégories d'usagers:**

			RWX
a) owner access	**7**	⇒	**1 1 1**
b) group access	**6**	⇒	**1 1 0**
c) others access	**1**	⇒	**0 0 1**

- **Demande au gestionnaire de créer un groupe, disons G, et ajouter des usagers à ce groupe**

- **Pour un fichier particulier, disons jeux, définir un accès approprié**



Changer le groupe d'un fichier

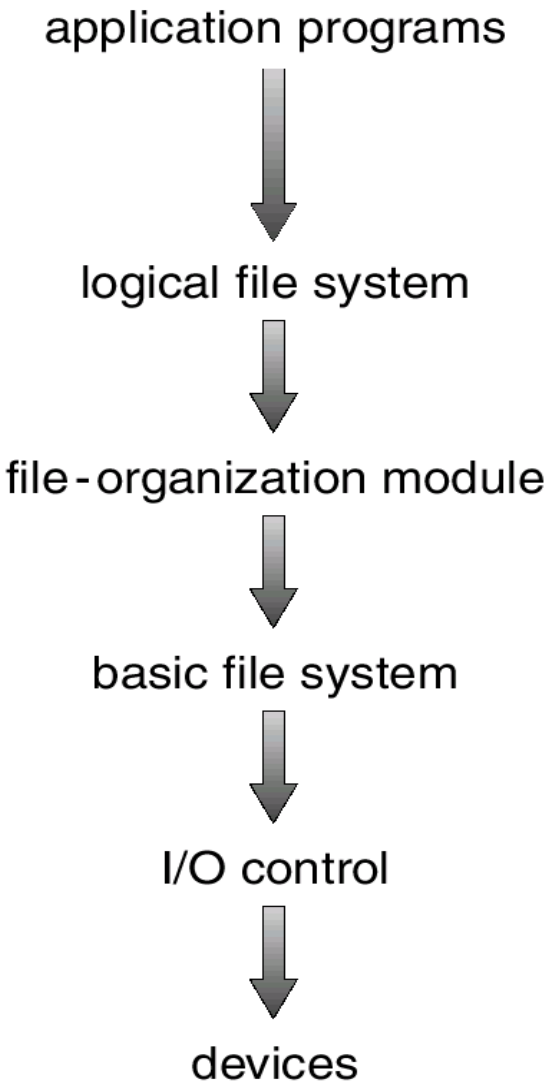
chgrp G *jeux*

Méthodes d’allocation

Structures de systèmes de fichiers

- **Structure de fichiers: deux façons de voir un fichier:**
  - unité d'allocation espace
  - collection d'informations reliées
- **Le système de fichiers réside dans la mémoire secondaire: disques, rubans...**
- **File control block: structure de données contenant de l'information d'un fichier**

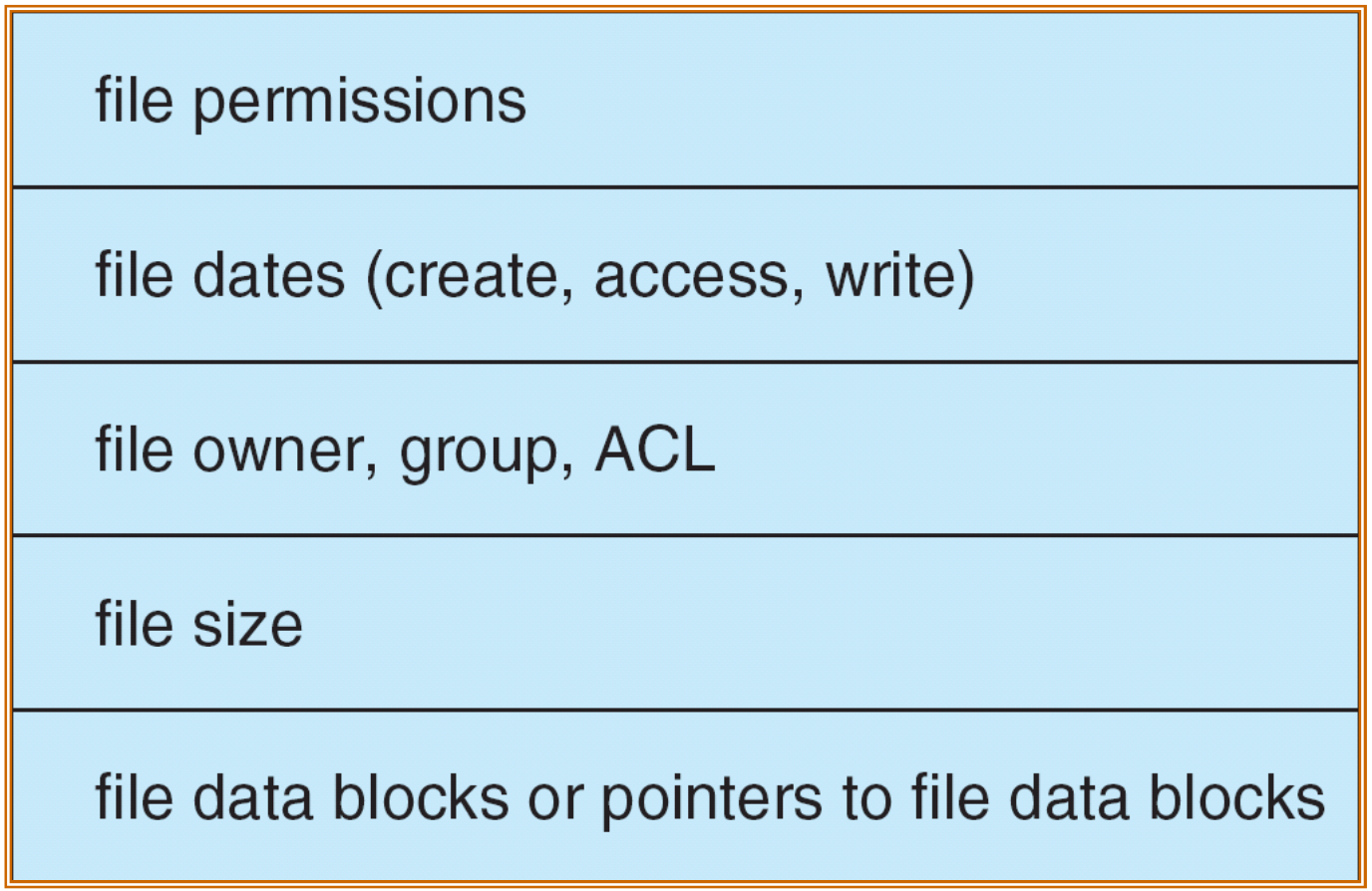
Systèmes de fichiers à couches



Structure physique des fichiers

- **La mémoire secondaire est subdivisée en blocs et chaque opération d'E/S s'effectue en unités de blocs**
  - Les blocs ruban sont de longueurs variables, mais les blocs disque sont de longueurs fixes
  - Sur disque, un bloc est constitué de multiples secteurs contiguës (ex: 1, 2, ou 4)
    - la taille d'un secteur est habituellement 512 bytes
- **Il faut donc insérer les enregistrements dans les blocs et les extraire par la suite**
  - Simple lorsque chaque octet est un enregistrement par lui-même
  - Plus complexe lorsque les enregistrements possèdent une structure (ex: « main-frame IBM »)
- **Les fichiers sont alloués en unité de blocs. Le dernier bloc est donc rarement rempli de données**
  - Fragmentation interne

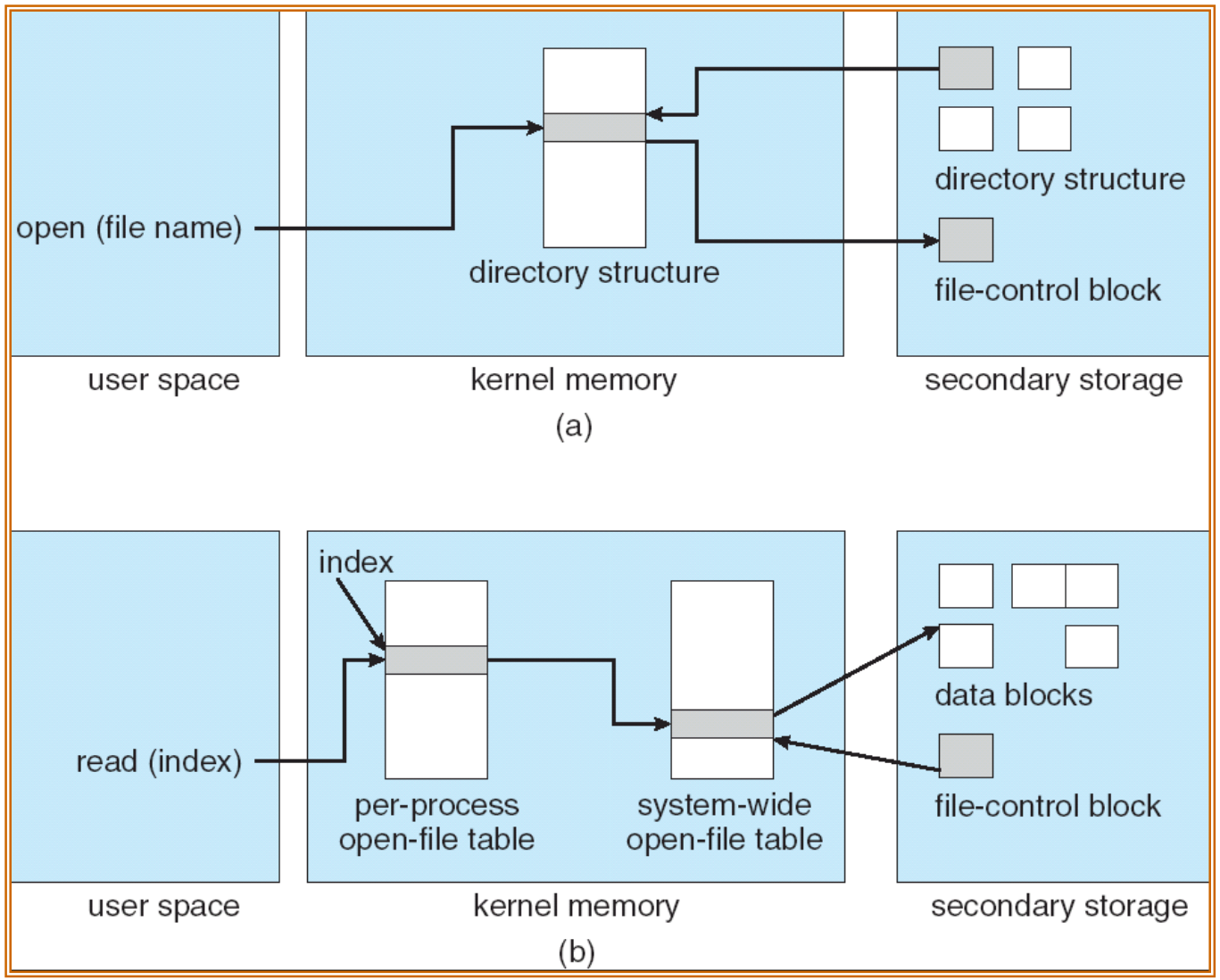
Un “File Control Block” typique



Structure en-mémoire du système de fichier

Overture d'un fichier

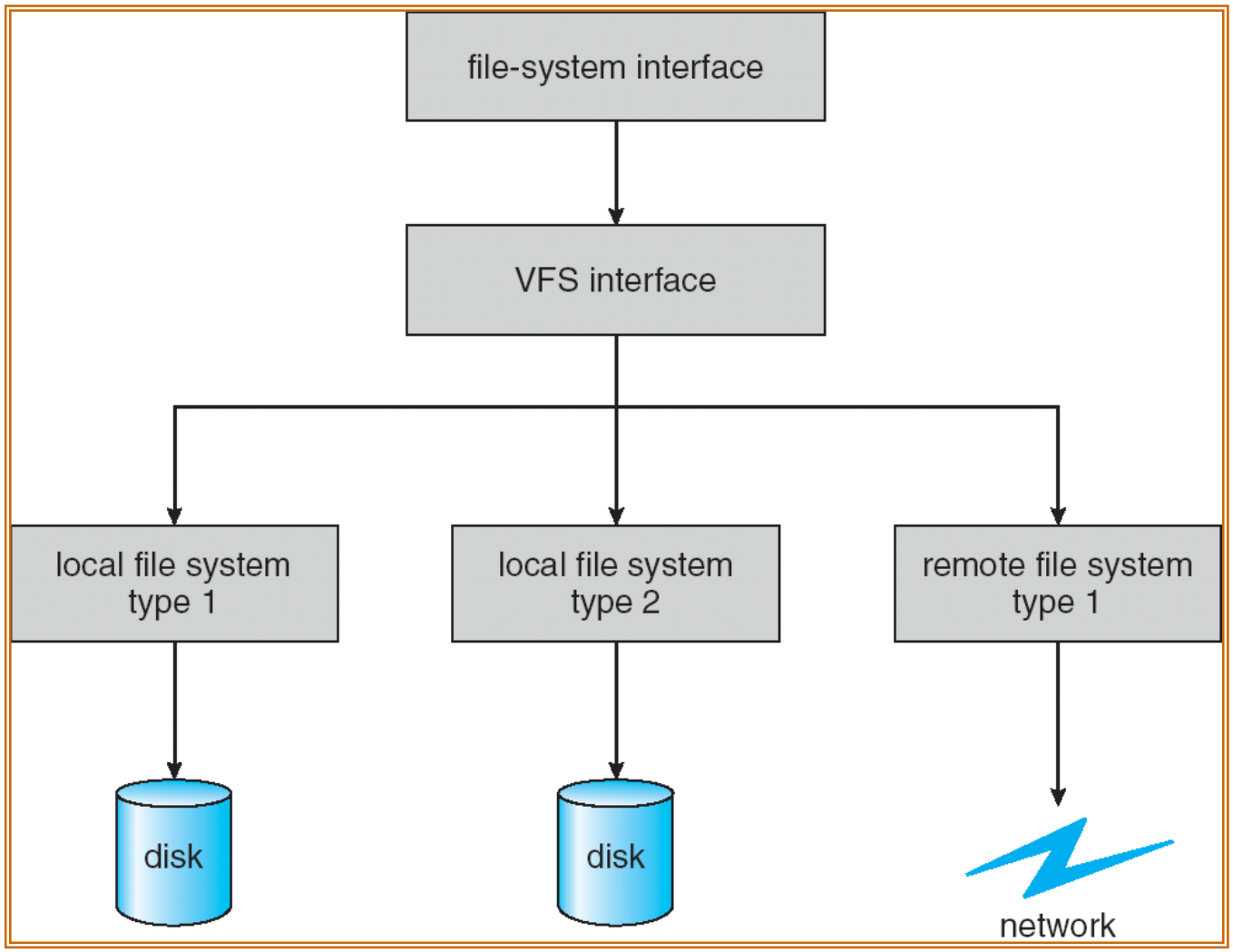
Lecture d'un fichier



Système de fichier virtuel

- VFS (virtual file system) utilise une approche objet orienté (OO) pour réaliser les systèmes de fichiers.
- VFS permet une interface d'appels systèmes (API) pour accès différents types de systèmes de fichier.
- Le API est l'interface au VFS, plutôt qu'à un type spécifique de système de fichier.

Schéma du VFS



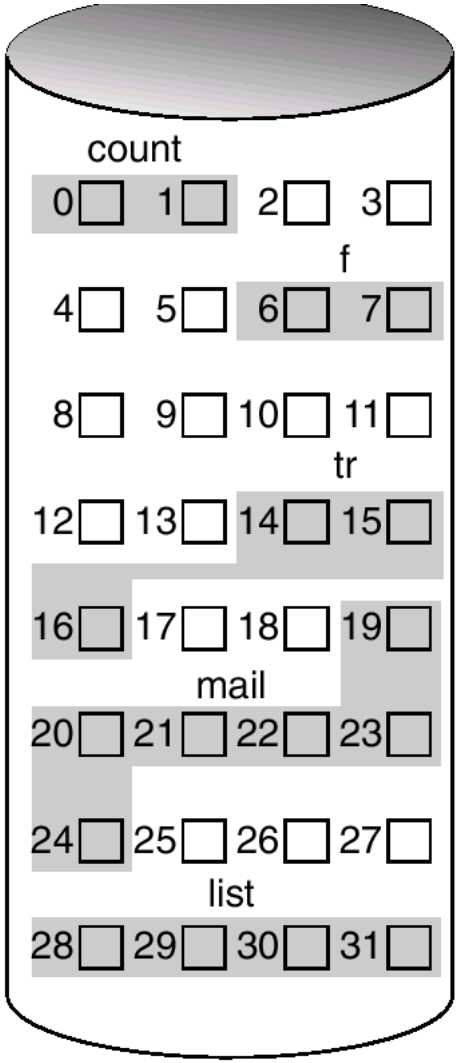
L'implémentation du répertoire

- Liste linéaire des noms de fichiers avec des pointeurs aux blocs de données.
  - **Facile à programmer**
  - **Prends un plus long temps d'exécution**
- Tableau de hachage – liste linéaire avec une structure de données hachées.
  - **Réduit le temps de recherche dans le répertoire**
  - **Collisions – situation où deux noms de fichiers sont hachés au même endroit**
  - **Grandeur du tableau est fixe**

Trois méthodes d'allocation de fichiers

- Allocation contiguë
- Allocation enchaînée
- Allocation indexée

Allocation contiguë sur disque



répertoire

directory

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

Allocation contiguë

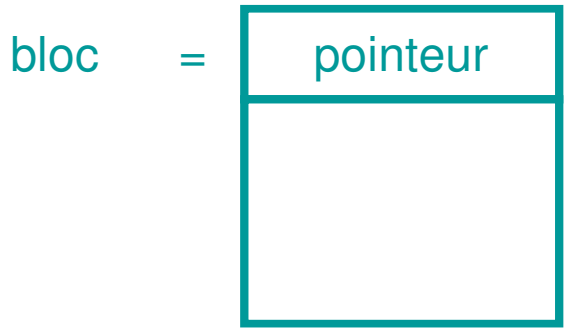
- Chaque fichier occupe un ensemble de blocs contigu sur disque
- Simple: nous n'avons besoin que de l'adresse du début et de la longueur
- Supporte aussi bien l'accès séquentiel, que l'accès direct
- Moins pratique pour les autres méthodes

Allocation contiguë

- Application des problèmes et méthodes vus dans le chapitre de l'allocation de mémoire contiguë
- Les fichiers ne peuvent pas grandir
- Impossible d'ajouter au milieu
- Exécution périodique d'une compression (compaction) pour récupérer l'espace libre

Allocation enchaînée

- Le répertoire contient l'adresse du premier et dernier bloc, possiblement le nombre de blocs
- Utilisé par MS-DOS et OS2.
- Chaque bloc contient un pointeur de l'adresse du prochain bloc:



Allocation enchaînée

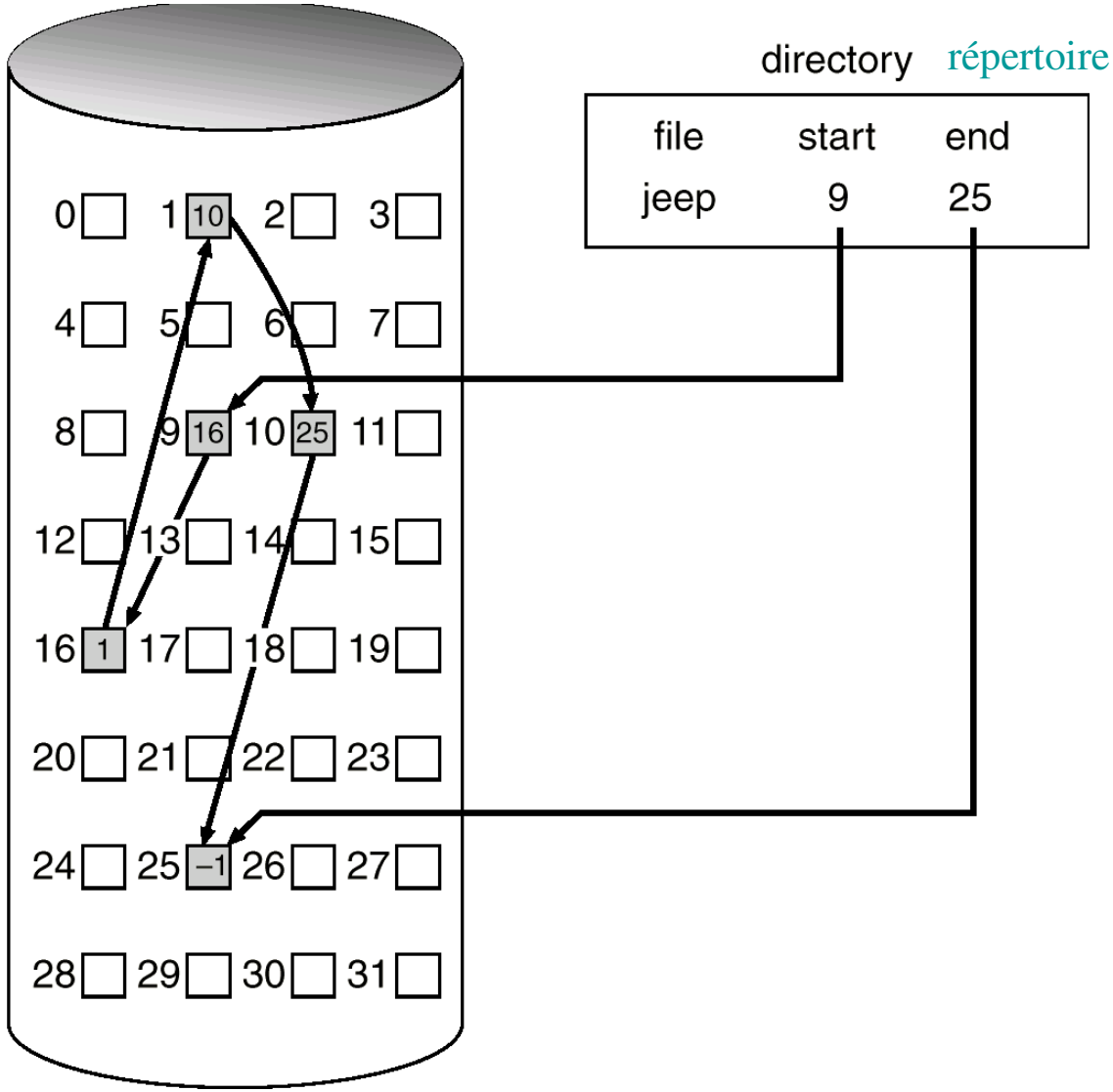
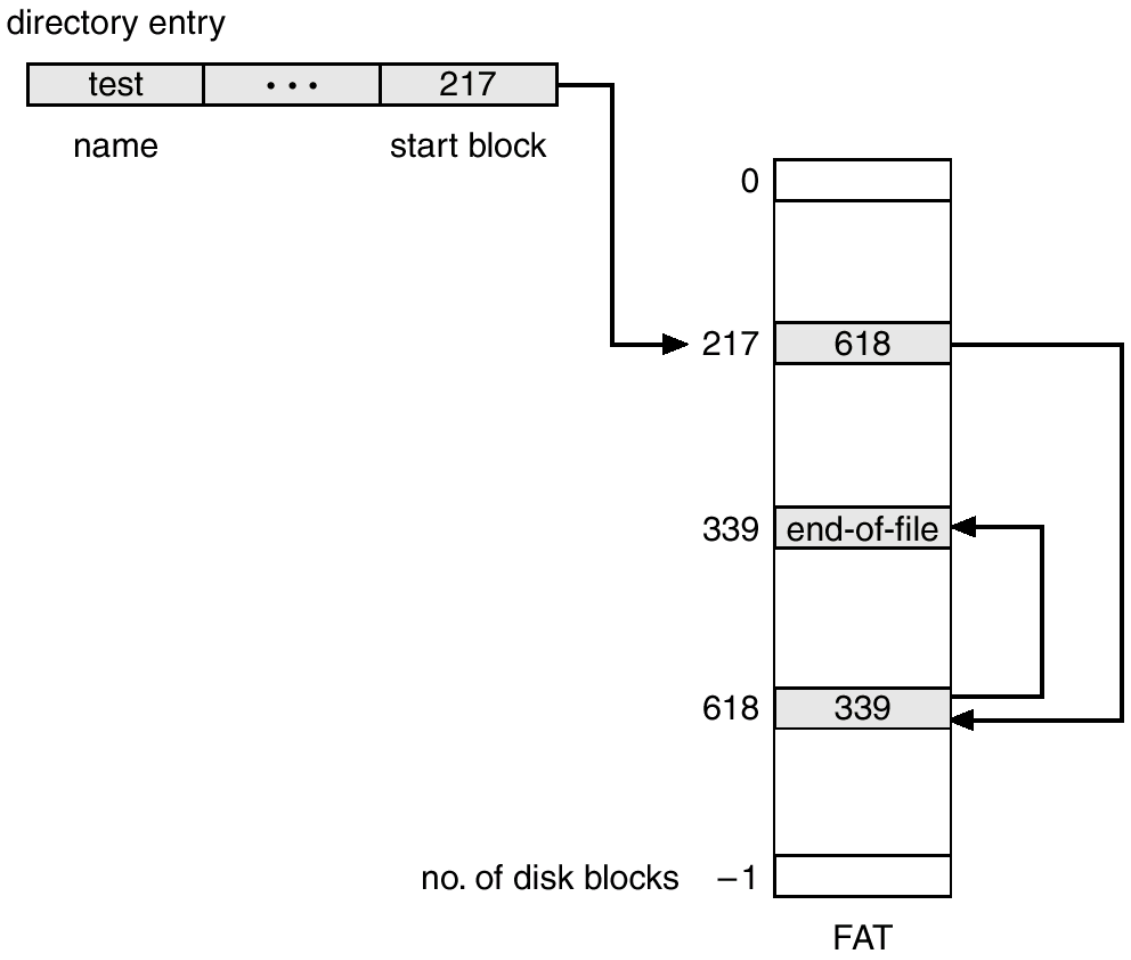


Tableau d'allocation de fichiers (FAT)

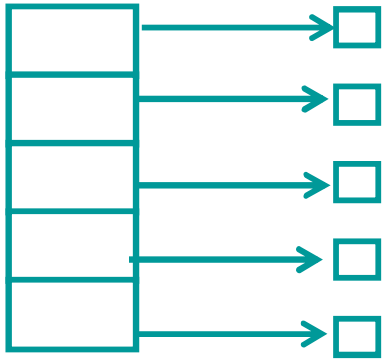


Avantages - désavantages

- **Pas de fragmentation externe - allocation de mémoire simple, pas besoin de compression**
- **L'accès à l'intérieur d'un fichier ne peut être que séquentiel**
  - Pas possible de trouver directement le 4ème enregistrement...
  - N'utilise pas la localité car les enregistrements seront éparpillés
- **L'intégrité des pointeurs est essentielle**
- **Les pointeurs gaspillent un peu d'espace**

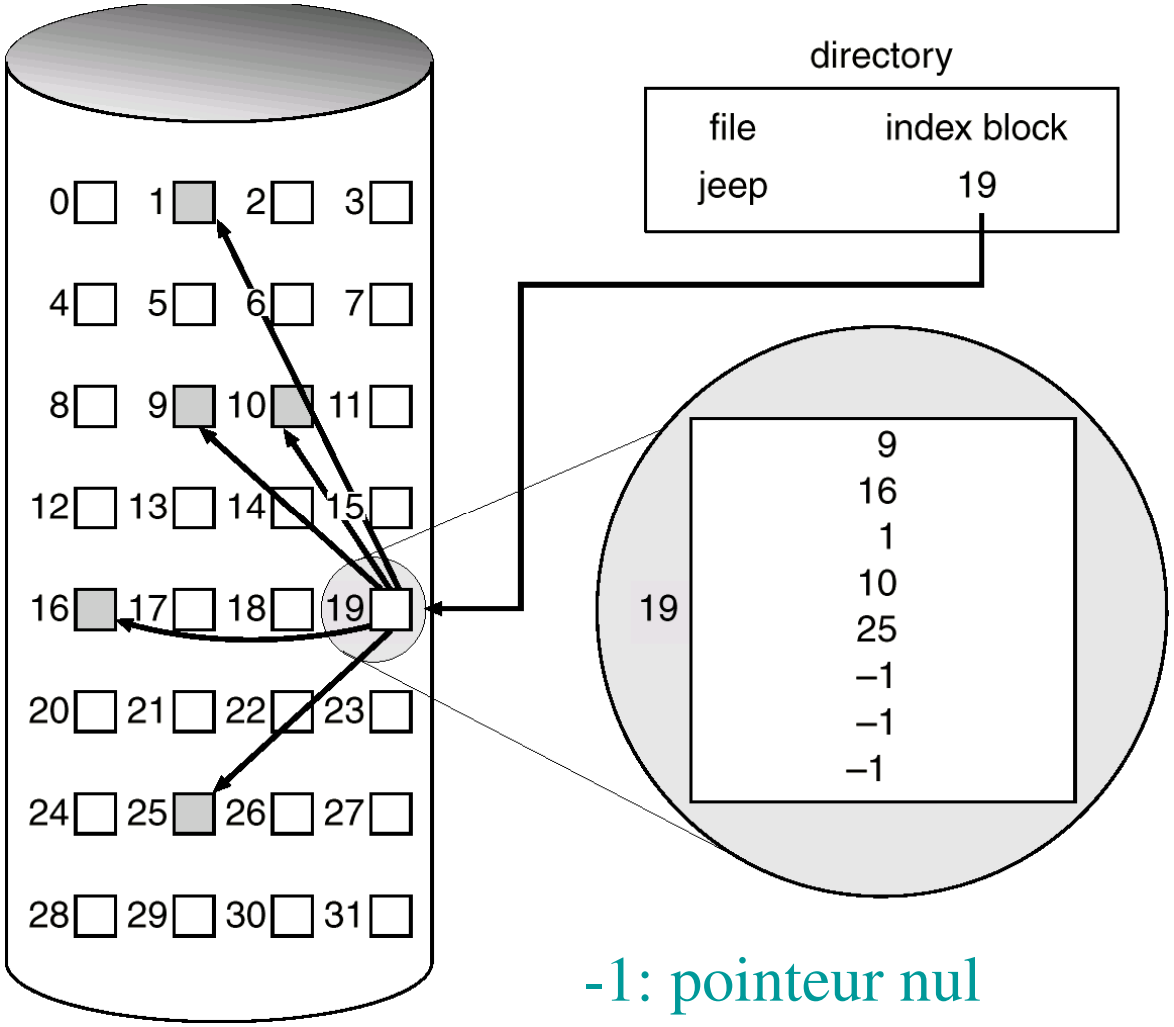
Allocation indexée: semblable à la pagination

- **Tous les pointeurs sont regroupés dans un tableau (index block)**



index table

Allocation indexée



-1: pointeur nul

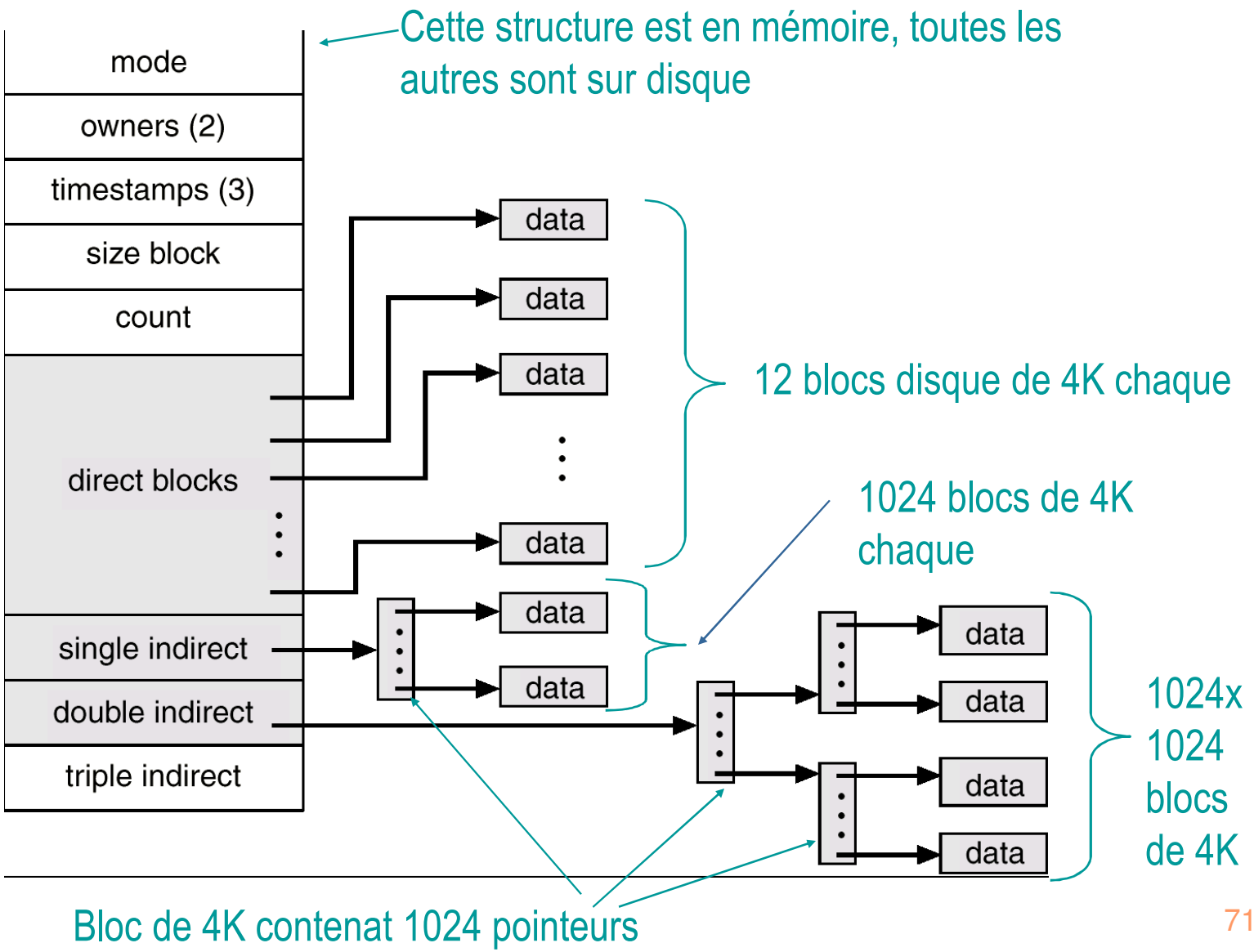
Allocation indexée

- **À la création d ’un fichier, tous les pointeurs dans le tableau sont *nil (-1)***
- **Chaque fois qu’un nouveau bloc doit être alloué, on trouve de l’espace disponible et on ajoute un pointeur avec son adresse**

Allocation indexée

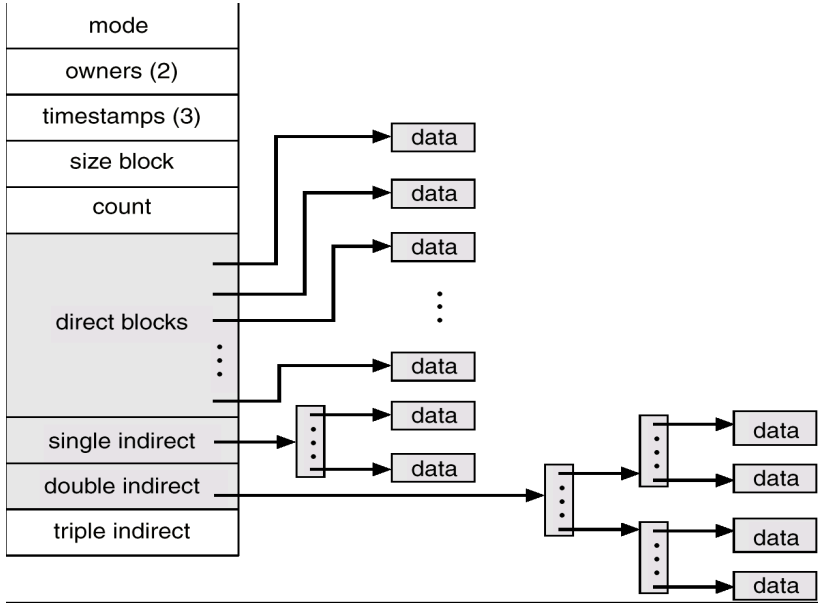
- **Pas de fragmentation externe, mais les index prennent de l'espace**
- **Permet un accès direct (aléatoire)**
- **Taille de fichiers limitée par la taille de l'indexe bloc**
  - Mais nous pouvons avoir plusieurs niveaux d'indexes: Unix
- **L'indexe bloc peut utiliser beaucoup de mémoire.**

UNIX BSD: indexé à niveaux (config. possible)



UNIX BSD

- Les premiers blocs d'un fichier sont accessibles directement
- Si le fichier contient des blocs additionnels, les premiers sont accessibles à travers un niveau d'indices
- Les suivants sont accessibles à travers 2 niveaux d'indices, etc.
- Donc le plus loin du début un enregistrement se trouve, le plus indirect est son accès
- Permet un accès rapide de petits fichiers, et au début des fichiers
- Permet l'accès de grands fichiers avec un petit répertoire en mémoire



Gestion de l’espace libre

Gestion d'espace libre

Solution 1: vecteur de bits (solution Macintosh, Windows 2000)

- **Vecteur de bits (*n* blocs)**

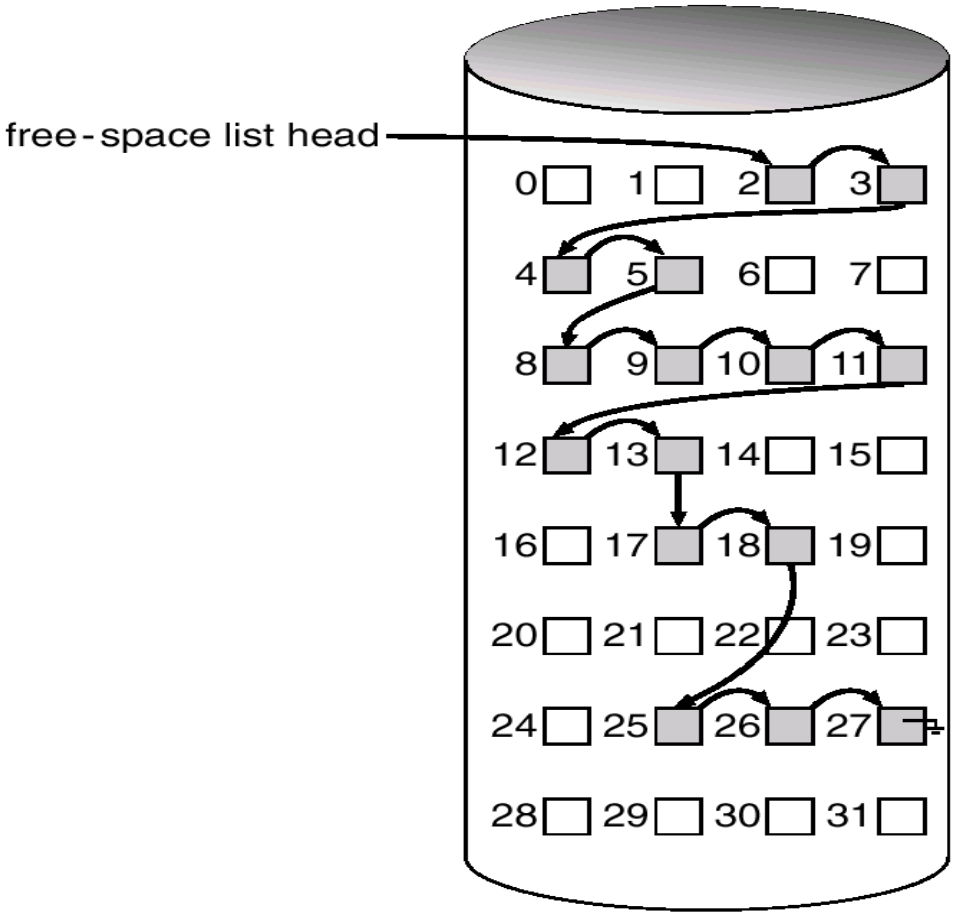
$$bit[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ libre} \\ 1 \Rightarrow \text{block}[i] \text{ occupé} \end{cases}$$

- **Exemple d'un vecteur de bits où les blocs 3, 4, 5, 9, 10, 15, 16 sont occupés:**
  - 0001110001100011…
- **L'adresse du premier bloc libre peut être trouvée par un simple calcul**

Gestion d'espace libre

Solution 2: Liste liée de mémoire libre (MS-DOS, Windows 9x)

Tous les blocs de mémoire libres sont reliés ensemble par des pointeurs



Comparaison

- **Bitmap:**
  - si le bitmap de toute la mémoire secondaire est gardée en mémoire principale, la méthode est rapide mais demande de l'espace de mémoire principale
  - si les bitmaps sont gardés en mémoire secondaire, il s'en suit un temps de lecture de mémoire secondaire...
    - Ils pourraient, par exemple, être paginés.
- **Liste liée**
  - Pour trouver plusieurs blocs de mémoire libre, plusieurs accès de disque pourraient être demandés
  - Pour augmenter l'efficacité, nous pouvons garder en mémoire centrale l'adresse du 1er bloc libre

Implantation de répertoires (directories)

- **Liste linéaire de noms de fichiers avec des pointeurs aux blocs de données**
  - accès séquentiel
  - simple à programmer
  - temps nécessaire pour parcourir la liste
- **Tableaux de hachage: tableaux calculés**
  - temps de recherche rapide
  - problème de collisions
  - dimension fixe du tableau

Efficacité et performance

- **L'efficacité dépend de:**
  - La méthode d'allocation et d'organisation répertoires
- **Pour augmenter la performance:**
  - Rendre efficace l'accès aux blocs souvent visités
    - Dédier des tampons de mémoire qui contiennent l'image des informations plus souvent utilisées
  - Optimiser l'accès séquentiel s'il est souvent utilisé: free behind and read ahead

Récupération: différentes méthodes

- **Contrôle de cohérence entre la structure de répertoires en mémoire centrale et le contenu des disques**
  - Essaye de réparer les incohérences
- **Programmes du système pour sauvegarder les données sur disque dans autres supports auxiliaires (backups) (ex. autres disques, rubans)**
- **Restaurer les disques à partir de ces supports quand nécessaire**

Systèmes de fichiers avec consignation (log structured or journaling file system)

- Les systèmes de fichiers avec consignation enregistrent chaque mis à jour du système de fichier comme une transaction
- Toutes transactions sont écrites dans un journal
  - **Une transaction est considérée engagée à l'écriture du journal**
  - **Mais, le système de fichier (sur disque) n'as pas encore été mis à jour**
- Les transactions dans le journal sont écrites aux système de fichiers de façon asynchrone
  - **Quand le système de fichiers est modifié, la transaction est enlevée du journal**
- Si un système tombe en panne, les transactions dans le journal doivent être appliquées

Concepts importants du Chapitre 11

- **Fichiers: structures, attributs, opérations**
- **Méthodes d'accès: séquentiel, séquentiel indexée, indexée, direct ou hachée**
- **Répertoires et leur structures: répertoires arborescents, sans ou avec cycles**
- **Partage de fichiers, protection, liste d'accès**
- **Allocation d'espace: contiguë, enchaînée, indexée**
  - Application en UNIX
- **Gestion d'espace libre: bitmap, liste liée**