LINGI 1113: Systèmes informatiques 2

Mission 6: entrées/sorties et fichiers

ÉCOLE
POLYTECHNIQUE
DE LOUVAIN

UCL

Entrées/sorties

Débits variés

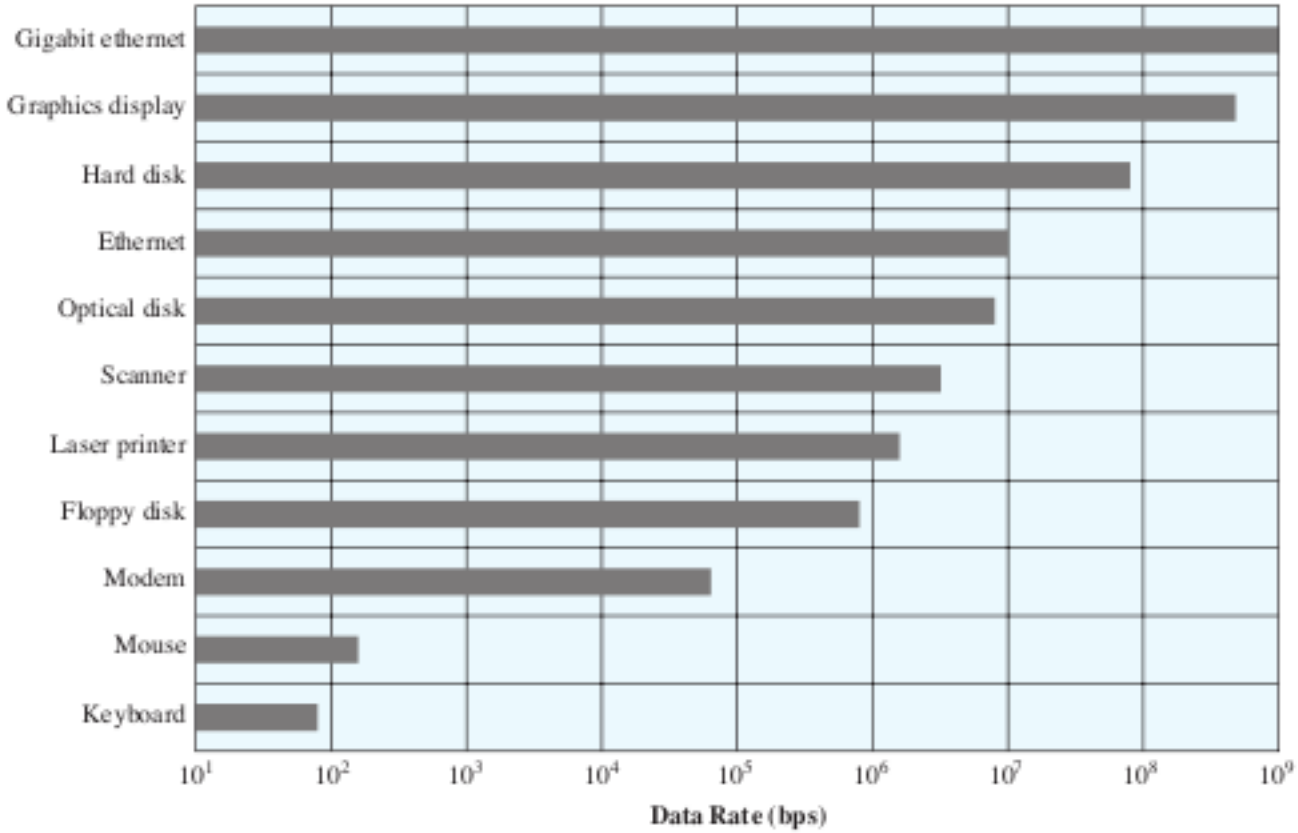


Figure 11.1 **Typical I/O Device Data Rates**

Entrées/sorties

Techniques variées:

- Programmed I/O: attente active
- Interrupt driven I/O
- Direct memory access:

  2 adresses

  1 adresse

  1 adresse

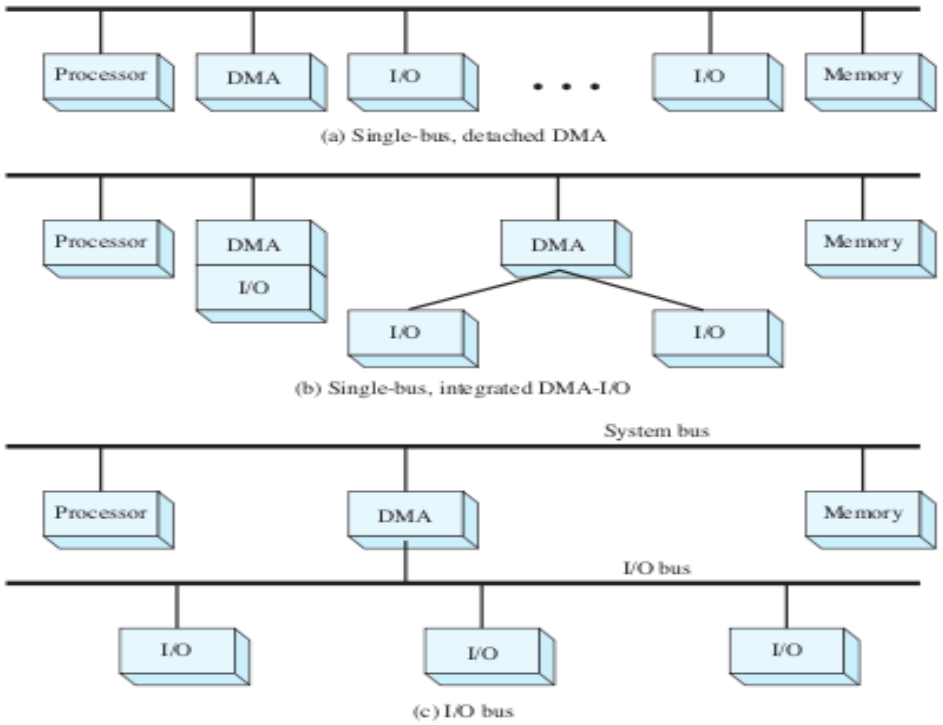


Figure 11.3 Alternative DMA Configurations

Entrées/sorties

Répartitions diverses du travail entre CPU et contrôleur I/O controller:

- Simple port //, le processeur fait tout par programme (interface disque des premiers Pcs DOS)
- Contrôleurs simples: port série UART(conversion série/parallèle, bits d'état et de contrôle, avec ou sans usage d' interruptions à la réception de caractères et quand prêt à envoyer
- Contrôleur I/O avec DMA: il faut pouvoir dire combien de caractères transférer et où les placer; interruption en fin de transfert
- Co-processeur chargé d'activités d'I/O (contrôleurs de terminaux et de disques sur mainframes: programmes en mémoire centrale)
- Co-processeurs avec mémoire propre, processeurs graphiques

Entrées/sorties

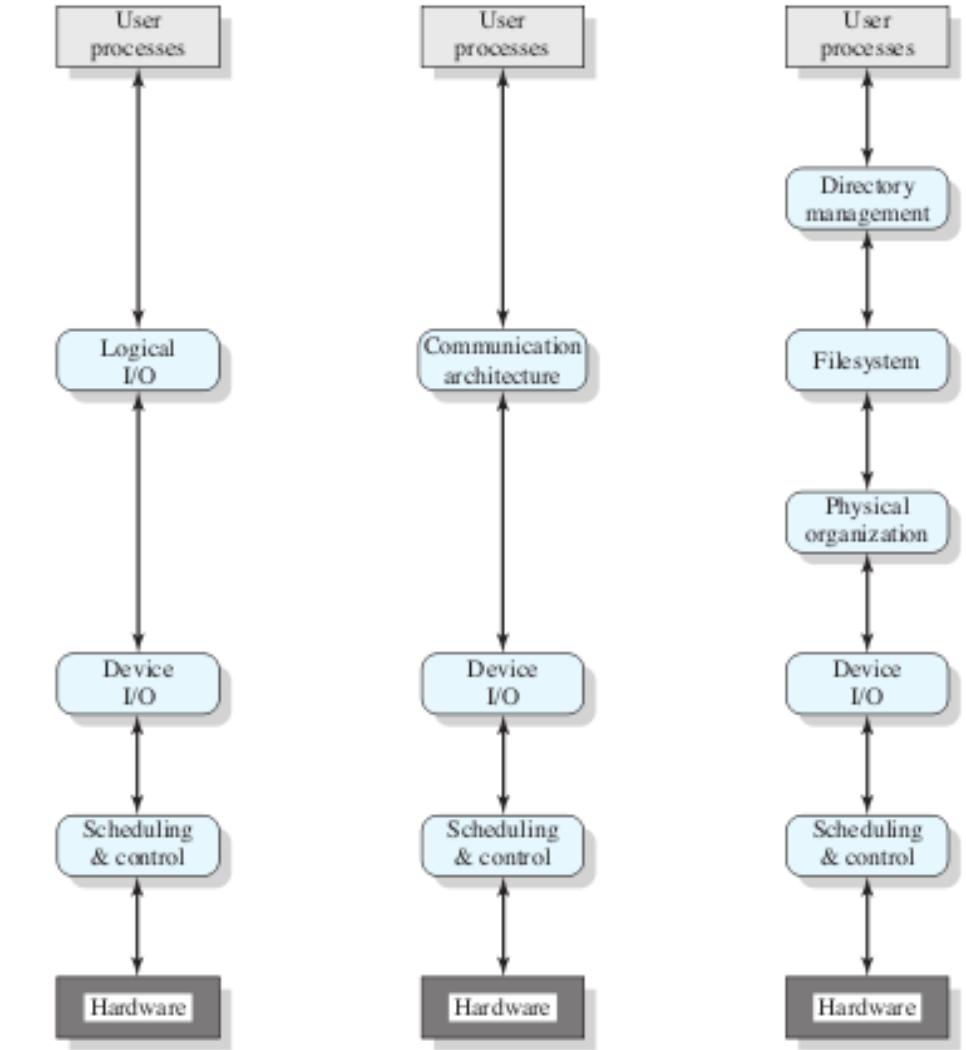
Objectifs de l'OS

- performance
- Généralité: cacher autant que possible au processus les particularités d'un périphérique donné:

  UNIX-like: open, read, write, comme dans un fichier
  => organisation en couches

Entrées/sorties

=>organisation en couches
pour généralité



(a) Local peripheral device (b) Communications port (c) Filesystem

Figure 11.4 A Model of I/O Organization

Entrées/sorties

Buffering pour performance

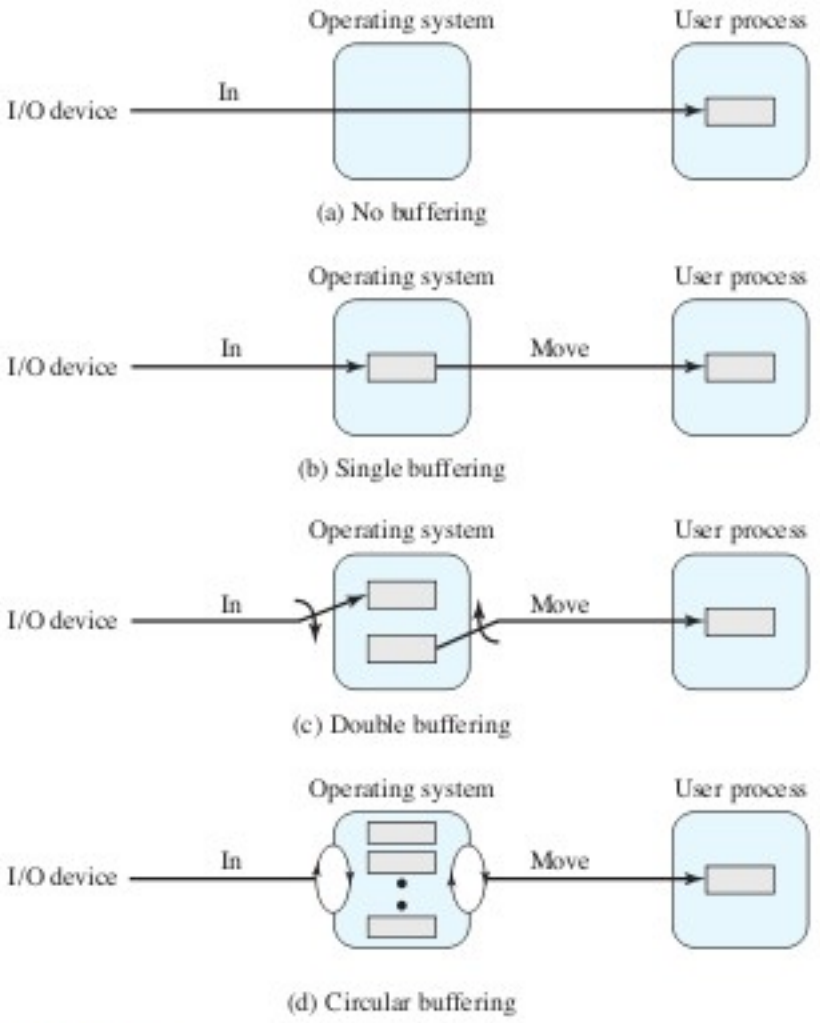


Figure 11.5 **I/O Buffering Schemes (input)**

Transfert disques

Timing

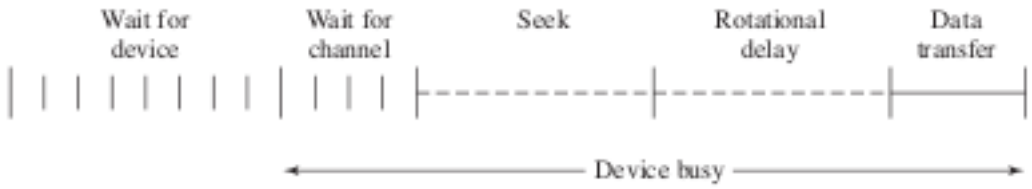


Figure 11.6 Timing of a Disk I/O Transfer

Ordonnancement des transferts:

- FIFO: mouvements du bras aléatoires, mais « fair »
- Shortest Service Time First (d'abord aux plus proches)
- Scan: comme ascenseur: ne change de sens que si plus de destinations dans le même sens
- C-scan: ne scanne que dans un sens

  Ces trois peuvent être injustes: si un processus à beaucoup d'activité sans movement, les autres n'auront rien

Transfert disques

Pour être efficace mais juste:

- Fscan:2 queues: on en scanne une et les nouvelles requetes vont dans l'autre puis on inverse
- N-step-scan: plusieurs queues de taille N, limitée: quand une est pleine, on la scanne et on en remplit une autre. S'il n'y en a plus qu'une, on la scanne sans ettendre qu'elle se remplisse.
  (NB: si N=1, c'est du FIFO!)

Autre façon d'améliorer les performances: parallelisme:

RAID: Redundant Array of Inexpensive Disks: 6 variantes dont 4 utilisées

RAIDS

Raid 0: disk striping - rapide mais si un disque crashe on perd toutes les données

Raid 1: mirroring – sûr mais cher: 2 fois plus de disques: souvent 2 disques mais tout nombre pair est possible; striping entre les disques d'une des copies

Raid 2: striping + redondance par code de Hamming (comme mémoires ECC) bit à bit : disques supplémentaires qui contiennet les bits de parité; permet de détecter 2 erreur et en corriger une

Raid 3: si on ne tient pas à se protéger d'erreurs ne concernant qu'un seul bit mais continuer à travailler si un des disques crashe et récupérer ses données, 1 seul bit de parité suffit

RAIDS

Raid 4: comme raid 3, mais parité au niveau du bloc, pas du bit

Raid 5: dans Raid 4 le disque de parité est sur-utilisé: les blocs de parité sont répartis entre tous les disques, chacun son tour

Raid 6: idem, mais avec 2 bits de parité dans 3 disques différents: il faut que 3 disques crashent pour perdre des données

cache disque

Et bien sûr, on peut accélérer les transferts disques en cachant des blocs en mémoire

Remplacement:

- Least recently used (on remet en tête de liste à chque usage et on sacrifie la queue de liste)
- Least frequently used: faut des compteurs, mais protège sans raison des blocs inactifs qui ont été actifs dans le passé
- Frequency based replacement: la file est divisée en 3 sections: les derniers arrivés sont traités en LRUet on ne les éjecte pas, les moyens en LFU mais on ne les éjecte pas, les plus vieux en LFU et on éjecte les moins utilisés

cache disque

- Frequency based replacement: la file est divisée en 3 sections: les derniers arrivés sont traités en LRUet on ne les éjecte pas, les moyens en LFU mais on ne les éjecte pas, les plus vieux en LFU et on éjecte les moins utilisés

  Explications

  - Première zone: pour ne pas défavoriser les nouveaux arrivants dont le compteur est encore bas (le compteur reste à 1, mais on les remet en tête de liste à chaque usage)
  - 2me zone idem car ceux là commence avec un compteur à 1: faut qu'ils aient le temps de le gonfler s'ils sont actifs
  - 3me zone, là, ils n'y a que des « vieux » et seuls les « vieux actifs » sont gardés

Exemple UNIX

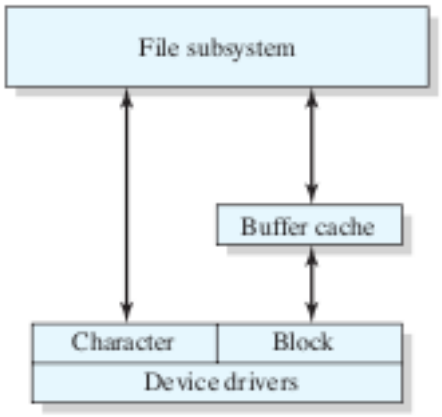


Figure 11.12 **UNIX I/O Structure**

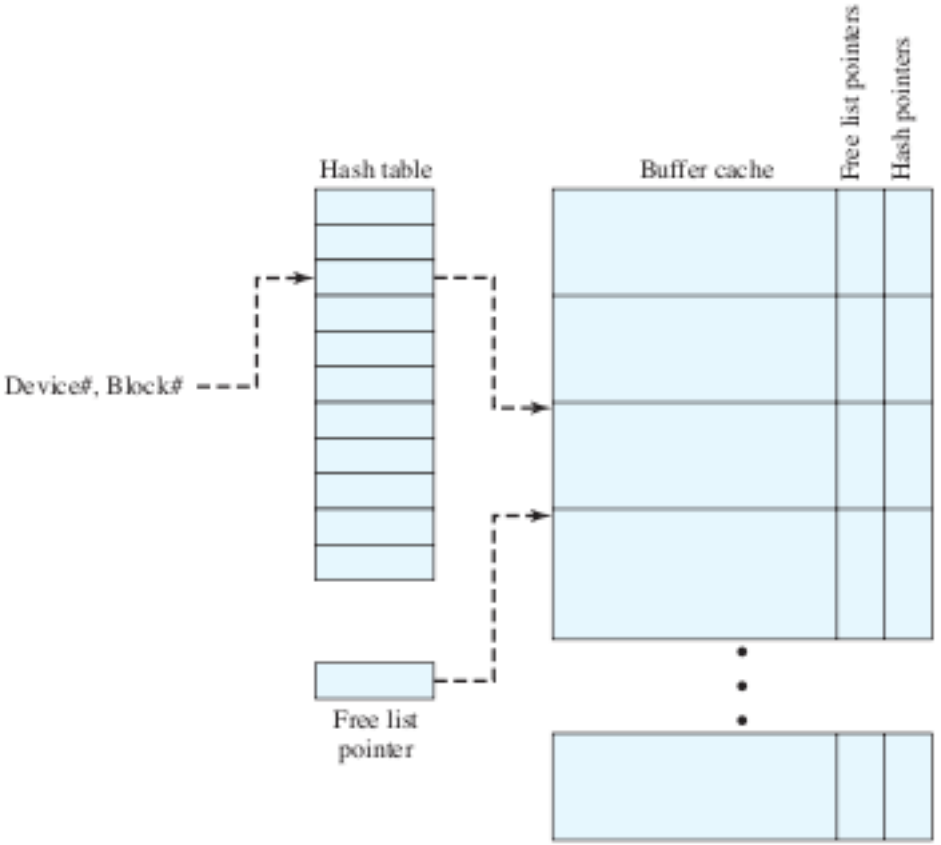


Figure 11.13 **UNIX Buffer Cache Organization**

Systèmes de fichiers

Deux approches:

- **Fichiers de caractères**; ce sont les applications (p. ex. bases de données) qui en définissent la structure et les protections (rwx) +setuid qui protègent contre accès sauvage
  (UNIX et autres OS récents)

- **Fichiers structurés en enregistrements**; les méthodes d'accès sont inclues dans l'OS
  (mainframes traditionnels)

  enregistrement = séquence de champs

Systèmes de fichiers

Les fichiers structurés

- **Empilement**: les entregistrements sont simplement insérés les uns à la suite des autres; ils peuvent être différents et doivent inclure des descripteurs permettant de les identifier; extraction d'un enregistrement par recherche exhaustive
- **Fichiers séquenciels**: tous les enregistrements sont de même type (mêmes champs de même longueur. Un champ (le 1er) sert de clé. Les enregistrements sont triés. Pour ajouter des enregistrements, on les met dans un fichier séparé qu'on intègre régulièrement dans le fichier principal

Systèmes de fichiers

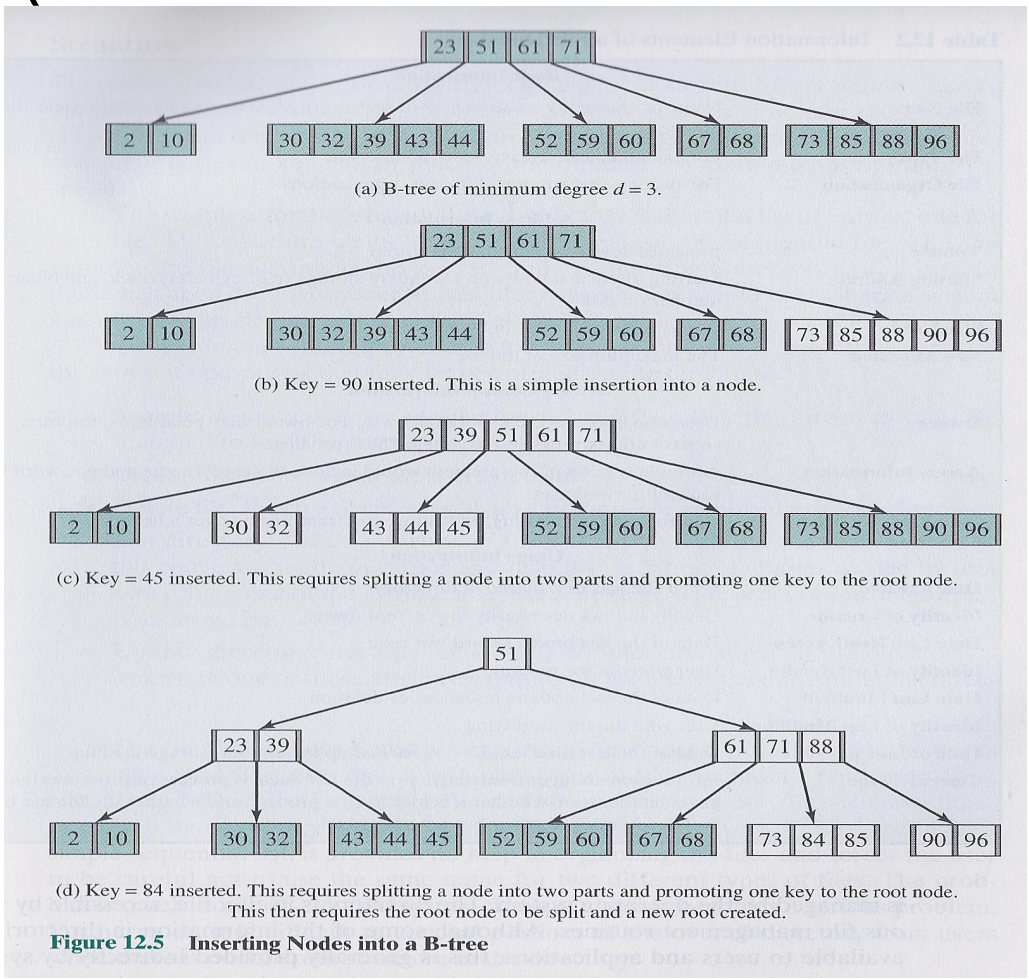
Les fichiers structurés (suite)

- **Fichiers séquenciels indexés**: idem mais on ajoute un index, qui permet d'arriver plus vite à l'enregistrement cherché (sans cela faut recherche dichotomique)
  pour ajouter des enregistrements, chaque enregistrement inclut un pointeur vers un fichier de débordement: on met donc le nouvel enregistrement dans fichier de débordement et on met à jour ce pointeur dans enregistrement précédent
- **Fichiers indexés**: pas triés; un index principal qui permet d'atteindre tous les enregistrements + autres index (partiels) sur autres champs
- **Fichiers en accès direct** : avec tâble de hachage

Systèmes de fichiers

Les fichiers structurés (suite)

- **B-Trees** *(cfr cours de structures de données)*



Systèmes de fichiers

Organisation d'un système de fichiers typiques

- Placé sur un volume (disque, partition, slice)
- Contient des répertoires et des fichiers

- Répertoire: lien entre nom du fichier et stucture de données sur le volume qui permet de retrouver le contenu du fichier et ses attributs (propriétaire, date de création, droits d'accès, ACL, etc.); souvent organisés hiérarchiquement; souvent gérés comme des fichiers de contenu particulier
- Volume: contient informations sur volume (blocs libres), sur les fichiers et le contenu des fichiers

Systèmes de fichiers

Allocation de l'espace aux fichiers

Table 12.3 File Allocation Methods

	Contiguous	Chained	Indexed	
Preallocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table size	One entry	One entry	Large	Medium

- indexé avec allocation de taille variable: on alloue plusieurs blocs contigus, avec pointeur vers 1er et Nblocs

Systèmes de fichiers

Blocs libres: se gèrent à peu près comme un fichier

- Zones libres chaînées entre elles
- Index vers zones libres (constituées de plusieurs blocs) de tailles variables
- Simple liste des numéros de blocs libres

Droits d'accès aux fichiers

Contrôle d'accès de base:

- RWX pour owner, group et autres
- Les attributs du fichier incluent les 9 bits, l'identification du propriétaire et d'un groupe
- Un utilisateur peut être membre de plusieurs groupes
- Bits setuid, setgid

Contrôle d'accès plus général

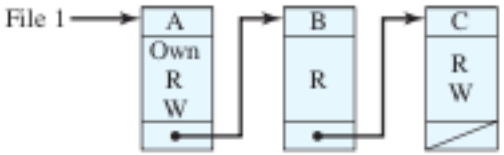
- Droits plus variés
- Structure logique matricielle: agents/objets, mais c'est une matrice creuse

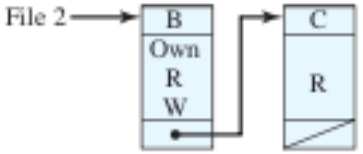
	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

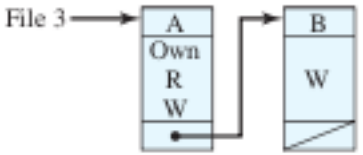
(a) Access matrix

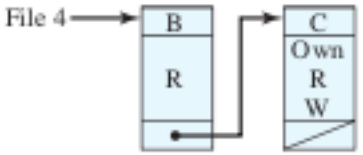
Droits d'accès aux fichiers

Structure logique matricielle: agents/objets, mais c'est une matrice creuse: représentation par colonnes (ACL) ou par lignes (capabilities)

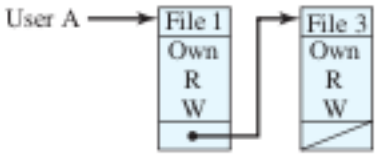


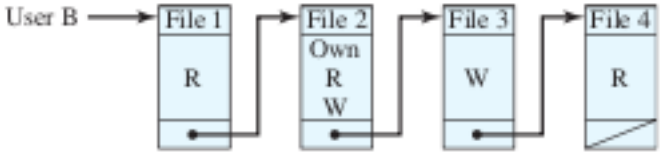


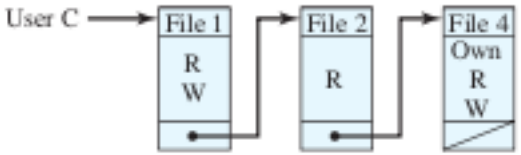




(b) Access control lists for files of part (a)







(c) Capability lists for files of part (a)

Figure 12.13 **Example of Access Control Structures**

Exemple UNIX

Stucture de données pour attributs des fichiers: **inodes**

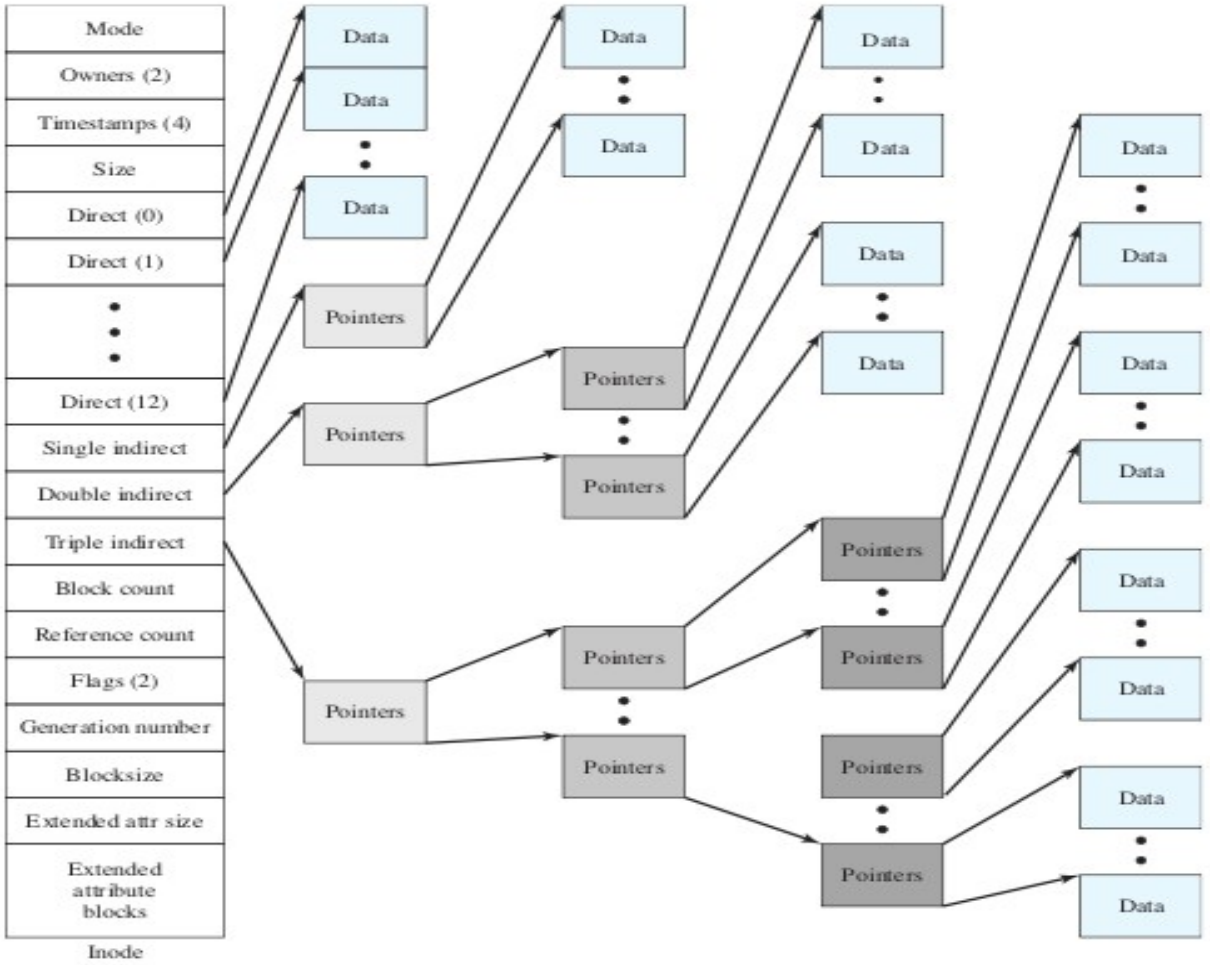


Figure 12.14 Structure of FreeBSD inode and File

Exemple UNIX

Inodes et répertoires

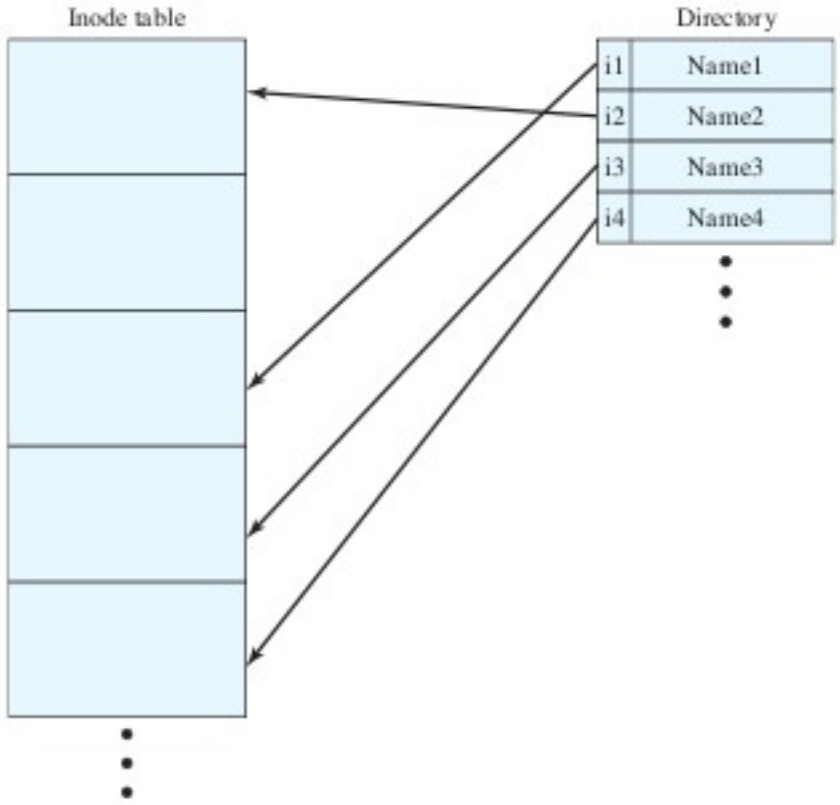


Figure 12.15 UNIX Directories and Inodes

Exemple UNIX

Systèmes de fichiers virtuels

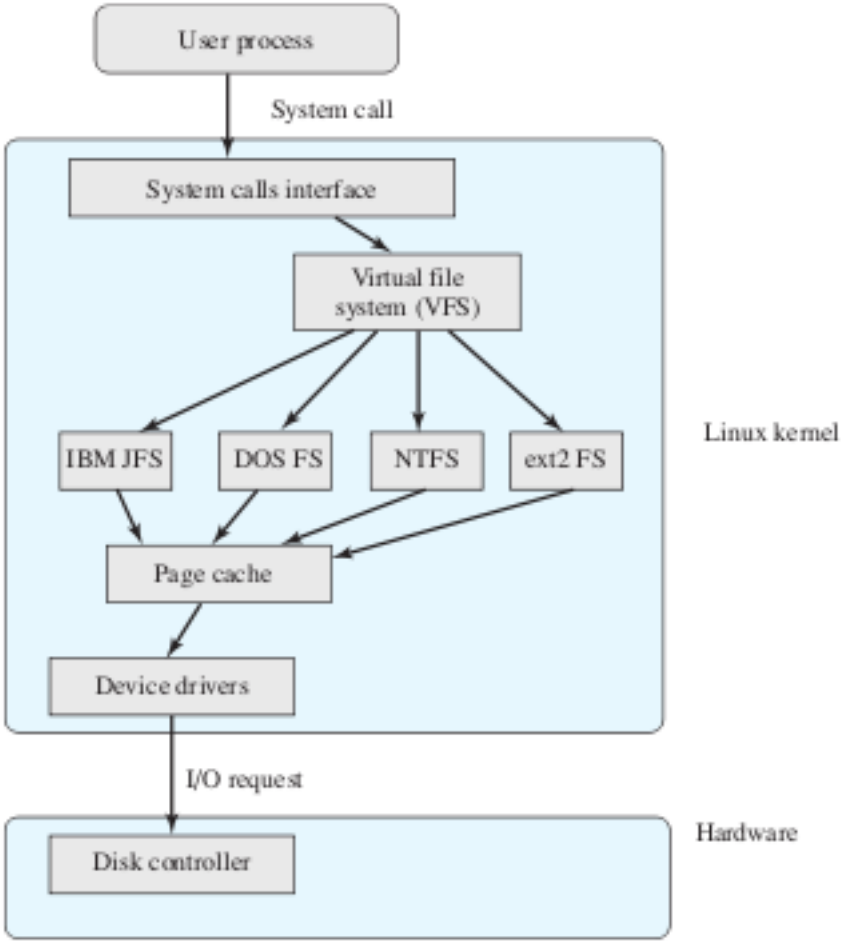


Figure 12.17 **Linux Virtual File System Context**

nouveautés

Systèmes de fichiers journalisés

- Éviter corruptions en cas de crashs
- Basés sur techniques de 2 phase commit
- Dans NTFS (windows), ZFS (solaris)

Systèmes de fichiers de grande taille

- ZFS= Zetabyte file system (données en 128 bits
- NTFS aussi

Raid software inclus (NTFS, ZFS)