Systèmes d'exploitation – Gestion des fichiers

Catalin Dima

Problématique

- Certains processus doivent stocker/avoir accès à une très grande quantité de mémoire
  - Parfois plusieurs ordres de grandeur par rapport à la taille du processus.
- L'information doit survivre à la terminaison du processus qui l'utilise.
- On doit permettre à plusieurs processus d'accéder à l'information, parfois en même temps.
- L'information utilisée par les processus doit être **indépendante** des processus qui l'utilisent.

Solution : **fichiers**.

Fichiers – raisons d'être

- Les programmes/tâches (=**processus**) peuvent demander de stocker/avoir accès à une très grande quantité de mémoire
  - Parfois plusieurs ordres de grandeur par rapport à la taille du programme/tâche.
- L'information doit survivre à la terminaison du processus qui l'utilise.
- On doit permettre à plusieurs processus d'accéder à l'information, parfois en même temps.
- L'information utilisée par les processus doit être **indépendante** des processus qui l'utilisent.
- Enfin, les programmes eux-mêmes doivent être stockés pour être lancés à n'importe quel moment.

Solution : **fichiers**.

Fichiers et systèmes de fichiers

- **Fichier** :
  - Ensemble d'informations en relation entre elles : programmes, données.
  - Unité "logique" de stockage d'information.
- **Le système de fichiers** :
  - L'ensemble des fonctionnalités mises en oeuvre pour la gestion des fichiers dans un SE – partie *essentielle* du système d'exploitation qui gère les fichiers.
- Fonctionnalités d'un système de fichiers :
  - Correspondance entre fichiers et dispositifs physiques = placement sur disques/bandes magnétiques/mémoires flash...
  - Organisation interne et externe des fichiers.
  - Gestion des requêtes pour l'accès aux fichiers.
  - Protection des fichiers.

Caractéristiques des fichiers

- Nommage de fichiers : utilisation d'extensions (`c, o, bak, htm(l), jpg, gif, mp3, pdf, txt, exe` etc.
- Chemin d'accès.
- Structure interne : séquence d'octets/enregistrements, structure arborescente.
- Types de fichiers.
- Accès aux fichiers : séquentiel/aléatoire.
- Attributs des fichiers : exécutable, archive, date de création, etc.
- Opérations sur les fichiers : lecture, modification, lancement en exécution, etc.

Types de fichiers

- Réguliers.
- Répertoire.
- Spécial caractère – terminaux, imprimantes, réseaux.
  - Accès séquentiel, sans possibilité de revenir en arrière.
  - Toujours associés à des variables tampon (buffer).
- Spécial bloc – disques, fichiers résidant en mémoire.
  - Accès aléatoire, on peut se déplacer dans le fichier dans les deux directions.
  - Toujours associés à des variables pointeur de fichier.

Attributs des fichiers

- Attributs de type : archive, exécutable, binaire, ASCII, caché, système, temporaire.
- Attributs quantitatifs : dimension, dimension maximale, date/heure de création ou du dernier accès/modification, nb de clefs, dimension de chaque enregistrement, etc.
- Attributs de gestion : proprietaire, créateur.
- Attributs de protection :
  - Protection entre différents utilisateurs,
  - Protection entre moments de connexion différents.
  - Protection contre utilisation incorrecte ou non-autorisée.

Répertoires

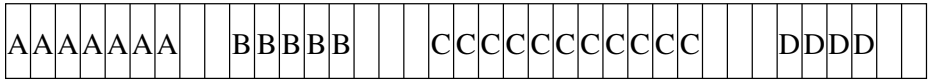
- Conteneurs de fichiers.
- Dans la plupart des SE, organisés en arborescence.
  - Une seule hiérarchie de répertoires et fichiers.
  - Si chargement d'un nouveau volume (floppy, stick USB) : *montage* = connexion de l'arborescence sur le nouveau volume dans l'arborescence existante.
- Mais Windows (≥ 98) et UNIX offrent la possibilité de *partager* des fichiers :
  - Pour rendre l'accès plus facile, il est permis à plusieurs répertoires de "contenir" le même fichier.
  - En fait, on a plusieurs *liens* vers le même fichier.
  - De même pour les répertoires : on peut avoir plusieurs liens vers le même répertoire dans des répertoires différents.
  - *Problème* : possibilités d'avoir des cycles dans le graphe de dépendance.
- Protection des répertoires : nom du proprietaire, droits d'accès.

Allocation des fichiers sur disque

- *Considerations d'architecture* : il est plus rentable de faire les communications entre le processeur/la mémoire et les disques (périphériques de stockage) par "blocs".
- Le problème se pose alors sur les modalités de diviser les fichiers en blocs et de stocker les fichiers sur un disque.
- Le système de fichiers doit avoir un moyen de calculer/mémoriser
  - La decomposition des fichiers en blocs.
  - Le positionnement du début de chaque fichier.
  - Les positions de tous les blocs du même fichier.
  - Tous les blocs de mémoire libre sur un disque.
- Premier souci : comment garder les *blocs libres*
  - Liste de blocs libres.
  - Bitmap.
- Chacune des deux solutions implique utilisation de l'espace disque !

Allocation contigüe et par liste chaînée

- Allocation contiguë :
  - Avantage : accès rapide,
  - Désavantage : fragmentation du disque, nécessite de rouler un "défragmenteur" pour pouvoir trouver de l'espace pour une nouvelle création de fichier.



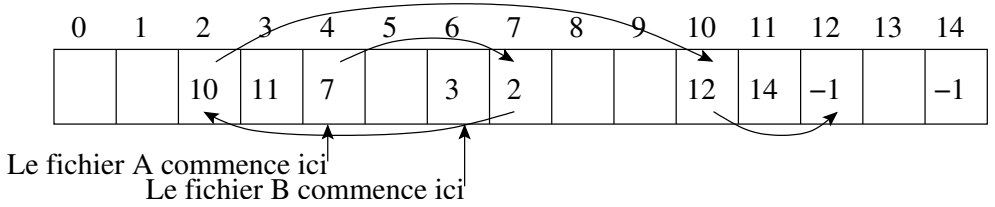
Si on veut créer un fichier de 7 octets, il faut déplacer au moins 2 fichiers.

- Allocation par liste chaînée :
  - Désavantage majeur : temps d'accès prohibitif, car accès séquentiel.

File Allocation Table – FAT

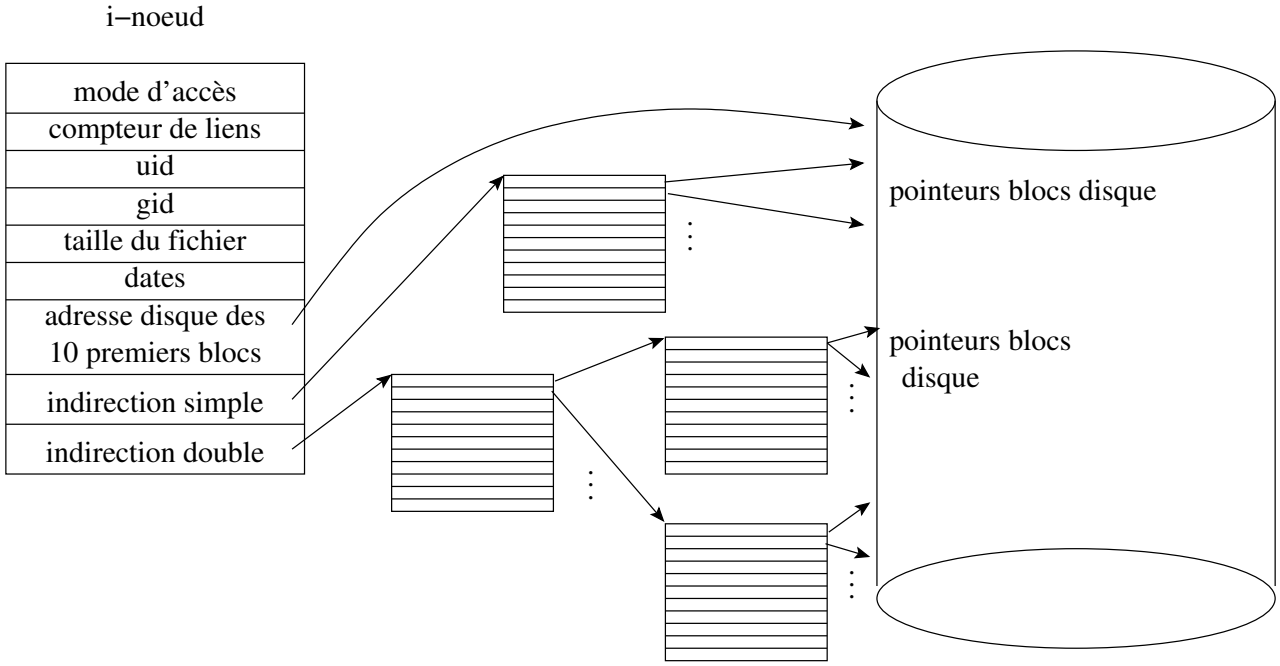
Liste chaînée et indexée.

- Blocs de dimension fixée (MSDOS) ou variable.
- *Avantage* par rapport à la liste chaînée : temps d'accès presque constant – on calcule le numéro du bloc qu'on veut charger en mémoire et on cherche, *dans la FAT* (pas sur disque) l'emplacement du bloc désiré.
- *Désavantage* : le tableau d'indices peut occuper bcp de mémoire, même si on le décompose en blocs.
- Parade possible : ne pas garder toute la FAT en mémoire – mais alors le temps de chargement d'un bloc désiré sera plus grand, puisqu'on se retrouve dans le cas de l'accès séquentiel pour la recherche de l'*indice* correspondant au bloc recherché.



I-nœuds

- Un compromis rapidité d'accès/compacité de la représentation en mémoire.



- Un tableau d'indirection = un bloc sur disque.
- Nb. moyen d'accès au disque pour retrouver un bloc de fichier – supérieur à la FAT.
- On peut avoir 2 ou 3 niveaux d'indirection.

I-nœuds

- Un disque est decomposé en une zone d'i-nœuds et une zone de données.
- On a alors une *table d'i-nœuds* du disque, et les i-nœuds sont identifiés par leur no. dans cette table.

Exercice : Calculer la taille de chaque tableau d'indirection si les blocs sur disque ont 1koctet et l'adressage sur disque se fait sur 32 bits. Calculer la taille maximale d'un fichier sur deux niveaux d'indirection.

I-nœuds et répertoires

- **Répertoire** $\longrightarrow$ i-nœud.
- **Répertoire** = suite d'élements de la forme *(numéro,nom)*.
- *numéro* = le no. de l'i-nœud qui porte l'information sur le fichier *nom*.
- Lors de la demande de blocs d'un fichier donné par un chemin relatif `rep1/rep2/fichier` le système exécute les actions suivantes :
  - cherche, dans le répertoire courant, l'entrée du nom `rep1`.
  - charge en mémoire son i-nœud et cherche, dans celui-ci, l'entrée du nom `rep2`.
  - charge en mémoire son i-nœud et cherche, dans celui-ci, l'entrée du nom `fichier`.
  - enfin, cherche dans les tables de ce dernier i-nœud, l'adresse du bloc demandé.

Liens en Unix

- **Lien dur** vers un fichier : entrée dans le répertoire indiquant le même i-nœud qu'une autre entrée.
  - Compteur de liens dans l'i-nœud.
  - Création de lien : commande `ln`.
  - Destruction de lien : `rm`, qui décremente le nb. de liens dans l'i-nœud du fichier !
- **Lien symbolique** :
  - Type particulier de fichier, qui *occupe un i-nœud* et qui contient seulement le chemin d'accès du répertoire (ou fichier) pointé.
  - Permet d'avoir le même répertoire en tant que sous-répertoire de différents répertoires.

Problèmes posés par les liens

- Chaque i-nœud possède un compteur de liens durs sur le fichier/répertoire pointé.
- Création de liens durs – refusée pour les répertoires (pourquoi ?).
- Suppression des liens
  - durs – décrement du nb. de liens ; mais que se passe-t-il si le fichier est partagé par des utilisateurs différents ?
  - symboliques – si on supprime le fichier, alors l'i-nœud va pointer vers un fichier qui n'existe plus.
- Surcoût dû aux liens symboliques : la recherche d'un fichier à travers des liens symboliques est plus lente qu'à travers des liens durs.
- "Boucles infinies" possibles avec les liens symboliques (pb. pour des scripts de recherche de fichiers).

Implémentation des répertoires

Problématique générale

- Où stocker les attributs :
  - dans le répertoire (FAT),
  - dans les entrées du fichier même (i-nœuds).
- Taille de noms de fichiers
  - fixe (MSDOS) – problèmes de gestion de noms.
  - non-limitée (Windows, UNIX).
- Stockage des entrées dans un répertoire et recherche des noms
  - Séquentiel : inéfficace pour des longs répertoires.
  - *Table de hachage* : utiliser une table de dimension fixe n dont chaque entrée peut être le début d'une liste d'entrées.

  Problèmes de gestion d'attributs : e.g. si les liens appartiennent à des utilisateurs différents (et on veut effacer des liens).
- Implémentation des *quotas*.

Problèmes de fiabilité

- **Backup** = sauvegarde des fichiers.

  Gros consommateur de temps et d'espace.
  - Quoi sauvegarder ?
    - Pas les fichiers temporaires/spéciaux
    - Pas de sauvegarde multiple pour les liens.
    - Parfois seulement certains blocs sont modifiés...
  - *Corbeille* (Windows) – garde-fous.
  - *Sauvegardes incrémentales* – seulement les fichiers ayant été modifiés.
  - Problèmes de sécurité.
  - Sauvegardes physiques/logiques.
- Cohérence du système des fichiers.
  - Windows/MSDOS : `scandisk`.
  - UNIX : `fsck`.
  - Vérification de la consistance des blocs/fichiers.

Vérification de la cohérence

Cohérence des blocs :

- On construit deux tableaux de bits de la taille du nb. de blocs :
  - nb. d'apparitions d'un bloc dans un fichier.
  - nb. d'apparitions d'un bloc dans la liste des blocs libres.
- Normalement, chaque bloc doit apparaître dans une seule des deux listes.
- *Blocs manquants* : n'apparaissent dans aucune des 2 listes. Solution : refaire la liste des blocs libres.
- *Duplication dans la liste des blocs manquants* – même solution.
- *Duplication dans la liste des blocs utilisés* ; solution : duplication du bloc.

Cohérence des fichiers :

- Tableau indexé sur les fichiers = dans combien de répertoires un fichier apparaît-il.
- On ne compte pas les liens symboliques.
- Nb. d'apparitions d'un fichier $\neq$ nb de liens dans son i-nœud – on met à jour le nb. de liens dans l'inœud.