Chapitre 2

Le système de gestion de fichiers

1. Qu'est-ce qu'un système de gestion de fichiers?

Le système de gestion de fichiers est le composant d'un système d'exploitation responsable de la gestion (stockage, lecture, écriture, etc.) des données des utilisateurs sous forme de fichier enregistré sur les supports de stockage de manière permanente (généralement sur le disque dur).

2. Rôle d'un système de gestion de fichiers

Le système de gestion de fichiers assure les fonctionnalités suivantes :

- Le stockage des données des utilisateurs sous forme de fichiers sur les supports de stockage (disque dur, etc.).
- La mise en œuvre d'un certain nombre de manipulations telles que la structuration de l'ensemble des données, l'insertion d'éléments, etc.
- la gestion de l'espace libre
- la gestion des accès simultanés aux données

3. La structure d'un système de fichiers

Un **fichier** est une unité de stockage logique mise à la disposition des utilisateurs pour l'enregistrement de leurs données. Le SGF établit de façon transparente aux utilisateurs la correspondance entre le fichier et les bits enregistrés dans le support de stockage.

Dans un fichier on peut écrire des images, du texte, des instructions d'un programme, etc. Ces données sont structurées logiquement sous forme de fichier et physiquement sous forme de bits.

Afin de différencier les fichiers, chacun possède un ensemble d'attributs comme le nom, l'extension, la date de création, la dernière date de modification, la taille.

Un **répertoire** est une unité logique qui permet de stocker plusieurs fichiers. Il ne contient pas directement des données, il contient des fichiers et ce sont les fichiers qui contiennent des données.

Les répertoires sont, eux aussi, des fichiers, constitués des noms et des références de tous les fichiers qu'ils contiennent. Cette structure permet alors de construire l'arborescence du système. Pour désigner un fichier quelconque, il suffit de spécifier l'enchaînement des répertoires nécessaires à son accès, à partir de la racine. Dans le système Unix, les répertoires de cet enchaînement sont séparés par une oblique : « / ». Dans le système DOS, par une contre-oblique : « \ ».

Dans le système Unix, chaque répertoire contient aussi sa propre référence, ainsi que celle du répertoire immédiatement supérieur. « **.** » désigne le répertoire courant, et « **..** », le répertoire supérieur. L'inclusion de ces deux références permet de désigner un fichier quelconque, relativement au répertoire courant.

Les systèmes d'exploitation modernes adoptent **une structure hiérarchique** des fichiers. Chaque fichier appartient à un groupe d'autres fichiers et chaque groupe appartient lui-même à un groupe d'ordre supérieur. On appelle ces groupes, des répertoires ou des dossiers, suivant les terminologies.

Le schéma de la structure générale d'un système de fichiers prend l'aspect d'un arbre, formé au départ d'un répertoire « racine » recouvrant des périphériques et notamment un ou plusieurs disques. Dans chacun des répertoires on pourra trouver d'autres répertoires ainsi que des fichiers de données ordinaires.

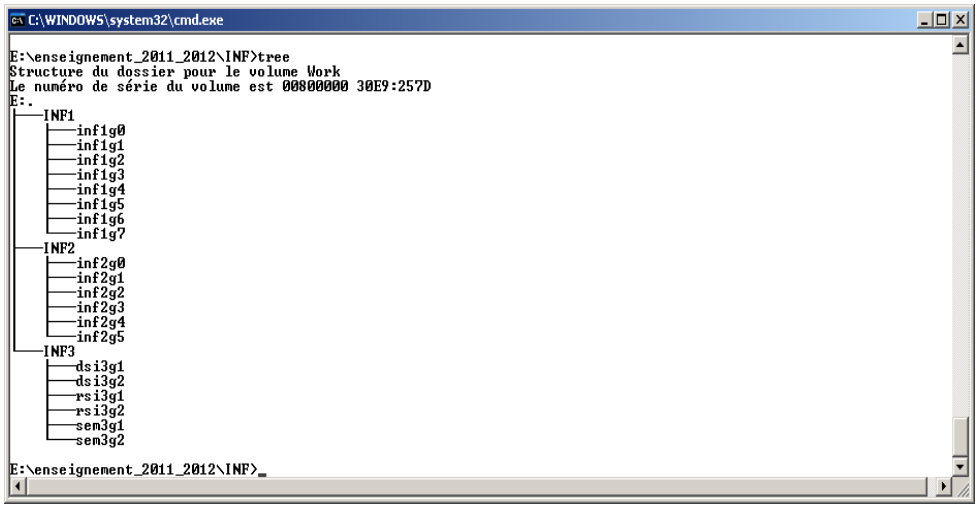


Figure 1. Résultat de la commande « tree » sous windows

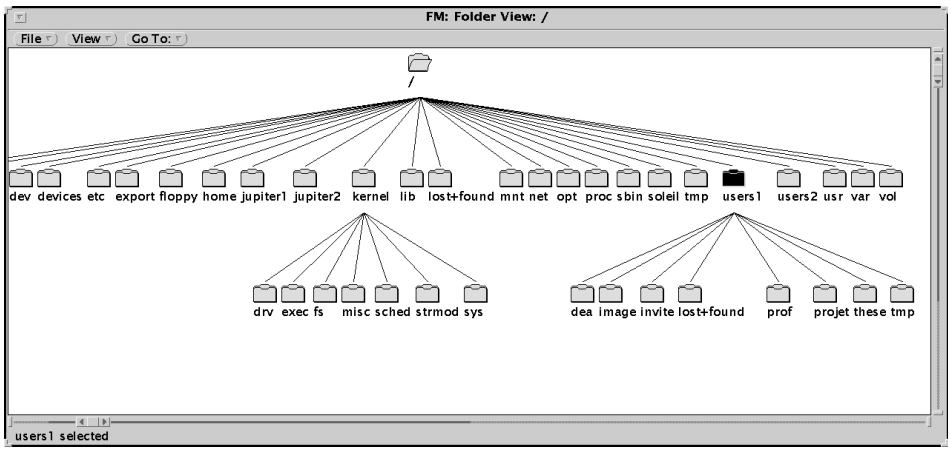


Figure 2 Une « vue » du gestionnaire de fichiers d'*Open Look*

4. Anatomie d'un disque dur

Un disque dur possède la forme suivante:

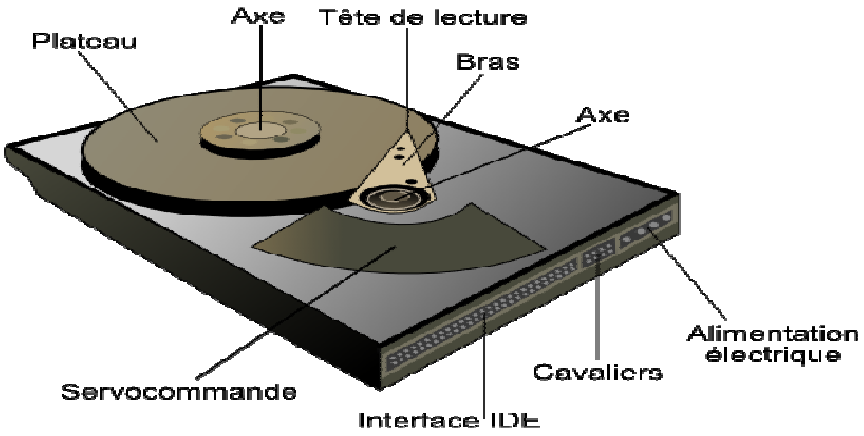


Figure 3. Structure d'un disque dur

Un disque dur est constitué des composants suivants:

Les plateaux : un plateau est le support destiné à accueillir les données. Les plateaux ont une forme circulaire et ils sont empilés les unes au-dessus des autres. Le diamètre d'un plateau indique la taille d'un disque (3 pouces et demi ou 5 pouces et ¼) généralement le nombre des plateaux n'excède pas une dizaine.

Les têtes de lecture-écriture : sur chaque plateau, on peut écrire les données sur les deux faces. Il faut ainsi associer une tête à chaque face. Les têtes sont reliées à un bras qui permet de les déplacer simultanément. Le bras se déplace de l'extérieur vers le centre et vice-versa. Chaque tête est collée à la face du plateau lorsque le disque est au repos. Quand la machine est mise sous tension, le disque entreprend un mouvement de rotation permanent et il se crée une dépression sous chaque tête l'obligeant à s'éloigner du plateau de qqs millimètres.

Un moteur rotatif : il permet de faire tourner les plateaux à une vitesse de 7200 tr/min et même plus.

Des positionneurs de tête : ils permettent de positionner la tête de lecture/écriture à l'emplacement voulu. Chaque face du plateau est constituée d'un ensemble de pistes

La surface de chaque plateau est divisée en pistes concentriques numérotées à partir de 0 en commençant à partir de la piste la plus à l'extérieur. Chaque piste est divisée en secteurs.

5. Le formatage et le partitionnement

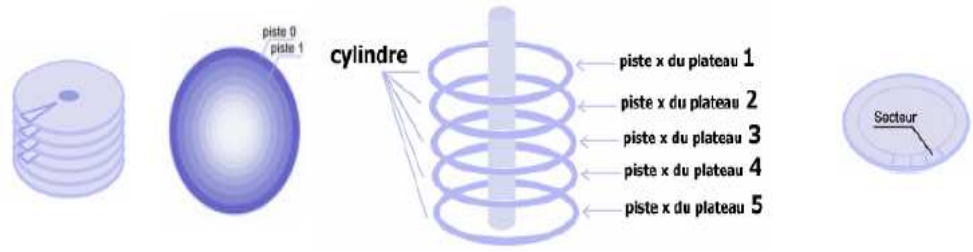


Figure 4. Formatage de bas niveau

5.1. Le formatage

Il existe deux types de formatage.

Le formatage de bas niveau

Il est réalisé en usine et consiste à tracer les pistes et les secteurs sur les plateaux. Des informations supplémentaires sont créées au niveau de chaque secteur pour qu'il puisse être correctement interprété et lu. Une piste n'est pas divisée en secteurs de 512 octets mais en zones de plus grande taille incluant le secteur.

Marque de début	Numéro du cylindre	Numéro du secteur	Données (512 octets)	Code vérif. Erreurs	Autres...

Figure 5. Structure d'un secteur

Le formatage de haut niveau

Le formatage de haut niveau consiste à organiser les pistes et les secteurs d'une manière compréhensible par le système d'exploitation. Avant d'utiliser un disque ou disquette il faut la formater et nul n'ignore qu'un disque formaté sous Unix ne peut être lu avec DOS car chaque système organise ses pistes et secteurs à sa manière.

Un système peut décider que l'unité d'allocation n'est pas le secteur mais un ensemble de secteurs. Lors du formatage, le SE crée les structures qu'il va utiliser pour gérer et organiser les fichiers et les répertoires. Cette structure sera enregistrée dans le disque lui-même.

5.2. Le partitionnement

Le partitionnement d'un disque consiste à le diviser en partitions. La partition est une zone du disque qui peut être considéré comme un disque logique à part. On parle aussi de lecteur logique.

Chaque partition peut recevoir un SE différent, pour il suffit que ce disque soit amorçable et que le programme d'amorçage du système soit placé dans le secteur de boot.

Pour partitionner un disque on peut utiliser la commande DOS FDISK ou un logiciel comme « partition magic ».

Une fois le disque partitionné, chaque partition doit être formatée pour le SE qui va la gérer. Donc pour installer un SE dans une partition, il faut qu'elle soit formatée selon le système. Exemple pour installer XP dans une partition il faut la formater avec FAT32 ou NTFS, pour LINUX il faut la formater ext2,ext3 ou swap.

Mis à part le démarrage avec un bloc de boot, l'organisation d'une partition de disque varie fortement d'un système de fichiers à l'autre. Le système de fichiers comprendra souvent quelques-uns des items présentés à la figure suivante. Le premier d'entre eux est le superbloc. Il contient tous les paramètres clés concernant le système de fichiers et est mis en mémoire quand l'ordinateur démarre ou quand le système de fichiers est modifié.

Viennent ensuite des informations sur les blocs libres du système de fichiers, par exemple sous la forme de tables de bits ou de listes chaînées. Elles sont suivies des i-nodes, un tableau de structures de données, une par fichier, donnant toutes les informations nécessaires d'un fichier. Enfin, on trouve le répertoire racine, qui contient le haut de l'arborescence du système de fichiers et, pour terminer, le reste du disque qui renferme tous les répertoires et les fichiers.

Les systèmes de fichiers les plus connus pour Windows sont:

- FAT 16: (File Allocation Table) compatible avec Ms-DOS, les numéros de clusters (unités d'allocation) s'écrivent sur 16 bits

- FAT 32 : pris en charge dès Windows95. Les numéros des clusters s'écrivent sur 32 bits et les partitions peuvent atteindre 2TO. (1 tétra O=1024GO)

- NTFS (NT File System) : il est conçu pour gérer les disques avec une capacité supérieure à 400 MO. Il autorise les noms de fichiers longs.

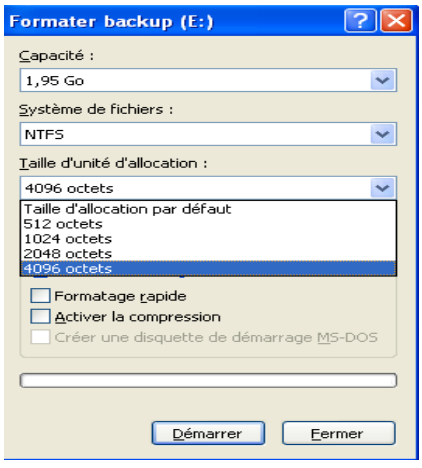


Figure 6. Formatage avec Windows

5.3. Accès aux éléments d'un fichier

Les fichiers de données sur disques se composent d'un ensemble de blocs, comprenant un nombre fixes d'octets. La première structure possible de ces fichiers correspond à la suite des octets des blocs; ces blocs étant ordonnés. L'accès à un octet se fait alors par un déplacement à partir de l'origine du premier bloc : le début du fichier. Le système Unix, ainsi que le système DOS ne connaissent que cet accès.

Certains systèmes structurent les fichiers sous la forme d'enregistrements séquentiels de tailles fixes, l'ensemble des blocs étant toujours ordonné linéairement. Ce principe est à peu près le même que le précédent. Il permet de lire, de substituer un élément ou d'en ajouter d'autres à la fin, mais on ne peut pas en détruire ou en insérer de nouveaux au milieu du fichier.

L'accès séquentiel indexé est une méthode plus élaborée, mise en œuvre, entre autres, sur les systèmes MVS d'IBM. On structure les blocs des fichiers sous la forme d'un arbre. Cette structure permet l'insertion d'éléments à n'importe quel endroit d'un fichier.

5.4. Les blocs du disque

Les fichiers de données sur les disques se répartissent dans des blocs de taille fixe correspondant à des unités d'entrées-sorties du contrôleur de ce périphérique. La lecture ou l'écriture d'un élément d'un fichier impliquera donc le transfert du bloc entier qui contient cet élément. On peut formater les disques, – les rendre utilisables par le système d'exploitation –, avec une taille particulière de blocs. Cette taille résulte d'un compromis entre la vitesse d'accès aux éléments des fichiers et l'espace perdu sur le disque.

Lors d'un transfert de données d'un disque vers l'espace d'adressage d'un processus, le temps de lecture ou d'écriture sur le disque est négligeable devant le temps d'accès au bloc, et ceci quelle que soit la taille du bloc. Pour un accès rapide, on aura donc intérêt à prendre des blocs de grande taille. Cependant, les fichiers, y compris les fichiers de 1 octet, ont une taille minimale de 1 bloc. Si un disque comprend beaucoup de fichiers de petite taille et si les blocs sont de grandes dimensions, l'espace gaspillé sera alors considérable.

Chaque disque conserve, dans un ou plusieurs blocs spécifiques, un certain nombre d'informations de fonctionnement, telles par exemple que le nombre de ses blocs, leur taille, … Il mémorise aussi en général l'ensemble de ses blocs ainsi que leur état dans une table. Si cette table est binaire, un disque de n blocs devra alors réserver une table de n bits; la position de chaque bit indiquant si le bloc est libre ou s'il est utilisé par un fichier, par exemple, 0 pour un bloc occupé et 1 pour un bloc libre. Certains systèmes stockent l'ensemble des blocs libres dans une liste chaînée.

6. La répartition physique des fichiers en blocs

À chaque fichier correspond une liste de blocs (ou clusters ou unités d'allocation) contenant ses données.

L'allocation de ces blocs peut être contiguë (le fichier est enregistré sur des blocs consécutifs sur le disque) ou non-contiguë.

L'allocation est en générale non contiguë et les blocs sont donc répartis quasi-aléatoirement sur le disque.

Les deux méthodes d'allocation les plus utilisées par les systèmes d'exploitation sont: la liste chaînée utilisant une table en mémoire et les nœuds d'index (i-nodes).

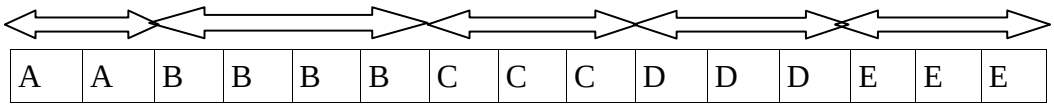
6.1. Allocation contiguë

La méthode d'allocation la plus simple consiste à stocker chaque fichier dans une suite de blocs consécutifs. Ainsi, dans un disque avec des blocs de 1 Ko, un fichier de 50 Ko se verra allouer 50 blocs consécutifs. Dans un disque avec des blocs de 2 Ko, ce fichier aura 25 blocs consécutifs.

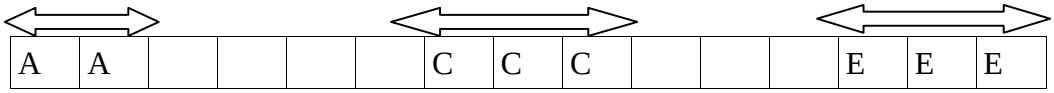
Pour chaque fichier à enregistrer le SE recherche une zone suffisamment grande pour accueillir le fichier. Le fichier sera constitué de plusieurs blocs contigus. L'entrée d'un répertoire dans un SE qui applique l'allocation contiguë contient l'adresse du premier bloc et la longueur de fichier.

En apparence le principal avantage de cette méthode est la simplicité et la rapidité lors de l'accès.

Malheureusement, l'allocation contiguë a aussi un inconvénient majeur : la fragmentation avec le temps de l'espace libre. Pour comprendre ce phénomène, examinons cette figure.



(a) avant suppression de B et D



(b) après suppression de B et D

Figure 7. Problème de fragmentation de l'espace libre

Lorsque les fichiers B et D sont supprimés, ils gardent des espaces libres qui ne peuvent être récupérés pour des gros fichiers. Dans ce cas on ne peut pas sauvegarder un fichier F de 5 blocs pourtant on a 7 blocs libres. Il faut compacter le disque (défragmenter le disque) pour déplacer les fichiers au début du disque (ou de la partition). Cette opération est très couteuse en temps et ressources.

6.2. La liste chaînée

L'ensemble des blocs d'un fichier peut être chaîné sous la forme d'une liste. Chaque bloc contiendra des données ainsi que l'adresse du bloc suivant. Le fichier devant mémoriser indépendamment le numéro du 1er bloc. Par exemple, si un bloc comporte 1024 octets et si le numéro d'un bloc se code sur 2 octets, 1022 octets seront réservés aux données et 2 octets au chaînage du bloc suivant.

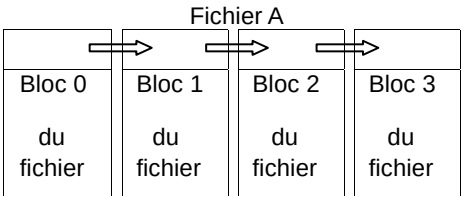


Figure 8. Allocation par liste chaînée

Cette méthode rend l'accès aléatoire aux éléments d'un fichier particulièrement inefficace lorsqu'elle est utilisée telle quelle. En effet pour atteindre un élément sur le bloc *n* d'un fichier, le système devra parcourir les *n-1* blocs précédents.

Par ailleurs, la quantité de données stockées dans un bloc n'est pas en puissance de deux, parce que le pointeur prend quelques octets. Même si le fait d'avoir une taille particulière n'est pas spécialement néfaste, c'est en tout cas moins efficace puisque que de nombreux programmes lisent et écrivent dans des blocs dont la taille est en puissance de deux. Lorsque les premiers octets de chaque bloc sont occupés par le pointeur sur le prochain bloc, la lecture d'un bloc complet nécessite l'acquisition et la concaténation de deux blocs du disque, d'où un temps supplémentaire pour la copie.

6.3. Allocation par liste chaînée utilisant une table en mémoire

Les deux inconvénients d'une liste d'allocation chaînée peuvent être éliminés en prenant le pointeur de chaque bloc du disque pour le ranger dans une table en mémoire. La Figure 9 montre à quoi ressemble la table de l'exemple de la figure 8. Dans les deux figures, nous avons deux fichiers. Le fichier A utilisé les blocs 4, 7, 2, 10 et 12 dans cet ordre, et le fichier B utilise les blocs 6, 3, 11 et 14 dans cet ordre. En nous aidant de la Figure 9, nous pouvons démarrer avec le bloc 4 et suivre la chaîne jusqu'à la fin. Nous pouvons procéder de même avec le bloc 6. Les deux chaînes se terminent avec un caractère spécial (-1 par exemple) qui n'est pas un numéro de bloc valide. Une telle table en mémoire principale est appelée FAT (File Allocation Table, table d'allocation des fichiers).

Grâce à cette organisation, le bloc est disponible dans sa totalité pour les données. De plus, l'accès aléatoire est facilité. Comme la chaîne est intégralement en mémoire, bien qu'il faille encore la parcourir pour trouver un déplacement donné à l'intérieur du fichier, cela peut être réalisé sans faire de nouveaux accès disque.

Comme dans la première méthode, il suffit, pour l'entrée du répertoire, de conserver un entier unique (le numéro du bloc de départ) ; il sera toujours possible de localiser tous les blocs, quelle que soit la taille du fichier.

Le principal inconvénient de cette méthode est que la totalité de la table doit se trouver en mémoire tout le temps. Avec un disque de 20 Go et une taille de bloc de 1 Ko, la table contient 20 millions d'entrées, c'est-à-dire une pour chacun des 20 millions de blocs. Chaque entrée doit avoir un minimum de 3 octets, et même de 4 octets si l'on veut que la recherche soit rapide. Ainsi, cette table prendra entre 60 Mo et 80 Mo de mémoire principale en permanence, suivant que le système est optimisé pour l'espace ou pour le temps. Cette table pourrait être placée dans une mémoire paginée, mais elle occuperait une grande partie de la mémoire virtuelle et de l'espace disque, et engendrerait encore plus de chargements de pages.

Le système MS-DOS utilise des listes chaînées. Il conserve le premier bloc de chacun des fichiers dans son répertoire. Il optimise ensuite l'accès des blocs suivants en gardant leurs références dans une Table d'Allocation de Fichiers (FAT). Chaque disque dispose d'une FAT et cette dernière possède autant d'entrées qu'il y a de blocs sur le disque. Chaque entrée de FAT contient le numéro du bloc suivant.

Avec la table suivante[1] :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…
X	X	EOF	13	2	9	8	L	4	12	3	L	EOF	EOF	L	BE	…

Figure 9. Allocation par liste chaînée utilisant une table en mémoire

où « XX » indique la taille du disque, « L » désigne un bloc libre et « BE » un bloc endommagé, le fichier commençant au bloc 6, sera constitué des blocs : 6 → 8 → 4 → 2.

[1] Reprise de Tanenbaum, op. cit., p. 288.

Le parcours de la FAT est nettement plus rapide que la chaîne des blocs. Cependant si elle n'est pas constamment, tout entière en mémoire, elle ne permet pas d'éviter les entrées-sorties du disque.

6.4. Le nœud d'index

Le système Unix répertorie chaque fichier par un numéro unique pour tout un disque. À chaque numéro, correspond un enregistrement, un nœud d'index, (i-node), comportant un nombre fixé de champs. Parmi ces champs, certains, 13 au total, mémorisent l'emplacement physique du fichier. Les 10 premiers champs (sur les 13) contiennent les numéros des 10 premiers blocs composant le fichier.

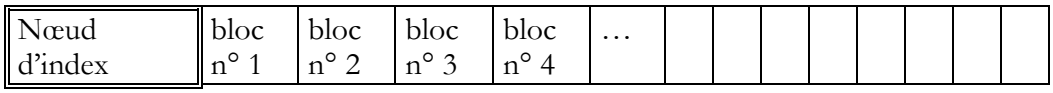


Figure 10. Les i-nodes

Pour les fichiers de plus de 10 blocs, on a recours à des indirections. Dans le cas du système Unix, le bloc n° 11 contient le numéro d'un bloc composé lui-même d'adresses de blocs de données. Si les blocs ont une taille de 1024 octets et s'ils sont numérotés sur 4 octets, le bloc n° 11 pourra désigner jusqu'à 256 blocs. Au total, le fichier utilisant la simple indirection aura alors une taille maximale de 266 blocs. De la même manière, le bloc n° 12 contient une adresse, un pointeur, à double indirection, et le bloc n° 13, un pointeur à triple indirection. Avec l'exemple que nous avons donné, un fichier peut avoir une taille maximale de 16 Go quand il utilise ces pointeurs à triple indirection.

Bloc n° 11	Bloc n° 12	Bloc n° 13
Pointeur à simple indirection	Pointeur à double indirection	Pointeur à triple indirection

Figure 11 . Bloc de redirection dans les i-nodes

7. Les dossiers ou les « répertoires »

Les répertoires des systèmes de fichiers hiérarchiques sont des fichiers dont le contenu est particulier. Ils permettent de référencer et de retrouver physiquement les fichiers immédiatement en dessous d'eux dans la hiérarchie. La structure la plus commune des répertoires prend la forme d'un arbre, que nous avons illustré au paragraphe 1. On peut parfois mettre en œuvre des structures plus complexes, de graphes acycliques ou de graphes généralisés.

Un graphe permet à deux répertoires de référencer le même fichier. On a ainsi le moyen de partager un fichier entre, par exemple, deux utilisateurs. En général, on construit les répertoires de manière à éviter les cycles :

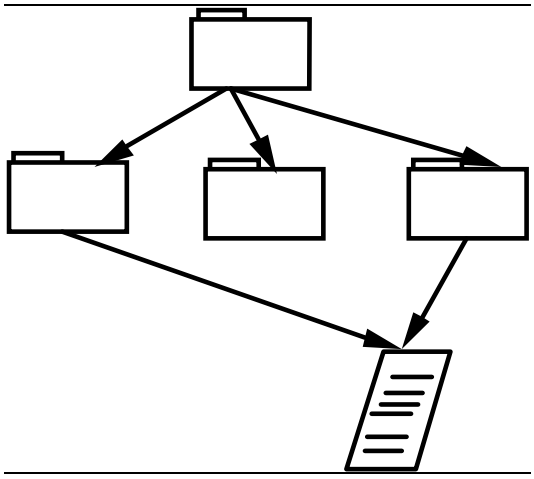


Figure 12

Cependant, on peut parfois construire des structures cycliques, telle que celles-ci :

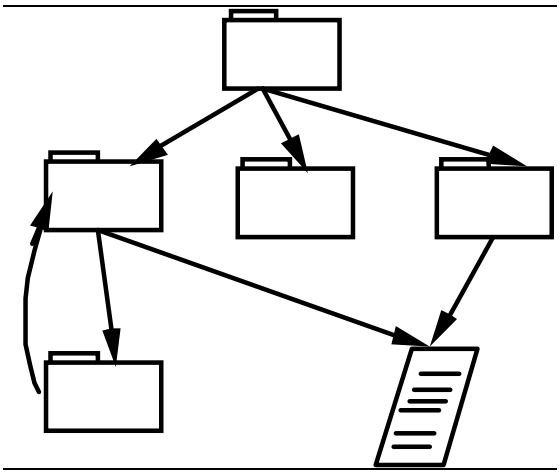


Figure 13

7.1. Les répertoires du système MS-DOS

Les répertoires du système MS-DOS possèdent une entrée par fichier et chaque entrée à la structure suivante :

Nom du fichier	Extension	Attributs	Réservé	Heure	Date	N° du 1er bloc	Taille

Figure 14

Le numéro du premier bloc, *cluster* dans la terminologie MS-DOS, indexe la Fat et permet de retrouver la suite des éléments du fichier.

7.2. Les répertoires d'Unix

Les répertoires du système Unix disposent, eux aussi, d'une entrée par fichier. Chaque entrée possède au moins les deux champs suivants, le numéro du nœud d'index et le nom du fichier. Par exemple :

26	.
59	..
267	toto
534	titi

Figure 15

La structure exacte d'une entrée varie suivant les versions d'Unix. Dans la version *System V*, elle est décrite par le fichier d'en-tête `/usr/include/sys/dir.h`.

On peut visualiser, et interpréter, le contenu du répertoire courant, par exemple, grâce à la commande : « `od -c .` »

7.3. Structure et manipulation des nœud d'index

Le numéro du nœud d'index renvoie à un enregistrement sur le disque contenant les numéros des blocs, comme on l'a présenté au paragraphe précédent. Cet enregistrement contient aussi des informations, concernant la gestion des fichiers[2], dont les principales sont les suivantes :

Nœud d'index
Uid du propriétaire
Gid du propriétaire
Type du fichier
Permissions
Nombre de liens
Taille du fichier
Date de création
Date du dernier accès
Date de dernière modif.
13 numéros de blocs et de pointeurs …

Figure 16

Lorsqu'on manipule des fichiers, les nœuds d'index correspondants, auxquels s'ajoutent quelques champs supplémentaires, sont chargés en mémoire. Parmi les

---

[2] Bach, p. 64, op. cit.

champs supplémentaires, on trouve notamment divers indicateurs, de verrouillage du nœud en mémoire, d'attente de processus sur le nœud, de modification en mémoire, du nœud ou du fichier, non encore reportée sur le disque. Le nœud en mémoire contient aussi, le numéro du périphérique (du disque) sur lequel se trouve le fichier, le numéro du nœud d'index, le nombre de fois où le fichier a été chargé en mémoire (le nombre d'ouvertures en cours)[3].

Pendant l'exécution d'un appel système concernant un fichier, le nœud d'index se verrouille. Deux processus ne peuvent donc pas effectuer, simultanément, une opération sur le même fichier. En cas de conflit, l'un des deux doit attendre. Ceci évite que le nœud ne se retrouve dans un état incohérent. Cependant ce nœud n'est pas verrouillé entre les appels système et des processus peuvent donc se partager un fichier ouvert. Si un utilisateur désire un accès exclusif, il devra mettre en œuvre, par exemple, un sémaphore.

La position courante – de lecture et d'écriture – des processus dans un fichier, n'est pas contenue dans le nœud d'index, mais dans une table séparée, globale à tous les processus. Cette organisation est nécessaire pour que deux processus puissent lire ou écrire à des endroits différents du fichier. On repère les entrées de cette table par les descripteurs des fichiers[4], rendus par `open`, et que manipule `read`, `write`, … En revanche, la position courante est partagée par les fils d'un processus, lorsque le fichier a été ouvert avant la création de ces fils. Les fils se réfèrent alors au fichier par le même numéro de descripteur. Grâce à ces propriétés, on peut, par exemple, établir une communication à travers un tube.

---

[3] Bach, p. 65, op. cit.

[4] Ces descripteurs de fichiers indicent une table privée (propre à chaque processus) dont les éléments pointent sur la cellule de la table globale où se trouve la position courante.

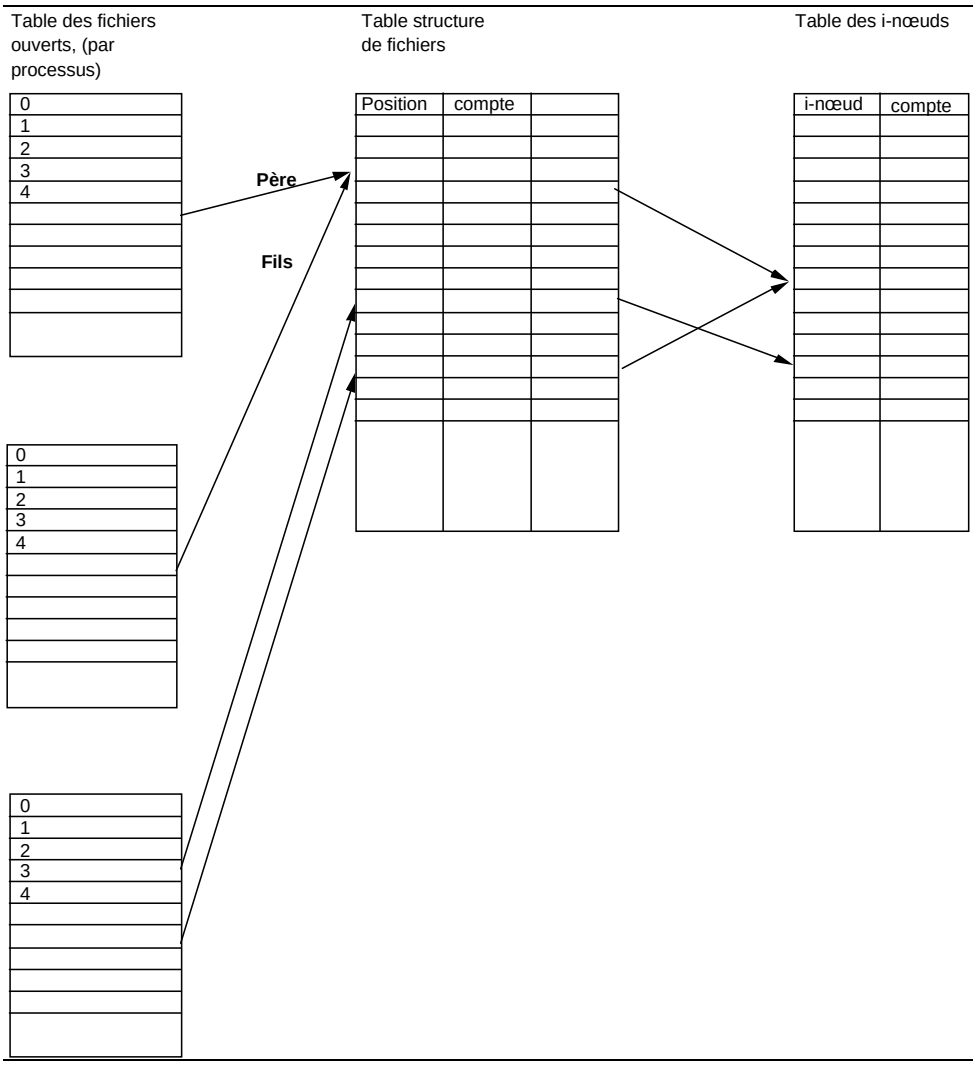


Figure 17

7.4. Le partage de fichiers par liens avec Unix

Un mécanisme permet de construire des structures de répertoires ayant la forme de graphes acycliques orientés et ainsi de partager des fichiers. Pour cela, plusieurs répertoires doivent référencer le même nœud d'index, éventuellement sous un nom différent. On appelle chaque nouveau référencement d'un nœud, la création d'un lien sur le fichier correspondant.

Une fois les liens établis, le fichier pourra être désigné sous l'un quelconque des noms. Pour sa part, le nœud d'index conservera, dans un de ses champs, le

nombre de fichiers qui le référence, indépendamment des protections sur le fichier. L'ajout d'une référence incrémentera le nombre de liens du nœud. L'élimination d'un fichier par un utilisateur décrémentera ce nombre. La destruction effective du nœud, et par là du fichier, aura lieu lorsque le compte des liens sera nul.

8. La mémoire cache

Les processus ne peuvent pas manipuler directement les données du disque. Ils sont obligés, pour cela, de les déplacer en mémoire centrale. Les transferts s'effectuent par blocs et il est souvent inutile d'effectuer autant d'entrées-sorties sur disque que d'accès au fichier. La plupart des systèmes de fichiers gèrent les entrées-sorties grâce à une mémoire intermédiaire : la mémoire cache ou l'antémémoire. Cette mémoire cache fait correspondre des blocs tampons en mémoires aux blocs du disque.

La stratégie générale d'utilisation de la mémoire cache est la suivante : on lui alloue un certain nombre de blocs en mémoire centrale. Lorsque l'on accède à un élément d'un fichier, on examine la suite de ces blocs. Si le bloc désiré – celui qui contient l'élément du fichier – se trouve dans la mémoire cache, on pourra y lire ou y écrire directement l'élément, sinon, on produira un bloc libre de la mémoire cache, on y chargera le bloc du disque et on effectuera les opérations de lecture ou d'écriture. Un bloc – un tampon – pourra être partagé dans la mémoire cache par plusieurs processus et il ne pourra s'y trouver qu'en un seul exemplaire.

Dans le système Unix[5], les blocs tampons sont reliés par un double chaînage des blocs libres et des blocs occupés. Ils possèdent, d'autre part, un en-tête indiquant le numéro du périphérique, le numéro du bloc, ainsi que l'état du tampon :

[5] Bach, chap. 3, op. cit.

N° du périph.	N° du bloc	État	Ptr sur le bloc de données	Ptr sur tampon occupé suiv.	Ptr sur tampon occupé préc.	Ptr sur tampon libre suiv.	Ptr sur tampon libre préc.

Figure 18

Le système Unix ordonne ses tampons suivant l'ordre dernière utilisation. En cas de demande de chargement d'un bloc du disque, il éliminera le moins récemment utilisé (algorithme LRU). Un codage par hachage accélère l'accès.

Il existe plusieurs politiques de recopie des blocs de la mémoire cache sur le disque lorsqu'ils ont été modifiés, en fonction d'un compromis entre la sécurité et la vitesse. Dans le système Unix, les nœuds d'index et les répertoires sont recopiés immédiatement après leur modification. Les blocs de données ordinaires sont recopiés si leur tampon atteint la fin de la liste LRU ou automatiquement par un démon toutes les 15 secondes. On peut déclencher ce démon par l'appel système par l'appel de la fonction `sync()`. Les tampons du système MS-DOS sont à recopie immédiate.

9. La structure physique d'un disque Unix

Chaque disque physique au format Unix possède la structure suivante :

Bloc démarrage	Super bloc	Table bits i-nœuds	Table bits blocs données	nœuds d'index (plusieurs blocs…)	Données (plusieurs blocs…)

Figure 19

Le bloc de démarrage contient le code nécessaire à tout disque pour se lancer.

Le super bloc donne des informations sur les fichiers, telles que le nombre de nœuds d'index, le nombre de blocs, le premier bloc de données, la taille maximale de fichiers, etc.

Le super bloc est chargé en mémoire où on lui ajoute quelques informations supplémentaires, telles que le numéro du périphérique, un indicateur de mise à jour, etc.

10. La structure logique d'un disque Unix

Le système Unix permet de réunir plusieurs disques sous une arborescence unique.

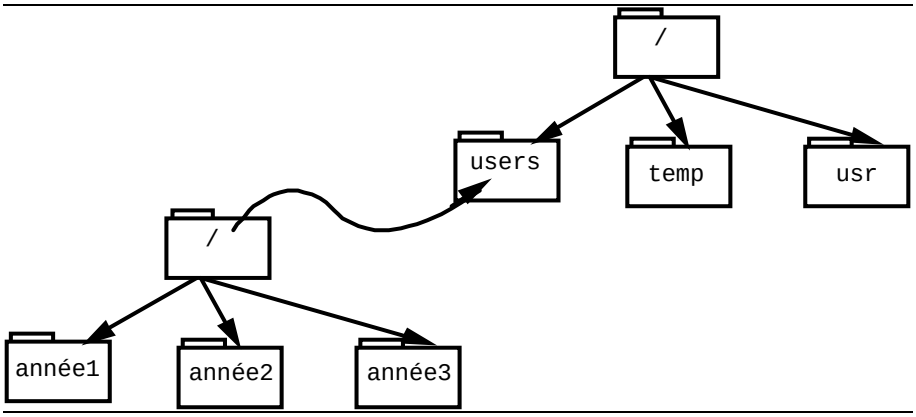


Figure 20

On réalise cet attachement par un « montage », grâce à la commande :

```
/etc/mount /dev/fd1 /users
```

où `/dev/fd1` désigne le disque à monter.

Le système « monté » devient :

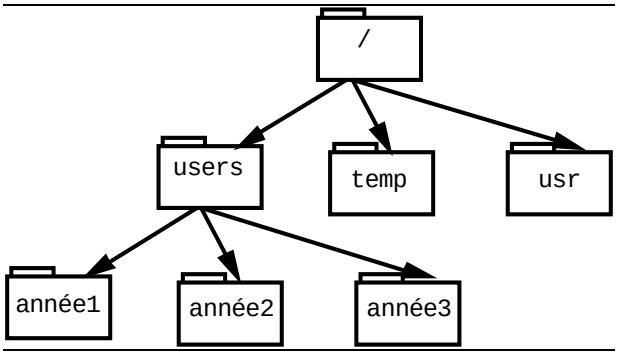


Figure 21

Une table des volumes montés garde la trace des différents périphériques[6]. Elle contient les éléments suivants :

N° du périph. monté	Ptr sur le tampon du super bloc	Ptr sur le i-nœud du système monté	Ptr sur le i-nœud du répert. de montage

Figure 22

Elle permet d'associer les disques à l'arborescence du système. Ainsi l'utilisateur peut parcourir les répertoires sans en supposer leur organisation physique.

11. Réparer le système de fichier

Les disques magnétiques sont des dispositifs délicats et il est fréquent de retrouver certains de leurs blocs, ou parfois même leurs références, dans un état incohérent.

Pour vérifier cette cohérence, on peut reconstruire la table des blocs occupés, en examinant tous les fichiers et en incrémentant le compte d'un bloc chaque fois

[6] Bach, p. 126, op. cit.

qu'il est référencé. Si le système est cohérent, la table des blocs occupés doit être complémentaire de celle des blocs libres.

Si un bloc n'apparaît ni dans la table des blocs libres, ni dans celle des blocs occupés, le bloc est dit manquant. On peut le réparer en le rajoutant à la liste des blocs libres.

Si un bloc appartient à deux ou plusieurs fichiers, le système est sans doute profondément incohérent. On peut effectuer une tentative de réparation en recopiant le bloc défectueux à des emplacements libres autant de fois qu'il est référencé puis en affectant chacun de ces nouveaux blocs à l'un des fichiers qui les référençaient.

On peut aussi vérifier la cohérence du point de vue des liens. Chaque numéro de nœud d'index doit apparaître autant de fois dans la structure arborescente qu'il possède de liens.

Sur la plupart des systèmes Unix, le programme `fsck` effectue cette tâche à chaque démarrage, si nécessaire.

Références

[1] SYSTÈMES D'EXPLOITATION & RÉSEAUX INFORMATIQUES, Notes de cours par Pierre Nugues.

[2] Cours Système d'exploitation, Oussama Mallouli, Mofdi Dhouib, Rachid Souissi, ISET Sfax.