Chapitre 5. Stockage des données.

Nous avons vu au chapitre précédent que l'on pouvait identifier quatre couches de fonctions d'entrée – sorties, à savoir, de la couche la plus haute à la plus basse : utilisation des données, organisation des données, stockage des données, et transfert des données. Nous avons déjà traité de ce qui concerne la couche de transfert. Nous allons maintenant nous intéresser à la couche suivante relative au stockage des données.

Le stockage des données est une fonctionnalité fondamentale du système d'exploitation, d'autant plus que ces données sont souvent partagées dans les environnements concurrents. Le mécanisme de base qui sous-tend le stockage des données est la gestion des fichiers. Mais de nos jours, le stockage des données est imperceptible à l'utilisateur, car encapsulée dans les applications qui manipulent ces données à travers des fichiers. Malheureusement, chaque application dispose de son propre format de fichiers, qui coexiste toutefois avec quelques formats standard : texte, document, jpeg, html, post-script, bitmap, etc.

Dans la suite de ce chapitre, nous entendrons par donnée, toute information produite par un programme et accessible par l'unité centrale. Ce terme couvre autant les données au sens classique du terme que les programmes sources et les programmes exécutables produits par les compilateurs.

Dans ce chapitre, nous présentons :

- les problèmes liés au stockage des données ;
- la mise en oeuvre du stockage des données ;
- le système de gestion des fichiers ;
- la sécurisation des données ;
- une étude de cas : le système de fichiers Unix.

Sans perdre la généralité, nous supposons dans ce chapitre, que le support des données est le disque dur, ou tout support de données similaire.

V.1. Problèmes liés au stockage des données.

Dans cette partie, après avoir présenté la notion de fichier, nous présentons le problème de la banalisation de l'accès aux données et la gestion de l'espace de stockage des données et leur temps d'accès.

V.1.1. Notion de fichier.

Le fichier est l'unité logique de stockage des données faisant abstraction de l'implantation physique des données sur le support. Cette notion de fichier est un héritage antique, systématisée par Barème, le fermier général de Louis XIV, pour répertorier les biens de son maître. En effet, les fiches étaient utilisées dans l'armée romaine pour enregistrer et garder les informations des recrues, à savoir des plaques de marbre gravées. Un fichier informatique (ou fichier dans cet ouvrage) est défini par un certain nombre d'attributs qui dépendent du système d'exploitation, et dont la plupart sont stockées dans le descripteur du fichier. Nous décrivons en détail ces attributs dans la section V.2.3.

Plus précisément, un fichier informatique est constitué de trois composantes, à savoir : le descripteur du fichier qui contient les attributs du fichier ; les données proprement dites qui sont souvent implantées en différents blocs ; et la structure de données servant à organiser l'espace occupé par les données et qui répertorie toutes les adresses des blocs où sont implantés les données.

V.1.2. Banalisation de l'accès aux données pour l'utilisateur.

Ce qui intéresse l'utilisateur, ce n'est pas la subtilité du stockage des données sur disque, mais d'une part, la manière dont ces données sont organisées ; et d'autre part, la technique pour y accéder. De plus, avec le développement fantastique des environnements d'exploitation sur les micro-ordinateurs, l'utilisation des interfaces graphiques tend à banaliser la notion de fichier, au point de confondre le fichier avec l'icône le représentant, la page de texte, le tableau, ou l'image qui est affichée là sur l'écran. Toutefois, même enfuies sous le graphisme le plus subtil de l'interface, ou banalisées par le «clic» de la souris, les opérations d'accès aux fichiers ne

demeurent pas moins présentes pour l'utilisateur. Le souci des concepteurs système est bien sûr de rendre ces opérations transparentes à l'utilisateur.

Que l'accès au fichier soit graphique ou par commande, on retrouve presque toujours les opérations de bases suivantes (table V.1) :

Opération	**Accès en mode graphique**	**Accès en mode commande**
Création	Elle est soit implicite avec un fichier vide créé d'avance à l'ouverture de l'application ; soit explicite en activant par exemple l'option ***nouveau*** dans le menu ***fichier*** des outils Microsoft.	Explicite avec une option de la commande d'ouverture
Ouverture	- spécification d'un nom - activation d'une icône - sélection dans un menu	Explicite avec une fonction d'ouverture indiquant le nom du fichier, par exemple la commande ou fonction ***open*** sous Unix.
Fermeture	- choix de l'option ***quitter*** ou ***fermer*** des outils Microsoft - activation d'un onglet de coin de fenêtre - ouverture d'un autre fichier pour certaines applications comme *Paint* dans Microsoft Windows	- Implicite car prise en charge par le système ; - Explicite avec une fonction de fermeture comme ***close*** sous Unix.
Modification	- Saisie directe des données à l'emplacement désiré	- Positionnement explicite à l'emplacement désiré (exemple: ***lseek*** sous Unix) et écriture avec une fonction (exemple : ***write*** sous Unix).
Suppression	- Capture et jet dans la poubelle - Sélection du fichier et application de la commande ou de l'option *supprimer*	- Commande particulière comme ***delete*** (Dos) ou ***remove*** (Unix), souvent le fichier n'est pas récupérable, sauf au prix de

	Dans les deux cas, le fichier subsiste dans la poubelle tant qu'elle n'est pas vidée.	nombreuses contorsions

- **Table** V.1 – Fonctionnalités de base d'un système de fichiers

Le système d'exploitation doit fournir l'ensemble des outils logiciels et graphiques permettant la conception d'applications manipulant des données. Il importe à l'utilisateur final que cet accès soit transparent aux autres applications, rapide, agréable, avec un nommage explicite des commandes ou une représentation iconique évocatrice. La normalisation de ces mécanismes devient une nécessité à laquelle il faut coller ; ainsi, les menus doivent se ressembler d'une application à une autre.

V.1.3. Gestion de l'espace et du temps.

Il n'y a rien de plus désagréable pour un utilisateur qu'une commande qui dure à se terminer. C'est que le temps de réponse est un facteur essentiel pour l'efficacité d'une application. Or plus la taille du fichier augmente ou plus sa structure se complexifie, plus le temps d'accès est long.

L'économie de place dans les disques était une fixation pour les concepteurs systèmes, il y a moins d'une décennie. Avec les technologies actuelles, ce problème semble résolu, car aujourd'hui, il devient de plus en plus difficile à un utilisateur de saturer son disque. Mais hélas, l'augmentation de l'espace disque s'est naturellement accompagnée de l'émergence de nouveaux types de données : l'image et le son, qui reposent à nouveau le problème d'économie de place sur le disque, et surtout sur les réseaux où ces données circulent.

Le système d'exploitation doit donc fournir des fonctions minimisant le temps de réponse, et permettant de réduire l'espace d'implantation des données sur le disque. C'est dans cette optique que des outils de compressions de données se développent, arrivant jusqu'à réduire de 80% l'espace de stockage d'un fichier avec la compression par ondelettes.

V.2. Mise en oeuvre du stockage des données.

Dans cette deuxième partie, nous présentons les mécanismes d'implantation physique des données. Pour cela nous montrons d'abord comment est organisé l'espace libre résultant du formatage du disque ; ensuite nous décrivons les techniques d'implantation physique des données ; et nous terminons avec les attributs d'un fichier stockés dans son descripteur et la description des opérations sur un fichier.

V.2.1. Organisation de l'espace libre.

Le temps d'accès au disque est très long : près de 10 000 fois le temps d'accès en mémoire. C'est pourquoi l'unité d'accès est en général un bloc pouvant contenir plusieurs centaines d'octets ; ce qui réduit le coût de transfert d'un octet. C'est aussi l'unité d'allocation de l'espace disque. L'espace tant libre qu'occupé, est donc découpé en blocs.

Lors du formatage du disque, les blocs libres non défectueux sont répertoriés dans une structure de données. Celle-ci est mise à jour lorsque des blocs pour le stockage des données sont sollicités, ou quand ceux-ci sont libérés et réintégrés à l'espace libre.

Deux techniques sont essentiellement utilisées pour gérer les blocs libres, à savoir :

- Vecteur de bits : chaque bloc est représenté dans ce vecteur par un bit qui vaut par exemple 0 quand le bloc est libre, 1 sinon. Cette technique est efficace quand les blocs sont de taille fixe et le disque de petite taille. Elle est presque abandonnée aujourd'hui.

- Listes chaînées : chaque bloc libre pointe sur le bloc suivant. Cette technique est efficace même quand les blocs sont de taille variable.

V.2.2. Implantation physique des données.

Dans la plupart des systèmes d'exploitation, l'espace de stockage des données est découpé en blocs de taille fixe. Au fur et à mesure du remplissage du fichier, des blocs libres sont prélevés dans la structure de données qui organise l'espace libre, et les données y sont stockées. Les blocs ainsi remplis intègrent l'espace occupé par le fichier. L'inscription physique des données dans ces blocs se fait avec les fonctions d'entrée-sortie de base comme celles qui ont été présentées au chapitre IV.

L'ajout de nouvelles données se fait à la suite du dernier bloc rempli. La modification des données est souvent une opération lourde qui nécessite de déplacer le contenu des blocs suivant le bloc modifié. Nous y revenons en examinant plus précisément le type d'organisation de l'espace des fichiers.

L'espace occupé par un fichier peut être géré selon trois techniques, à savoir :

- ***Espace linéaire*** : tous les octets du fichier occupent un espace contigu. L'ajout de données nouvelles est souvent difficile, et ne peut se faire qu'au prix d'une réorganisation de l'espace de stockage. Ainsi, pour modifier un fichier, la technique la plus simple consiste à le copier en mémoire et à le supprimer sur le disque, puis à lui réallouer un nouvel emplacement compatible avec sa nouvelle taille.

- ***Liste chaînée*** : chaque bloc du fichier pointe sur le bloc suivant. L'accès aux données devra se faire de manière séquentielle. La modification d'un bloc de données peut conduire à décaler le contenu de tous les blocks qui le suivent. Une solution pragmatique consiste à copier ces blocs en mémoire, à les libérer sur le disque, et à les réallouer. La solution est brutale mais facile à mettre en œuvre. L'ajout de données se fait en fin de liste.

- ***Tables des index*** : chaque fichier comporte une table qui contient les adresses des blocs de données qui le composent : la table des index, qui est implantée dans un ou plusieurs blocs. Cette table peut comporter plusieurs niveaux d'indirection

(tables de tables) pour permettre l'implantation de fichiers très volumineux. Comme pour la liste chaînée, la modification est coûteuse à mettre en œuvre ; l'utilisation de la technique de suppression – réallocation décrite plus haut est souvent l'alternative pragmatique. L'ajout se fait par remplissage du dernier bloc occupé. Cette technique de tables des index est mise en oeuvre dans le système de fichiers Unix et Linux, avec une adaptation judicieuse.

En général, un bloc particulier appelé bloc de contrôle ou descripteur du fichier, est dédié pour contenir les attributs ou informations de contrôle du fichier. Cette structure de données est décrite dans la section suivante.

V.2.3. Attribut et descripteur d'un fichier.

Un fichier est défini par un certain nombre d'attributs qui dépendent du système d'exploitation, et dont la plupart sont stockés dans une structure de données appelée ***descripteur du fichier***. De manière générale on trouve les attributs ci-dessous :

- **Le nom du fichier** : fixé par l'utilisateur, ou donné par défaut par l'application créatrice, il est souhaitable qu'il ait un lien avec son contenu. Dans des systèmes comme Unix ou Linux, un fichier peut avoir plusieurs noms, car l'attribut 'nom' n'est pas implanté avec le fichier, mais dans un fichier spécial appelé répertoire ; et chaque répertoire établit un lien entre un nom et un fichier ; nous y reviendrons.

- **L'adresse** : elle désigne l'emplacement du fichier dans le support d'enregistrement. En général l'adresse d'un fichier est en fait l'adresse de son descripteur.

- **Les liens de chaînage** : il s'agit de pointeurs vers la structure de données qui organise l'espace occupé par les données (voir § : V.2.2). Pour l'organisation linéaire, il s'agit de l'adresse de début de l'espace des données. Pour la liste chaînée, il s'agit des adresses du premier et du dernier bloc de données. Enfin, pour l'organisation indexée, il s'agit de l'adresse de la table des index.

- **Le numéro** : le nom du fichier étant souvent long et sujet à être modifié, de même que l'adresse est complexe en plus du fait que le fichier peut être déplacé, dans certains systèmes d'exploitation comme Unix, le fichier est identifié par un numéro unique.

- **Le type** : selon les systèmes d'exploitation, il indique le type d'information que le fichier contient, le mode d'organisation de ces données ou le nom de l'application qui a créé ce fichier. Comme type d'information, on peut par exemple avoir des fichiers sources avec l'extension du compilateur à utiliser, les fichiers objets et les fichiers exécutables. Comme modes d'organisation, on peut citer les fichiers textes qui sont des séquences de caractères, des fichiers images qui sont des matrices, et des tableaux qui sont des structures à deux dimensions. Enfin, dans les environnements graphiques, souvent chaque fichier est représenté par l'icône de l'application qui l'a créé.

- **La taille** : elle désigne le nombre d'unités de données que contient le fichier. De manière générale, l'unité de base est l'octet. La taille est mise à jour au fil de l'évolution du fichier et est appelée ***taille de données***. La taille ***réelle d'un fichier*** est constituée de la taille des données à laquelle on ajoute la taille du descripteur et la taille de l'espace occupé par les index ; cette taille n'est jamais nulle.

- **Les droits** : ils permettent de spécifier les protections du fichier et le type d'opérations (ou modes d'accès) qu'on peut y effectuer, ainsi que le type d'utilisateurs qui peuvent y accéder.

- **Les informations d'identification** : il s'agit du nom ou du numéro de créateur du fichier, de son groupe, ou de la dernier utilisateur, etc.

- **Les dates** : se sont les dates de création, de dernière modification, de dernier accès, etc.

Le descripteur de fichier qui contient l'essentiel de ses attributs est souvent implanté dans :

- une table de descripteurs de fichiers : il s'agit d'un espace défini dans le support des données, destiné à contenir exclusivement les descripteurs de fichiers. Comme ces descripteurs ont la même taille, l'espace qu'ils occupent peut donc être géré comme une table. C'est la solution retenue dans le système de fichiers Unix / Linux ;

- un bloc dédié : chaque descripteur de fichier est logé dans un bloc qui lui est propre. Dans ce cas, un bloc particulier doit contenir la liste de tous les descripteurs constituant le système de fichiers ;

- dans une liste chaînée : chaque descripteur de fichier est lié au fichier suivant par un lien de chaînage.

V.2.4. Opérations sur un fichier.

Les principales opérations sur les fichiers pour l'accès aux données sont :

- ***Création de fichier*** : elle revient à allouer le bloc de contrôle du fichier et à inscrire son adresse dans la structure de données idoine (table ou liste des descripteurs de fichiers). Les différents champs du bloc de contrôle doivent aussi être initialisés avec les valeurs idoines des attributs.

- ***Ouverture du fichier*** : elle revient soit à copier le bloc de contrôle du fichier en mémoire (en général dans la table des fichiers liée au programme demandeur ou au système d'exploitation) ; soit à inscrire son adresse dans le table des fichiers du programme demandeur. Le bloc de contrôle du fichier peut être mis à jour pour inscrire le mode d'accès et l'identité de l'utilisateur ; il est aussi utilisé pour le contrôle des droits d'accès.

- ***Lecture*** : à chaque accès au disque, un bloc entier est lu et transféré en mémoire, ensuite l'accès aux octets se fait dans la mémoire. Le prochain d'accès au disque sera effectué quand un octet non présent en mémoire sera sollicité ; à ce moment, tout le bloc contenant cet octet est lu et transféré en mémoire. L'accès à chaque bloc nécessite le parcours de la table des index du fichier. La même fonction de lecture doit garantir l'accès des octets en mémoire et la récupération des blocs sur disque. En général une variable système associée au fichier indique la position courante ; c'est-à-dire le numéro du dernier octet lu.

- ***Ecriture*** : elle se fait en général à partir de la fin du fichier. Le dernier bloc est rempli ; ensuite, un nouveau bloc est récupéré dans l'espace libre et attaché à la

table des index, avant d'être rempli. La taille du fichier est mise à jour. L'insertion de données en milieu de fichier nécessite souvent de décaler des données qui suivent.

- ***Suppression*** : les blocs de données et des index sont restitués dans l'espace libre, ainsi que le descripteur du fichier.

V.3. Système de gestion de fichiers.

Il est souvent nécessaire de structurer l'ensemble des fichiers de façon à faciliter l'accès aux données. La technique utilisée pour cela est la mise en place d'un système de gestion de fichiers.

Un système gestion de fichiers (SGF) ou système de fichiers par abus de langage, est une structure de données qui répertorie l'ensemble des fichiers afin d'en faciliter l'utilisation, associées à un ensemble de fonctionnalités de gestion de cette structure de données et des fichiers associés.

Le système de gestion de fichier structure donc tant l'espace libre que l'espace occupé par les données, et il offre les fonctions d'accès à ces données organisées sous forme de fichiers.

V.3.1. Organisation d'un système de fichiers.

Le fichier, comme nous pouvons l'appréhender dans les sections précédentes, est parfaitement défini si l'on connaît son adresse de début ou l'adresse de son descripteur. En pratique, la gestion d'identificateurs numériques (ex : adresses de fichiers) pour désigner une entité n'est pas aisée. C'est pourquoi, on attribue un nom à chaque fichier, sachant que la correspondance entre le nom et l'adresse physique du fichier sur le disque est faite à l'aide d'une table de correspondance appelée ***répertoire***. Cette table est souvent organisée de manière hiérarchique (tables de tables) ; c'est le principe de base de la mise en œuvre des répertoires.

Plus précisément, un répertoire est une table de correspondance mettant en relation chaque nom de fichier à chaque adresse de fichier et à une icône. Les fichiers sont appelés éléments ou membres du répertoire, sachant qu'un répertoire peut être lui-même élément d'un autre

répertoire. Un répertoire est en fait un fichier contenant des références de fichiers et répertoires. L'organisation des fichiers en répertoire conduit souvent à une structure hiérarchique. Un exemple complet est donné dans la section V.4 avec le système de fichiers Unix.

V.3.2. Opérations sur un système de fichiers.

Le système d'exploitation met à la disposition de chaque utilisateur, un ensemble de fonctionnalité tant pour la gestion des fichiers que pour la manipulation des données.

En général le système tient une table des fichiers ouverts, qui contient leurs descripteurs. De même, à chaque utilisateur est associée une table des fichiers ouverts.

V.3.3. Accessibilité et Partage des données.

Dans la plupart des systèmes d'exploitation, chaque programme dispose d'une ***table des fichiers*** qu'il manipule. Celle-ci contient une entrée par fichier, constituée des informations propre à ce programme utilisateur pour ce fichier. Comme le même fichier peut être ouvert simultanément par plusieurs programmes, certains systèmes d'exploitation comme Linux ou Unix, disposent d'une table système pour tous les fichiers ouverts. Celle-ci contient une entrée par fichier ouvert, comprenant les informations propres au fichier. L'essentiel de ces informations émanent du descripteur des fichiers. En effet, quand un programme ouvre un fichier, si c'est la première fois que celui-ci est ouvert, alors une entrée est créée dans la table système, et on y recopie les informations contenues dans le descripteur du fichier. Dans tous les cas, une entrée est créée dans la table des fichiers du processus pour ce fichier. Sans être exhaustif, la structure de chaque entrée des deux tables est donnée ci-dessous (tables V.2 et V.3).

Type d'information	Désignation
Type de fichier	Indique le format de représentation des données ou l'application qui l'a créé.
Emplacement physique	Indique l'adresse physique du fichier sur disque, c'est-à-dire l'adresse de son descripteur
Compteur d'utilisateurs	Indique le nombre de programmes qui ont ouvert ce fichier
Taille du fichier	Nombre de caractères ou d'unités de données du fichier
Flag d'écriture	Indique si le fichier a été ouvert en écriture
Droits d'accès	Indique les droits d'accès que chaque type de programme utilisateur peut avoir sur ce fichier. Ces droits fixent en même temps le type d'opérations que l'utilisateur peut effectuer sur ce fichier

- **Table** V.2 – Structure d'un descripteur de fichier dans la table système des fichiers

Type d'information	Désignation
Point de lecture	Position courante de lecture dans le fichier
Point d'écriture	Position courante d'écriture dans le fichier
Droits d'accès individuels	Indiquent le type d'opérations que l'utilisateur peut effectuer sur ce fichier. Ces droits sont calculés en utilisant les droits du fichier donnés par la table système des fichiers et le type de l'utilisateur.

- **Table** V.3 – Structure d'un descripteur de fichier dans la table des fichiers d'un programme.

Toutefois, le système de fichier ne comporte en général pas de mécanisme d'exclusion mutuelle. Aussi revient-il à l'utilisateurs de veiller résoudre lui-même les conflits d'accès aux fichiers. Certaines applications signalent à l'utilisateur les ouvertures multiples d'un même fichier. D'autres arrivent à refuser la double écriture : le premier programme qui écrit dans un fichier garde l'exclusivité de l'utiliser en écriture.

V.4. Sécurité et protection des données.

Garantir la sécurité des informations est une préoccupation permanente des administrateurs systèmes. Cette préoccupation est d'autant plus grande que les données sont dites sensibles, c'est à dire que leur diffusion ou leur perte peut être préjudiciable à leur propriétaire.

V.4.1. Cause de la perte et de la fuite d'information.

Les causes de la perte d'informations sont multiples. Les plus importantes sont :

- ***Les défaillances matérielles*** : réchauffement du matériel, courts circuits, rayures dues à des poussières, etc.
- ***Les défaillances logicielles*** : erreurs de programmation ou de nommage de fichiers, exécution intempestive, etc.
- ***Les virus*** : c'est aujourd'hui un très grand fléau, avec les milliers de types de virus répertoriés sans compter que des dizaines apparaissent chaque mois. Les virus peuvent ravager un disque dur en détruisant les tables des secteurs, ou en modifiant les données. Leur capacité de nuisance est presque sans limite et sans frontière avec l'évolution de l'interconnexion des réseaux.
- ***Les maladresses humaines*** : l'emploi des commandes d'effacement de fichier par inadvertance, le formatage d'un support contenant des données, la réponse à certains anti-virus qui détruisent les fichiers infectés, le déplacement accidentel de fichiers, etc.
- ***La loi de Dieu*** : les catastrophes naturelles, les intempéries, les bestioles, la poussière, les orages magnétiques, l'humidité, les incendies, etc.
- ***La loi du mal*** : Beaucoup d'érudits de l'informatique ont versé dans la malveillance, soit par méchanceté, soit par vengeance, soit par un plaisir cynique,

pour endommager des systèmes d'information. Ils tentent souvent de pénétrer frauduleusement dans un système pour endommager données et programmes.

Souvent pour certaines de ces causes, il est possible de souscrire une police d'assurance. Pour d'autre quelques précautions élémentaires suffisent. Enfin pour certaines, il ne reste que la résignation.

V.4.2. Quelques techniques pour sécuriser les données.

Il existe une variété de techniques pour mettre en œuvre la sécurisation des données. Parmi les plus courantes, nous pouvons citer :

- ***Les sauvegardes*** : Elles permettent de répondre au moins partiellement à la destruction de données. On ne le dira jamais assez, un informaticien prudent doit régulièrement effectuer la sauvegarde de ses données. Plus elles sont sensibles, plus elles varient dans le temps, plus souvent il faut effectuer des sauvegardes globales (tous les fichiers), partielles (uniquement certains fichiers) ou incrémentales (uniquement les fichiers modifiés depuis la dernière sauvegarde). Certes, le volume de plus en plus élevé des disques et la diversité des applications et des fichiers compliquent cette tache. Mais la sauvegarde est un choix courageux à faire et un engagement à prendre. Aujourd'hui, des bandes, des cassettes et des disques à lecture rapide existent. Le graveur de CD-ROM, , le lecteur ZIP ou la clé USB sont d'autres alternatives.

- ***La machine miroir*** : il s'agit d'un ordinateur qui héberge une copie du système d'information d'un autre ordinateur. La copie de l'ordinateur vers son miroir se fait périodiquement et de manière automatique. La mise en œuvre d'un miroir nécessite de synchroniser les deux systèmes de sorte que la copie se fasse en minimisant les conflits d'accès (i.e. lecture et écriture simultanées). La technique de sauvegarde incrémentale minimisera le trafic. Avec les outils réseaux (ex : ***ftp*** (*File Transfer Protocol*) ou ***nfs*** (*Network File System*)), la mise en œuvre d'un miroir ne pose aucune difficulté technique majeure, sauf dans la détection des conflits d'accès. Afin de rendre les pannes transparentes à l'utilisateur, le basculement de l'activité

de la machine primaire à son miroir peut être rendue automatique. Au lieu de dédier un ordinateur pour être miroir d'un autre (ce qui est fort coûteux), on peut se contenter de dédier un disque sur une machine de second plan, donc le rôle est d'héberger une copie du disque de la machine primaire : c'est un disque miroir. En cas de panne, le contenu du disque miroir peut être basculé dans le disque primaire ; et si ce dernier est altéré, le miroir peut être monté à sa place.

- ***Les mots de passes*** : dans les gros systèmes, le mot de passe est une véritable barrière contre les intrus, à condition que des mots fantaisistes et des lieux communs ne soient pas utilisés. Certains logiciels permettent de verrouiller les fichiers avec un mot de passe. Aujourd'hui, il existe des techniques éprouvées de cryptage de mots de passe, telle que si un intrus accède à un fichier de mot de passe, il ne puisse rien décrypter. Une de ces techniques consiste à ajouter un nombre aléatoire au mot de passe crypté. Un mot de passe doit être changé régulièrement, et cela, d'autant que les données sont sensibles. Dans les petits systèmes d'exploitation sur PC comme Windows (XP, Vista), le mot de passe peut parfois être un simple écran de fumée. Dans ce cas, nous recommandons de procéder au cryptage des données pour mieux les protéger des intrusions.

- ***Les cartes d'accès et l'identification physiologique***. Depuis l'aube des temps informatiques, l'accès aux salles machines a toujours été restreint à un petit nombre d'usagers. Le contrôle d'accès est effectué de diverses manières : badges contrôlés par un vigile, carte magnétique ou optique, lecteurs d'empreintes digitales, analyseur vocal, etc. Ces détecteurs sont souvent reliés à un central qui est capable de tenir une trace des différents passages.

- ***Les cyberflics*** : Avec l'avènement de l'Internet et la recrudescence du banditisme cybernétique, les services centraux de police recrutent de plus en plus de spécialistes de l'informatique pour traquer les intrus (pirates) dans certains systèmes étatiques. Les sites de la NASA sont par exemple extrêmement sollicités, et le FBI a une équipe d'érudits pour y faire face. Plus modestement dans des centres sensibles, il peut s'avérer nécessaire d'avoir un limier capable d'écouter le système pour détecter les attaques ou les intrus. Pour ce faire, le système Unix enregistre une trace des connexions consécutives et présente en permanence la liste

d'utilisateurs connectés. Il suffit donc de développent un utilitaire d'analyse périodique de trace pour mettre en œuvre la détection (*tracking*) des intrus.

V.4.3. Les antivirus.

Nous pensons qu'il y a lieu de donner à la lutte contre les virus un caractère spécial, car de loin, les virus sont aujourd'hui la cause première de la perte de données, en particulier dans les environnements équipés de systèmes Microsoft. Le milieu UNIX est bien moins agressé. Les virus attaquent pèle mêle, les programmes exécutables (sur disque ou en mémoire), les fichiers résultant de certains logiciels (ex : *word*, *excel*) et dans ces fichiers, certaines composantes (ex : macros de *word*), etc. Les attaquent peuvent être insidieuses (sournoises et graduelles), ou virulentes (agressives et massives).

Il y a une diversité de virus, et une petite famille d'antivirus. La plupart de ces antivirus se greffent dans le système immédiatement après le programme de boot (démarrage du chargement du système) de façon à anticiper l'action des virus en contrôlant tous les fichiers auxquels le système pourra accéder par la suite. Un antivirus peut être actif ou passif. Quand il est actif, il analyse chaque fichier dès lors qu'il est sollicité par le système. Un antivirus est passif quand l'analyse est déclenchée explicitement par l'utilisateur.

Les environnements utilisant couramment des lecteurs disquettes et les clés USB nécessitent des antivirus actifs, car après Internet, les disques amovibles (disquettes, clés USB) constituent le deuxième vecteur de diffusion des virus, en particulier dans les milieux étudiants.

Nous préconisons au moins un contrôle complet du disque une fois par jour si l'ordinateur est relié à un réseau, ou s'il reçoit des disques amovibles plusieurs fois par jour. L'option d'analyse des fichiers exécutables doit être activée si elle existe, de même que l'analyse systématique de tout disque amovible.

Un antivirus devient caduque après trois mois, période d'apparition de nouveaux virus, toujours plus sophistiqués. Les antivirus du commerce comme NORTON (Symetec), QUICK HEAL, ou ViruScan (MCAFEE), proposent souvent des mises à jour périodiques qu'il faut se procurer par téléchargement. Il est fortement recommandé de souscrire à ses mises à jour, pour que le système

ne soit abusivement infecté. Certains antivirus sont gratuits, mais en général ils ne sont pas régulièrement mis à jour, et ne couvrent souvent qu'une petite partie des virus existants.

V.4.4. Développement de contrôles et protections.

Tout comme les techniques de sauvegarde automatiques, les techniques de contrôles automatiques peuvent être développées par n'importe quel informaticien à condition d'y consacrer un peu de temps, et d'avoir de la patience. Sous Unix par exemple, les ficelles suivantes peuvent être exploitées :

- Pour éviter l'effacement accidentel d'un fichier : créer un lien sur ce fichier avec la commande *ln* (*link*). Un fichier UNIX n'est effacé que quand le dernier lien est supprimé.

- Restreindre l'accès à un fichier : utiliser les différentes options de la commande *chmod* (*change mode*), pour restreindre l'accès aux fichiers pour soi-même, pour son groupe ou les autres utilisateurs.

- Connaître toutes les personnes connectées : utiliser la commande *who*. Exécutée périodiquement en toile de fond (*background*), c'est un moyen bon marché de connaître les utilisateurs du système. En redirigeant son résultat vers un fichier de trace, on peut développer un utilitaire qui analyse cette trace pour détecter les connexions frauduleuses.

- Connaître les caractéristiques d'un utilisateur : utiliser la commande *finger*. Une fois un intrus déniché, il est possible de mieux cerner son identité, et même de lui envoyer un message (commande *write*).

Plusieurs opérations de contrôle peuvent se faire en écrivant des scripts ; mais le langage C offre une plus grande souplesse. Entendu que les données utiles à ces contrôles sont produites par les commandes UNIX, le résultat de ces commandes doit donc être dirigé non sur l'écran mais dans un fichier. Ce qui se fait simplement de la façon suivante :

cmd > ftempo

où *ftempo* est le nom d'un fichier temporaire qui va contenir le résultat de la commande *cmd*. Il suffira ensuite d'ouvrir ce fichier dans le programme C pour disposer du résultat de la commande *cmd*. Mieux encore, la commande *cmd* peut être exécutée directement à l'intérieur du programme C. Pour cela, il suffit d'incorporer l'appel à la fonction ***system*** dans le programme C :

system*(''cmd > ftempo'') ;*

A partir de ces techniques fort simples, il est possible de développer de puissants outils de contrôles. Les concepteurs d'outils système ne font pas autrement.

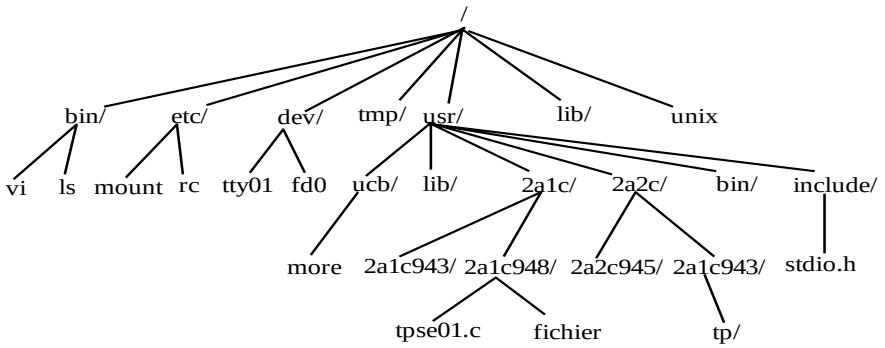
Le développeur système peut aller au-delà, en accédant à certaines structures de données systèmes (descripteurs de fichiers ou de liens de communication, structures de données des processus, etc.). Cela nécessite du temps et de la détermination. Mais c'est à la portée d'un grand nombre d'informaticiens. La section suivante en dit plus sur le système de fichier UNIX.

V.5. Système de fichiers UNIX.

La gestion des fichiers est l'une des fonctions du noyau d'UNIX. Toutes les entrées-sorties sont en fait gérées comme des fichiers ; ce qui en donne une approche simple et unique. Le système des fichiers est chargé à la fois de la modélisation des données, de leur stockage et d'en permettre la manipulation à travers un ensemble de fonction et de commandes.

V.5.1 organisation arborescente des fichiers.

Sous UNIX, les fichiers sont organisés sous une forme arborescente. La racine notée / et les nœuds internes sont appelés répertoires (*directories*) ; et les feuilles sont des fichiers ou des répertoires vides. Pour spécifier le chemin d'accès (*path name*) à un fichier, on indique tous les répertoires de la racine au fichier, en séparant les noms par /. La figure IV.1 représente quelques fichiers et répertoires de base du système UNIX. Les répertoires sont notés en gras. Le chemin d'accès au fichier *more* est : **/usr/ucb/more**.



- **Figure IV.1** - Arborescence de base partielle.

Quelques-uns des principaux répertoires qu'il est bon de connaître :

- /usr/bin et /usr/ucb : Contiennent des commandes.
- /dev : Les périphériques ou les fichiers spéciaux qui correspondent aux dispositifs d'entrée-sortie.
- /etc : Commandes et fichiers de maintenance et administration du système.
- /lib et /usr/lib : Les bibliothèques.
- /tmp et /usr/tmp : Les fichiers temporaires.
- /usr/adm : Fichiers de comptabilité.
- /usr/include : Les fichiers en-tête pour les programmes.
- /usr/src : Les sources du système.

A chaque instant l'utilisateur dispose de trois répertoires prédéfinis :

- **Le répertoire de référence** : *home directory*, localise le nœud sous lequel on se trouve automatiquement dès qu'on se connecte sous UNIX. On revient à ce nœud en tapant simplement la commande *cd*

- **Le répertoire courant** : est le nœud de l'arborescence sous lequel on se trouve.

- **Le répertoire père** : est le père du répertoire courant.

On notera que les répertoires . et .. sont présents dans tous les répertoires ; en somme, un répertoire n'est jamais vide.

On se déplace dans l'arbre des répertoires grâce à la commande **cd** (*change directory*) :

cd *dirpathname*

où *dirpathename* est le chemin d'accès du répertoire auquel on veut se déplacer. Il existe deux façons de spécifier ce chemin.

Le chemin d'accès **absolu** d'un répertoire est la séquence des noms des nœuds suivis pour arriver à ce répertoire à partir de la racine /. Dans l'arbre de la figure IV.1, la commande pour accéder au répertoire *2a2c943* par le chemin absolu est :

cd */usr/2a2c/2a2c943*

Le chemin d'accès **relatif** à un répertoire est constitué de la liste des noms de répertoires qu'on rencontre sur le chemin de l'arbre en partant du répertoire courant. Les noms . et .. peuvent être introduits dans cette liste. Dans l'arbre de la figure IV.1, si le répertoire courant est *2a2c943*, et qu'on désire se rendre au répertoire *2a1c*, on utilisera la commande suivante :

cd *../../ 2a1c*

V.5.2. Types des fichiers UNIX.

Tout fichier UNIX est une suite d'octets. Une structure ne peut être plaquée à un fichier qu'à l'intérieur d'un programme qui utilise ce fichier. Il y a trois types de fichiers :

1 - Les fichiers ordinaires : un fichier ordinaire contient des caractères. Le caractère NUL : \0 désigne en général la fin de fichier; le programmeur veillera donc à ne pas l'employer autrement. Dans certaines versions d'UNIX, il est recommandé de convertir les données numériques en chaînes de caractères avant de les enregistrer. UNIX offre des fonctions qui permettent d'exploiter des fichiers de manière structurée à l'intérieur de programmes.

2 - Les répertoires : contiennent des fichiers ou d'autres répertoires. Un répertoire peut être lu comme un fichier ordinaire.

3 - Les fichiers spéciaux : Ils sont réservés aux périphériques, et sont souvent regroupés dans le répertoire */dev*, mais peuvent être déplacés dans n'importe quel

répertoire. Ces fichiers servent également à lier les tubes (*pipes*) nommés au système de fichiers. Les fichiers spéciaux se divisent en deux catégories :

- Dispositifs caractères (*character devices*) : L'échange des données se fait par caractères (ex : terminaux : /dev/tty00, dans la figure IV.1).
- Dispositifs blocs (*block devices*) : l'échange des données se fait par blocs (ex : disques durs et disquettes). Un bloc est généralement de 512 ou 1024 caractères.

Chaque fichier comporte des informations de contrôle : le type du fichier (usuel, répertoire, spécial) ; le nom du fichier ; le nom et le groupe du propriétaire ; le mot de contrôle d'accès de 9 champs ; les informations de localisation ; la taille en nombre de caractères ; les dates de création, de dernière modification, de dernier accès, un numéro d'accès à la table des descripteurs associée au périphérique (utiliser la commande Unix *ls* avec l'option *-l*).

Notons enfin que chaque fichier spécial est distingué par deux numéros. Le numéro majeur qui distingue le type de l'unité d'entrée/sortie et le numéro mineur qui identifie cette unité parmi les unités du même type. Ces numéros sont attribués aux fichiers spéciaux à l'installation.

V.5.3. Droits d'accès et protection de fichiers UNIX.

A la création d'un compte pour un utilisateur, l'administrateur le met dans un groupe. L'ensemble des utilisateurs est décrit dans le fichier ***/etc/passwd*** ; dans lequel, chaque utilisateur est représenté par une ligne comportant dans l'ordre les champs suivants (séparés par des virgules) :

- Nom
- Mot de passe crypté (optionnel)
- Numéro d'utilisateur (*uid*)
- Numéro de groupe (*gid*)
- Commentaires
- Répertoire d'accueil *(home directory*)
- Programme à lancer, en général le shell (optionnel)

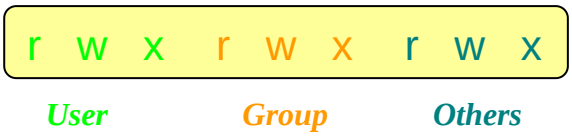
Les mots de passe peuvent être regroupés dans un fichier caché. Dans ce cas le champ mot de passe du fichier ***/etc/passwd*** contient un caractère spécial comme *.

Un groupe rassemble des utilisateurs pouvant partager des fichiers. L'ensemble des groupes est décrit dans le fichier ***/ect/group***. Chaque groupe est décrit par une ligne comportant les champs :

- Nom du groupe
- Non utilisé
- Numéro de groupe
- Liste des utilisateurs.

Un utilisateur ne peut appartenir qu'à un seul groupe à la fois. Il peut changer de groupe grâce à la commande ***newgrp***.

Un fichier a trois types d'utilisateurs : le propriétaire (*user*, c'est souvent le créateur du fichier), les membres du groupe du propriétaire (*group*), et les autres utilisateurs (*others*). Un fichier comporte trois modes d'accès : lecture (*read*), écriture (*write*), et exécution (*execute*). On appelle droits d'accès à un fichier, les types d'accès autorisés aux utilisateurs. Le propriétaire attribue ces droits d'accès lors de la création du fichier, et peut les modifier par la suite avec les commandes *chmod*, *chgroup* et *chown*. Chaque fichier est protégé par 9 bits définissant les droits d'accès, qui sont répartis suivant la figure IV.2.



- **Figure IV.2** - Répartition des droits d'accès.

La commande ***ls -l*** permet de connaître les droits d'accès des fichiers d'un répertoire.

V.5.4. Structure du système de fichiers UNIX.

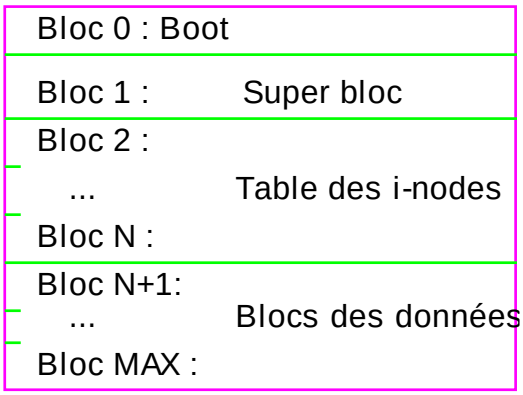
Dans l'optique de la programmation modulaire qui caractérise UNIX, il est possible de créer plusieurs systèmes de fichiers sur le même support physique. Ce choix est essentiellement lié aux besoins de :

1. Sécurité. Si un système de fichier est endommagé, les autres sont préservés.
2. Fonctionnalité. Meilleure gestion des ressources, et rapidité du système de fichier.
3. Performances. Limitation de la fragmentation d'un même espace de mémorisation.

On crée un système de fichiers sur un disque ou une disquette grâce à la commande ***mkfs*** :

/etc/mkfs ***file device***

Cette commande crée sur le support (*device*) représenté par le fichier spécial *file*, les structures de blocs nécessaires au système de fichier (format donné par la figure IV.3).



- Figure IV.3 - Structure d'un système de fichiers.

Le 1er bloc contient le programme de chargement du système (*boot block*). Ce bloc n'est utilisé que pour le système de fichier primaire. Les sections suivantes décrivent les autres blocs.

V.5.4.1. Le super bloc d'un système de fichiers UNIX.

Le **super bloc** est le descripteur du système de fichiers UNIX. Il contient des informations générales du système de fichiers, à savoir :

1. La taille de la table des index (voir plus bas).
2. Le nombre total de bloc du système de fichier
3. Le descriptif des blocs libres
4. Les sémaphores de manipulation du super bloc.
5. La date de dernière modification du super bloc

6. La variable persistante *s_clean*, chargée avec la valeur *S_CLEAN*, qui permet de savoir si le système a été proprement arrêté (utilisation de la commande *shutdown*). Rappelons que les constantes sont définies dans le fichier */usr/include/sys/param.h*.

Le super bloc est mis à jour par le système d'exploitation chaque fois qu'un fichier est créé ou supprimé, ou sa taille modifiée. Sa structure est décrite dans le fichier */usr/include/sys/filsys.h*. Pour augmenter la performance, le super bloc est copié en mémoire centrale ; copie sur laquelle sont effectués les changements. Pour assurer la cohérence du système de fichiers, une mise à jour sur disque est effectuée.

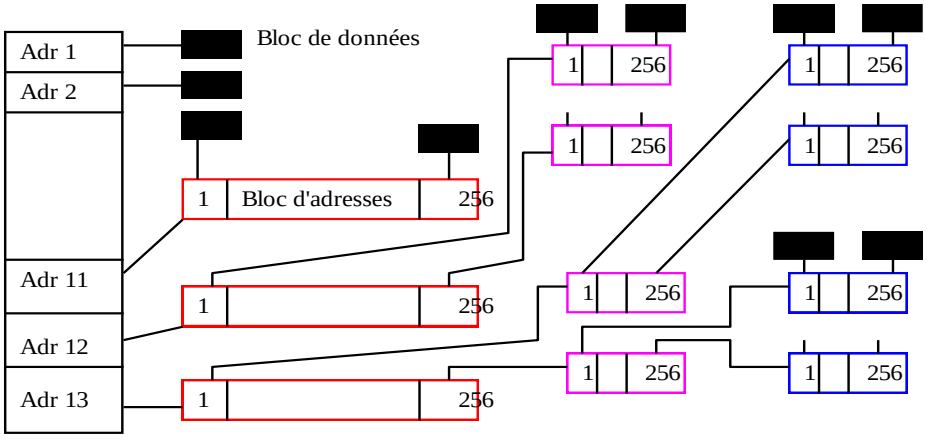
V.5.4.2. Table des index d'un fichier UNIX.

La table des index (*i-nodes*) du système des fichiers occupe les blocs de 2 à N ; où N est un paramètre du système d'exploitation. Chaque bloc contient 16 i-nodes. Un i-node identifie totalement un fichier et contient toutes les informations sur ce dernier. Un i-node comporte 64 octets repartis en 10 champs :

- Type du fichier.
- Nombre de liens.
- Identificateur du propriétaire (*uid*).
- Identificateur du groupe du propriétaire (*gid*).
- Taille du fichier (nombre d'octets).
- Adresses physiques des blocs du fichier (39 octets).
- Non utilisé.
- Date du dernier accès.
- Date de la dernière modification.
- Date de création.

L'implantation physique du fichier est décrite par les 39 octets repartis en 13 adresses. Les 10 premières adresses pointent directement sur des blocs de 1 KO. Si le fichier comporte plus de 10 blocs, la 11ème adresse pointe sur bloc de 256 adresses de blocs ; chaque bloc ayant 1 KO. Si le fichier dépasse 256 KO, la 12ème adresse point sur un bloc de 256 adresses, chacune pointant sur 256 adresses de blocs de données. Enfin, la 13ème adresse permet d'étendre considérablement cette indexation sur 3 niveaux. Chaque adresse étant codée sur 3 octets, on

peut adresser 224 blocs de 1 KO. Cette organisation est décrite par la figure IV.4. Il faut enfin noter qu'un support physique peut contenir plusieurs systèmes de fichiers (un par partition).



- **Figure IV.4** - Implantation d'un fichier.

Le premier i-node (i-node 0) dans la table n'est pas utilisé. Le deuxième (i-node 1) pointe sur la table des i-nodes elle-même. Le troisième (i-node 2) pointe sur la base du système de fichiers.

V.5.4.3. Les répertoires : accès aux fichiers UNIX.

IV.5.4.3.1. Les répertoires UNIX.

On notera que le nom n'apparaît pas dans les informations d'un fichier présentées plus haut. En fait le nom d'un fichier est associé à un répertoire. Un répertoire sert à mettre en correspondance un nom de fichier avec un i-node. C'est un fichier ordinaire, comportant un i-node, mais que seul le système peut modifier. Un fichier de type répertoire comporte 16 octets pour chaque fichier membre de ce répertoire : 2 octets pou l'i-node, et 14 pour le nom du fichier. On notera que le nom d'un fichier comportera au plus 14 caractères.

Dès qu'un répertoire est crée, le système lui associe automatiquement deux fichiers qui sont en fait des répertoires. Le premier nommé (..) est le répertoire père. Le deuxième nommé (.) est le répertoire lui-même. Supposons qu'on ait deux répertoires de noms *carte1* et *carte*. Le premier contient les fichiers *pays* et *image*. Le deuxième contient les fichiers *ville* et *dessin*. Ces deux répertoires peuvent être représentés de la façon suivante (fig. IV.5) :

I-NODE	NOM
8	..
9	.
33	pays
78	image

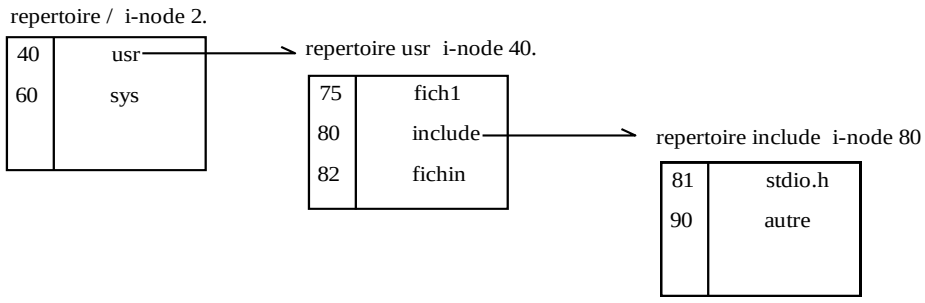
I-NODE	NOM
8	..
20	.
45	ville
78	dessin

- **Figure IV.5** - Répertoire carte1 Répertoire carte2

On remarque aussi que les fichiers *image* et *dessin* ont le même i-node. Ce sont donc deux noms différents pour le même fichier. Ce sont deux **liens** pour ce fichier. De même, les deux répertoires ont le même père. Un **lien** pour un fichier est un autre nom de ce fichier ayant le même i-node. Notons enfin que la structure du système de fichiers telle qu'on vient de la définir correspond à une structure d'arborescence ayant comme racine la racine du système de fichiers. Toutefois, l'utilisation de lien transforme le système des fichiers en un graphe acyclique.

IV.5.4.3.2. Accès à un fichier UNIX.

L'accès à un fichier démarre par son i-node, qui, rappelons le, comporte 64 octets identifiant totalement un fichier. Nous allons illustrer l'accès à un fichier par l'exemple de la figure IV.6. Soit à accéder au fichier */usr/include/stdio.h* (fig. IV.1). Le système connaît l'i-node de la racine, qui est toujours 2. Accédant à cet i-node et à ses blocs de données, il cherche le nom *usr* et récupère son i-node 40. En suite, le système accède au i-node 40 et aux blocs de données associés pour chercher *include* et son i-node 80. Accédant de même à l'i-node 80, le système retrouve le fichier *stdio.h* et son i-node 81. L'i-node 81 donne l'implantation physique du fichier *stdio.h*.



- **Figure IV.6** - Représentation des répertoires.

Cette opération n'est effectuée qu'à l'ouverture du fichier. L'i-node est alors recopié en mémoire dans une table du système d'exploitation, ainsi qu'une copie du super-bloc qui est mis à jour. La commande *sync* permet de mettre à jour les i-nodes et les blocs de données des fichiers ouverts. Il est recommandé de l'exécuter de temps en temps, surtout en quittant le système, car celui-ci effectue les mises à jour périodiquement.

On voit donc que l'ouverture d'un fichier occasionne plusieurs accès au disque. Pour cette raison, il est recommandé de limiter un répertoire à 64 fichiers pour qu'il tienne dans un seul bloc de données.

V.5.5 Montage et démontage d'un système de fichiers

Dans une machine, il peut y avoir plusieurs disques. Sur chaque disque on peut avoir un ou plusieurs systèmes de fichiers. Parmi eux, il y a un baptisé *système de fichiers racine*. La racine de ce système est la racine de l'arborescence de base de la machine **/** et il est connu par le noyau grâce au choix de l'administrateur à l'installation. Il est actif automatiquement dès que UNIX est en marche. En d'autres termes on peut accéder au contenu du système de fichiers. Quant aux autres systèmes de fichiers, ils doivent être "activés" explicitement pour qu'ils soient accessibles. En effet, il s'agit de placer son arbre dans l'arborescence de base et de faire connaître l'adresse de son support physique. Cette opération est appelée le *montage* et la commande qui s'en charge est **/etc/mount**. On doit choisir un répertoire déjà créé, *dir* par exemple, puis effectuer la commande suivante :

/etc/mount *file* *dir*

où *file* est un fichier spécial désignant le dispositif contenant le système de fichiers et *dir* sera le nom de la racine de ce système. Si *dir* n'était pas vide avant le montage ses fichiers seront inaccessibles tant que le système de fichiers est monté. Ils redeviendront accessibles après le démontage de celui-ci. Cette opération est effectuée grâce à la commande :

/etc/umount *file*