Table des matières

Bibliographie

Les sites http://fr.wikipedia.org/ , http://wiki.mandriva.com/ , http://www.bellamyjc.org/fr/theoriemultiboot1.html et http://www.traduc.org/docs/howto/vf/Ext2fs-Undeletion.html .

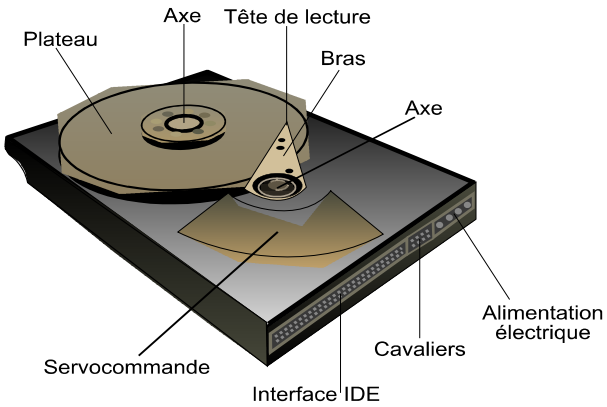
Disque dur (*hard disk*)

Un disque dur, en anglais *hard drive* (HD) ou *hard disk drive* (HDD), est une **mémoire de masse magnétique**.

Principe de fonctionnement

Dans un disque dur, on trouve des plateaux rigides en rotation. Chaque plateau est constitué d'un disque réalisé généralement en aluminium, qui a les avantages d'être léger, facilement usinable et non magnétique. Des technologies plus récentes utilisent le verre ou la céramique, qui permettent des états de surface encore meilleurs que ceux de l'aluminium. Les faces de ces plateaux sont recouvertes d'une couche magnétique, sur laquelle sont stockées les données. Ces données sont écrites en code binaire (0,1) sur le disque grâce à une tête de lecture/écriture. Suivant le flux électrique qui traverse cette tête, elle modifie le champ magnétique local pour écrire soit un 1, soit un 0, à la surface du disque. Pour lire, c'est le même principe inverse qui est utilisé : le champ magnétique local engendre un flux électrique au sein de la tête qui dépend de la valeur précédemment écrite, on peut ainsi lire un 1 ou un 0.

Un disque dur typique contient un axe central autour duquel les plateaux tournent à une vitesse de rotation constante (jusqu'à 15 000 tours/minute). Les têtes de lecture/écriture sont reliées à une même armature qui se déplace à la surface des plateaux, avec une tête par plateau. L'armature déplace les têtes radialement à travers les plateaux pendant qu'ils tournent, permettant ainsi d'accéder à la totalité de leur surface.



Les *firmwares* des disques durs récents sont capables d'organiser les requêtes de manière à minimiser le temps d'accès aux données, et donc à maximiser les performances du disque.

Contrôleur de disque

Un contrôleur de disque est l'ensemble électronique qui est connecté directement à la mécanique d'un disque dur. La mission de cet ensemble est de piloter les moteurs de rotation et de déplacement des têtes de lecture/enregistrement, ainsi que d'interpréter les signaux électriques reçus de ces têtes afin de les convertir en bits ou réaliser l'opération inverse afin d'enregistrer des données à un emplacement particulier de la surface des disques composant le disque dur.

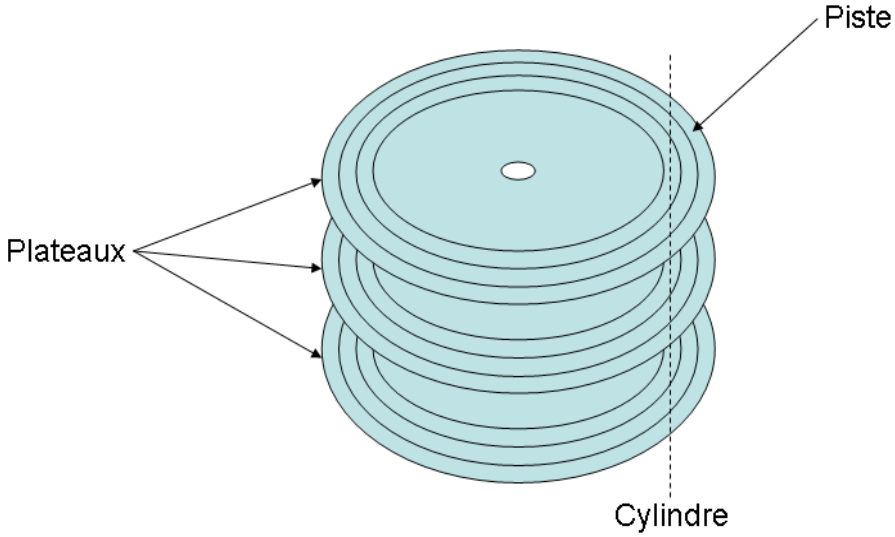
Types d'interface des disques durs

Les principales interfaces possibles :

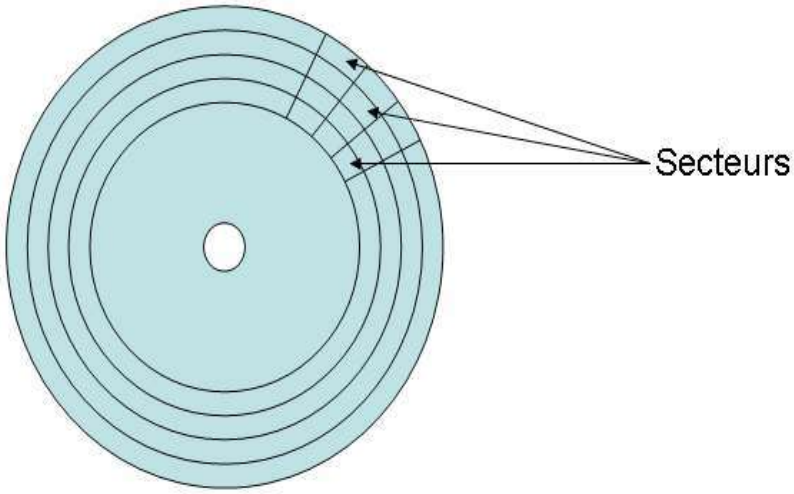
- L'interface **IDE** (ou PATA par opposition au SATA, voir plus loin), la plus courante dans les machines personnelles jusqu'à 2005, appelée aussi ATA (AT ATACHMENT)
- **SCSI** (*Small Computer System Interface*), plus chère, mais offrant des performances supérieures. Toujours utilisée et améliorée (passage de 8 à 16 bits notamment, et augmentation de la vitesse de transfert, normes SCSI-1, SCSI-2, SCSI-3).
- **Serial ATA** (ou S-ATA), est une interface série, peu coûteuse et plus rapide qu'ATA (normes SATA et SATA II), c'est la plus courante désormais (2008).
- **SAS** (*Serial Attached SCSI*), combine les avantages du SCSI avec ceux du Serial ATA (elle est compatible avec cette dernière).
- **Fibre-Channel** (FC-AL), est un successeur du SCSI. La liaison est série et peut utiliser une connectique fibre optique ou cuivre. Principalement utilisée sur les serveurs.

Géométrie

Chaque plateau (possédant le plus souvent 2 surfaces utilisables) est composé de **pistes** concentriques. Les pistes situées à un même rayon forment un **cylindre**. Il faut l'ensemble des **têtes** pour accèder à un cylindre.



La piste est délimitée en **secteurs** (aussi appelés blocs) contenant les données.



En adressage CHS (*Cylinder/Head/Sector*), il faut donc trois coordonnées pour accéder à un bloc (ou secteur) de disque :

1 . le numéro de la tête de lecture (choix de la surface)
2 . le numéro de la piste (détermine la position du bras portant l'ensemble des têtes)
3 . le numéro du bloc (ou secteur) sur cette piste (détermine à partir de quand il faut commencer à lire les données).

Le tout premier secteur d'un disque est à l'adresse 0 / 0 / 1 : c'est le premier secteur accédé par la première tête positionnée sur le premier cylindre. Le suivant sera 0 / 0 / 2 (ce secteur est naturellement atteint juste après par la même tête), et ainsi de suite jusqu'à ce que la surface ait effectué une rotation complète. Le dernier secteur accédé ici porte l'adresse 0 / 0 / NS.

Le secteur suivant est à l'adresse 0 / 1 / 1 : c'est le premier secteur accédé par la tête suivante (idéalement la sélection électronique d'une tête prend moins de temps qu'il ne faut au disque pour présenter de nouveau le secteur numéro 1), puis chaque secteur de la piste qui défile devant cette seconde tête est exploré et ainsi de suite jusqu'à

avoir employé toutes les têtes, le dernier secteur du premier cylindre porte ainsi l'adresse 0 / NT-1 / NS.

Le secteur suivant est à l'adresse 1 / 0 / 1 : le bras des têtes de lecture/enregistrement devra préalablement s'être déplacé (ce qui peut prendre de moins d'une milliseconde à plusieurs centaines de millisecondes) puis l'ensemble des opérations décrites plus haut (le parcourt de chaque secteur de chaque piste) pourra se répéter pour ce cylindre.

Le tout dernier secteur du disque est à l'adresse NC-1 / NT-1 / NS. Le **nombre total de secteurs** accessibles par ce moyen d'adressage (la capacité totale du disque en fait) est simplement **NC × NT × NS**.

Comme le BIOS code le **numéro de cylindre avec 10 bits, le numéro de tête avec 8 bits et le numéro de secteur avec 6 bits**, un disque accédé en CHS n'aura jamais plus de 1024 cylindres, 256 têtes (ce qui est mécaniquement impossible car demanderait 128 plateaux : même le RAMAC d'IBM n'en avait que 50) et 63 secteurs par rotation donc une capacité maximale d'un peu moins de 8 Gio (le produit de ces trois nombres par 512 qui est le nombre usuel d'octets par secteur de données vaut exactement 8 455 716 864 octets soit 7,875 Gio).

L'adressage CHS reste cependant employé dans les premières phases de démarrage d'un ordinateur puisqu'il permet toujours d'accéder aux premiers secteurs d'un disque. Ainsi le BIOS charge le secteur 0 / 0 / 1 du premier disque dur qui est souvent un MBR, ce dernier emploie à son tour une adresse CHS pour charger le secteur de boot de la partition active. C'est l'emploi d'adresses CHS durant cette phase qui fait que beaucoup d'utilitaires disques vous alertent si la partition active se trouve au delà des premiers 8 Go d'un disque (le secteur de boot deviendrait alors inaccessible par une adresse CHS).

Aussi les deux méthodes d'adressages ECHS pour (*Enhanced CHS* en anglais, soit « Cylindre/Tête/Secteur amélioré » en français) puis LBA ont-elles été mises en place à partir de mi-1994 pour contourner les limites des adresses CHS.

Cette conversion est faite le plus souvent par le contrôleur du disque à partir d'**une adresse absolue de bloc appelée LBA** (un numéro compris entre 0 et le nombre total de blocs du disque - 1).

L'adressage en LBA (abréviation de *Logical Block Addressing* en anglais soit « Adressage par bloc logique » en français) est le moyen moderne d'adresser les secteurs de données stockés sur un disque dur. Cette méthode d'adressage a depuis été généralisée à un grand nombre de supports informatiques.

Cette adresse permet de désigner d'une façon unique un secteur de données d'un disque (la plus petite unité de données transférée par ce dernier), sa taille est le plus souvent 512 octets. Deux versions d'adresse LBA ont existé sur les disques durs IDE, une première version utilisant 28 bits pour coder l'adresse et permettant de gérer des disques d'une capacité maximale de 128 Gio soit 2^{28}*512 = 137438953472 octets et une seconde plus récente (en 2002 dans la norme ATA/ATAPI-6) utilisant 48 bits et qui permet de gérer des disques d'une capacité maximale de 128 Pio.

Capacité

La capacité d'un disque dur peut être calculée ainsi : **nombre de cylindres * nombre de têtes * nombre de secteurs par piste * nombre d'octets par secteur** (généralement 512 octets).

Cependant les nombre de cylindres, têtes et secteurs sont fausses pour les disques utilisant le *zone bit recording* (enregistrement à densité constante), ou la translation d'adresses LBA. Sur les disques ATA de taille supérieure à 8 Go, les valeurs sont fixées à **255 têtes, 63 secteurs et un nombre de cylindres dépendant de la capacité réelle** du disque afin de maintenir la compatibilité avec les systèmes d'exploitation plus anciens.

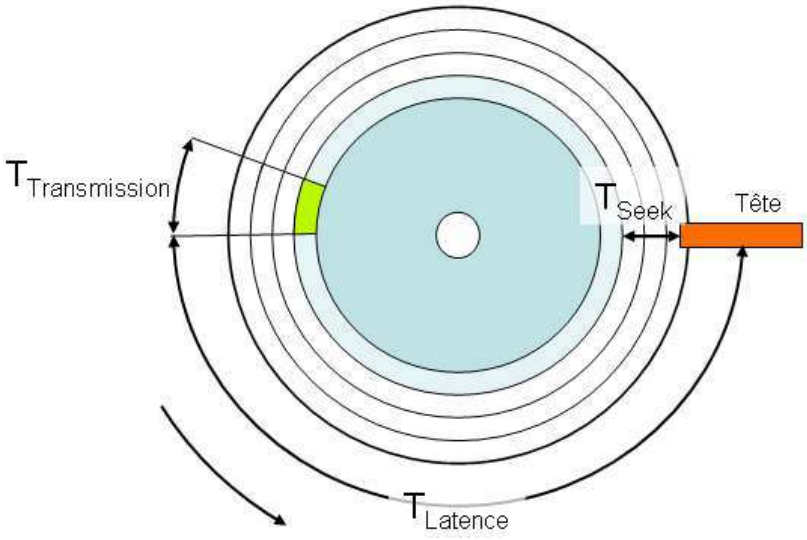
Performances

Le **temps d'accès et le débit d'un disque dur** permettent d'en mesurer les performances. Les facteurs principaux à prendre en compte sont :

1 . le **temps de latence**, facteur de la vitesse de rotation des plateaux. Le temps de latence (en secondes) est égal à 60 divisé par la vitesse de rotation en tours par minute. Le temps de latence moyen est égal au temps de latence divisé par 2 (car on estime que statistiquement les données sont à un demi-tour près des têtes).
2 . le **temps de recherche**, ou *seek time* en anglais, est le temps que met la tête pour se déplacer jusqu'au cylindre choisi. C'est une moyenne entre le temps piste à piste, et le plus long possible (full-stroke).
3 . le **temps de transfert** est le temps que vont mettre les données à être transférées entre le disque dur et l'ordinateur par le biais de son interface.

Pour estimer le temps de transfert total, on additionne ces trois temps. On pourra rajouter le temps de réponse du contrôleur, etc. Il faut souvent faire attention aux spécifications des constructeurs, ceux-ci auront tendance à communiquer les valeurs de pointe au lieu des valeurs soutenues (par exemple pour les débits).

Pour lire le secteur (en vert) situé sur une piste interne à l'opposé de la tête de lecture (en rouge), il faut déplacer la tête vers l'intérieur (TSeek), attendre que le bloc arrive sous la tête (TLatence) puis lire la totalité du bloc (TTransmission). Il est possible d'optimiser le temps d'accès en prenant en compte la vitesse de rotation pendant que la tête se déplace.



Mémoire cache

L'ajout de mémoire vive sur le contrôleur du disque permet d'**augmenter les performances**. Cette mémoire sera remplie par les blocs qui suivent le bloc demandé, en espérant que l'accès aux données sera séquentiel. En écriture, le disque peut informer l'hôte qui a initié le transfert que celui-ci est terminé alors que les données ne sont pas encore écrites sur le média lui-même. Comme tout système de cache, cela pose un problème de cohérence des données.

Émulation

Parfois il est nécessaire d'avoir un périphérique en tout point similaire à un disque dur, mais avec des **temps d'accès beaucoup plus rapides**, au détriment de la capacité. Il y a deux façons d'atteindre ce but : soit par l'utilisation d'un **disque SSD**, soit par la création d'un **disque virtuel** (parfois aussi appelés RAM Disques).

Un **disque virtuel** est un artifice qui permet d'émuler un disque dur à partir d'un espace alloué en mémoire centrale. Sa création, son effacement et son accès se font par le biais d'appels systèmes (le noyau peut contenir des pilotes adéquats). Les temps d'accès sont extrêmement rapides ; en revanche, par construction, leur capacité ne peut excéder la taille de la mémoire centrale.

Les données étant perdues si la mémoire n'est plus alimentée électriquement, on les utilise en général pour des fichiers en lecture seule, copies de données sur disque, ou pour des fichiers intermédiaires dont la perte importe peu.

Exemples :
- rangement de données très souvent consultées (par exemples fichiers .h en langage C)
- rangement de fichiers intermédiaires de compilation (sous Linux, fichiers .o)

Un **SSD** (pour *Solid State Drive*) a extérieurement l'apparence d'un disque dur classique, y compris l'interface, mais est constitué de plusieurs puces de mémoire *flash* et ne contient aucun élément mécanique.

Avantages : les temps d'accès sont très rapides pour une consommation généralement inférieure
Inconvénients : leur capacité est encore limitée, en 2009, on trouve des modèles de 128 Go à des prix d'environ 350 $ ce qui reste nettement plus cher qu'un disque dur. Leur nombre de cycles d'écriture est limité (donc la durée de vie) mais reste largement suffisant pour une utilisation classique.

Source : http://fr.wikipedia.org/wiki/Disque_dur

BIOS (*Basic Input Output System*)

Le BIOS (système élémentaire d'entrée/sortie) est, au sens strict, un **ensemble de fonctions**, contenu dans la mémoire morte (ROM) de la carte mère d'un ordinateur lui **permettant d'effectuer des opérations élémentaires** lors de sa mise sous tension, par exemple la lecture d'un secteur sur un disque.

Par extension, le terme est souvent utilisé pour décrire l'ensemble du micrologiciel (« logiciel embarqué » ou « ***firmware*** ») de la carte mère.

Le BIOS est écrit en code machine et a généralement été développé en langage assembleur. Il est presque toujours développé par le fabricant de cette carte mère car il contient les routines élémentaires pour effectuer les opérations simples d'entrée/sorties.

Boot BIOS : POST (Power-On Self-Test)

Le BIOS comprend également le **POST** (*Power-On Self-Test*), exécuté au démarrage de l'ordinateur, qui déclare les disques, configure les composants et recherche un système d'exploitation avant de le lancer. Sa tâche principale est de fournir un support de bas niveau pour communiquer avec les périphériques. Habituellement ceci inclut le support du clavier au moins dans un mode (pas forcément l'USB) et d'un mode d'affichage simplifié.

Le BIOS émet les premières commandes au système durant la phase de démarrage, pour indiquer par exemple sur quel disque et à quel endroit de celui-ci trouver le **chargeur d'amorçage** (ou *Boot loader*) du système d'exploitation, en général Windows, Linux, Mac OS ou autre. Dans le cas de Windows, il s'agit du NTLDR, dans le cas d'un système en multiboot (possibilité de démarrer plusieurs systèmes d'exploitation sur un même ordinateur), Lilo dans les cas simples (Windows et Linux), GRUB dans les cas plus sophistiqués (tous systèmes supportés).

Le BIOS contient également des outils de diagnostic pour vérifier sommairement l'intégrité des composants critiques comme la mémoire, le clavier, le disque dur, les ports d'entrée/sortie, etc.

Mémoire CMOS (Setup)

Certains paramètres du BIOS peuvent être réglés par l'utilisateur (ordre des périphériques à scruter pour détecter une zone de boot, type et fréquence du processeur, etc.).

L'ensemble de ces paramètres est stocké de manière permanente grâce à une mémoire de taille réduite (quelques centaines d'octets) à faible consommation (type CMOS) alimentée par une pile (généralement au lithium) présente sur la carte mère. Cette mémoire est communément appelée, par abus, « CMOS ».

Services BIOS

Historiquement, en plus des fonctions de diagnostic et de configuration, le BIOS fournit un **ensemble de services permettant de faire le plus abstraction possible de la couche matérielle**. Peu importe comment le fabricant a développé cette carte mère (quels composants il a choisi, peu importe comment fonctionne le « hardware » de cette carte mère), en utilisant les mêmes fonctions du BIOS sur deux cartes mères différentes, on obtiendra le même résultat. Ce sont ces fonctions (services) que les systèmes d'exploitation utilisent pour faire fonctionner les applications. Cependant, dans la pratique, les systèmes d'exploitation récents utilisent peu ces services.

Firmware BIOS

Le BIOS est parfois appelé *firmware*, car il est très proche du hardware. Avant les années 1990, les BIOS étaient stockés sur des puces ROM qui ne pouvaient être modifiées. Au fur et à mesure que leur complexité, et le besoin de mises à jour se sont fait sentir, ils furent stockés sur des mémoires EEPROM ou Flash qui pouvaient être modifiées.

Ainsi, il est possible de mettre à jour, de manière logicielle, le BIOS d'un ordinateur. Cette action est appelée « Flasher le BIOS ».

Partition

Une partition est une **partie d'un disque dur destinée à accueillir un système de fichiers**. Par exemple, pour parler d'une partition accueillant un système de fichiers FAT32, on parle couramment de « partition FAT32 ».

Le **partitionnement est un fractionnement d'un disque dur réel (matériel) en plusieurs disques virtuels (logiciels)**.

Chaque partition possède donc son système de fichiers, qui permettra de stocker ensuite les données.

Pour rappel, le fichier est la plus petite entité logique de stockage sur un disque.

Un disque peut contenir **une ou plusieurs partitions**.

Lorsqu'il contient plusieurs partitions, celles-ci apparaissent au système d'exploitation comme des disques (ou « volumes ») séparés. Dans Windows, elles auront généralement des lettres de lecteur différentes (C:, D:, etc.). Dans Mac OS, elles apparaissent en général chacune avec son icône propre sur le Bureau. Sous UNIX, elles sont cachées de l'utilisateur final, les fichiers étant accédés à travers l'arborescence unique (ainsi d'ailleurs que les périphériques physiques) mais sont visibles à travers diverses commandes d'administration, notamment celles affichant les points de montage (mount, df).

On nomme « **partition d'amorçage** », celle qui prend le contrôle au démarrage, qu'elle contienne ou non le système d'exploitation.

Un disque dur peut être partitionné pour différentes architectures. On aura ainsi le partitionnement de type MBR pour la majorité des ordinateurs personnels (PC) ou GPT pour les architectures plus récentes (Macintosh).

Tables de partitions

Les informations sur les partitions sont conservées sur le disque lui-même dans des zones qu'on appelle tables de partitions. La table de partitions principale est contenue dans le premier secteur du disque ou secteur d'amorçage (*Master Boot Record* ou **MBR**) qui contient également le programme d'amorçage. Chaque ligne d'une table de partitions contient l'adresse de début de la partition et sa taille. Il peut s'agir de partitions primaires qui contiendront un système de fichiers ou de partitions étendues qui contiendront à leur tour une table de partitions ayant la même structure que la table principale.

Partitions primaires

Seules les partitions primaires peuvent contenir la partition d'amorçage du système d'exploitation Windows.

Or il existe des restrictions sur les tables de partitions, certaines liées à la place occupée dans le secteur d'amorçage, d'autres pour simplifier le fonctionnement du système d'exploitation.

Dans la table de partitions principale, on peut créer **au plus quatre partitions**, soit quatre partitions primaires, soit de 1 à 3 partitions principales puis une partition étendue (qui souvent est la dernière).

Sur un ordinateur de type PC, un identificateur associé à chaque partition permet de connaître a priori quel type de système de fichier elle abrite. Cet identificateur occupe un octet.

Partition étendue, partitions secondaires, etc ...

Lorsque l'on veut plus de quatre partitions, il faut donc créer une partition étendue. Cette dernière n'est ni plus ni moins qu'une partition primaire spéciale qui va contenir des partitions secondaires (ou lecteurs logiques pour MS-DOS et Windows). Une partition étendue peut donc **contenir plusieurs partitions secondaires, qui sont au nombre de 4** et ne se distinguent pas pour un programme utilisateur (ni pour le système) des autres partitions.

Seules les partitions primaires sont directement reconnues par le BIOS. La table de partition étendue est contenue dans l'**EBR**. L'EBR peut, lui aussi à son tour, contenir une partition étendue qui contiendra à ce moment là des partitions tertiaires et ainsi de suite. L'EBR est une structure située en tête d'une partition étendue, sur le disque dur d'un PC. Son contenu est identique à celui du MBR.

Partition et disque

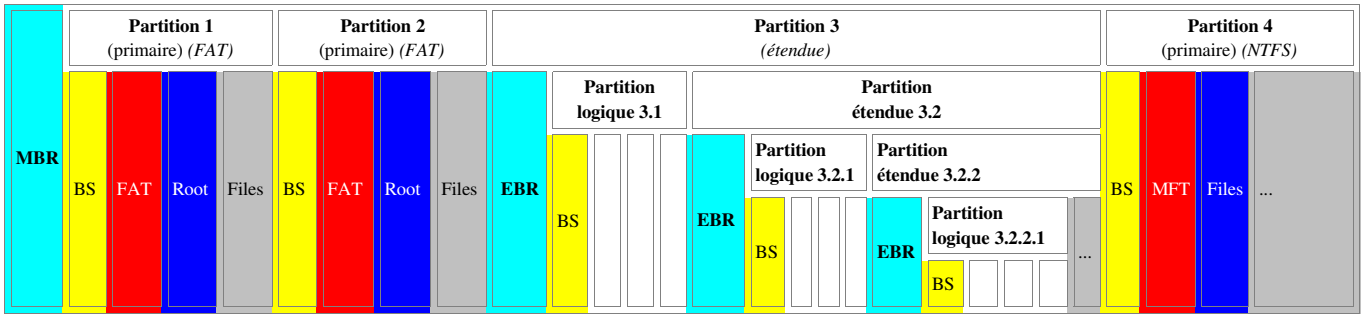
Dans un disque PHYSIQUE, le secteur 1, piste 0, cylindre 0, ("MBR") contient une table définissant 4 partitions au maximum. Ces partitions sont soit primaires, soit étendue (une seule d'entre elles pouvant être étendue).

Ces partitions démarrent toujours au secteur n°1 d'une piste :

- à la piste n°1 d'un cylindre pour la 1ère partition primaire
- à la piste n°0 d'un cylindre pour les autres

Espace perdu : on perdra donc soit une piste complète (1 piste = 63 secteurs * 512 octets = 32 256 octets soit environ 32 KiO) soit un cylindre complet (1 cylindre = 255 têtes * 63 secteurs = 16065 secteurs * 512 octets = 8 225 280 octets soit environ 7,8 moi).

Exemple détaillé



Abréviation	*Signification*	*Commentaire*
MBR	*Master Boot Record*	secteur de partition principal
EBR	*Extended Boot Record*	secteur de partition secondaire : contient la table des partitions d'une partition étendue
BS	*Boot Sector*	secteur de boot
Root	*répertoire racine*	Limité en taille sous FAT12 ou FAT16
FAT	*File Allocation Table*	table de chaînage de clusters (agrégats de secteurs) permettant l'adressage et la reconstitution des fichiers a donné son nom au type de partition du même nom, utilisé par DOS, Windows 95/98, Windows NT
NTFS	*NT File Sytem*	système de fichiers utilisé par Windows NT/W2K/XP
MFT	*Master File Table*	composant important d'une partition NTFS. Contient la table des fichiers, index, droits,...

Ce disque (physique) est constitué de :

- 3 partitions PRIMAIRES
- 3 partitions LOGIQUES (les partitions étendues n'étant que des « conteneurs »)

Outils

Voici une liste non-exhaustive d'outils permettant de créer des partitions et (pour certains d'entre-eux) de formater ces partitions :

- **fdisk** : Utilitaire utilisé sous DOS, Linux et Mac OS X pour manipuler les partitions et la table de démarrage. Bien que le nom de cet outil soit identique sur ces trois systèmes, il ne s'agit absolument pas du même programme. Sous DOS l'utilitaire est présenté sous forme de menus ; sous Linux, sous forme d'un outil en ligne de commande (prompt) ; sous MacOSX, il faut ouvrir une fenêtre Terminal pour l'utiliser de manière interactive.
- **DiskDrake** est un utilitaire de Mandriva permettant de créer, enlever, redimensionner les partitions via une interface visuelle.
- **cfdisk** est un utilitaire disponible sous Linux pour manipuler les partitions. Il est présenté sous forme de menus à la manière de fdisk sous DOS.
- **PartitionMagic** est un utilitaire payant de la société Symantec Corporation fonctionnant sous DOS ou Windows. Norton PartitionMagic est notamment reconnu pour pouvoir redimensionner, déplacer ou fusionner une ou des partitions sans perdre les données qu'elle contient. Il permet diverses opérations impossibles à réaliser avec les outils fournis en standard par Microsoft comme par exemple convertir un système de fichiers NTFS en FAT (l'inverse est possible sous Windows (NT) grâce à la commande convert) ou réparer une table de partitions endommagée. Attention cependant à la compatibilité avec vista. Il est connu un phénomène de disparition de partition difficilement réversible ...
- **GNU Parted** est un utilitaire GNU en ligne de commande ou avec une interface graphique fonctionnant sur le système d'exploitation Linux et permettant d'effectuer diverses opérations impossibles avec des outils standards comme le redimensionnement de partitions par exemple. Parted est une interface à la bibliothèque libparted, qui constitue réellement le noyau de l'utilitaire (page officielle du projet Parted)
- **QtParted** se définit comme « un clone de Partition Magic écrit en C++ en utilisant la bibliothèque graphique Qt ». Ce dernier est en fait une interface graphique au programme GNU Parted, ou plus

précisément, libparted.

- **GParted** est une interface graphique au programme GNU Parted (libparted). Il utilise la bibliothèque graphique GTK et peut manipuler de nombreux formats de systèmes de fichiers. (Fonctionnalités de Gparted) . À noter que Gparted existe en version LiveCD qui permet de préparer/modifier des partitions sans système installé sur la machine.

Master Boot Record (MBR)

Le *Master Boot Record* ou MBR (parfois aussi appelé "Zone amorce") est le **nom donné au premier secteur adressable d'un disque dur** (cylindre 0, tête 0 et secteur 1, ou secteur 0 en adressage logique) dans le cadre d'un partitionnement Intel. Sa **taille est de 512 octets**.

Le MBR contient :

- la **routine d'amorçage** dont le but est de charger le système d'exploitation (ou le *boot loader*/chargeur d'amorçage s'il existe) présent sur la partition active
- la **table des partitions** (les 4 partitions primaires) du disque dur.

Structure du MBR

Adresse	Contenu	Taille (octets)
0x000	**Programme** (**routine d'amorçage**)	442
0x1B8	**Signature** (sous Windows NT/2000/XP/2003)	4
0x1BE	**1**ère entrée dans la table de partition	16
0x1CE	**2**ème entrée dans la table de partition	16
0x1DE	**3**ème entrée dans la table de partition	16
0x1EE	**4**ème entrée dans la table de partition	16
0x1FE	**AA55** (code d'identification)	2
	Total	*512*

Structure d'une entrée dans la table de partition

Adresse	Offset dans le MBR (ou EBR) Entrées :				Contenu	Taille (octets)
	1	2	3	4		
0x00	**0x1BE**	**0x1CE**	**0x1DE**	**0x1EE**	**Etat de la partition** : - 00 : partition non active - 80 : partition active	**1**
0x01	**0x1BF**	**0x1CF**	**0x1DF**	**0x1EF**	**N° de tête** où commence la partition	**1**
0x02	**0x1C0**	**0x1D0**	**0x1E0**	**0x1F0**	**N° de secteur** et **cylindre** où commence la partition	**2**
0x04	**0x1C2**	**0x1D2**	**0x1E2**	**0x1F2**	**Type de partition**	**1**
0x05	**0x1C3**	**0x1D3**	**0x1E3**	**0x1F3**	**N° de tête** où finit la partition	**1**
0x06	**0x1C4**	**0x1D4**	**0x1E4**	**0x1F4**	**N° de secteur** et **cylindre** où finit la partition	**2**
0x08	**0x1C6**	**0x1D6**	**0x1E6**	**0x1F6**	**Distance** en secteurs entre secteur de partition et secteur de boot de la partition	**4**
0x0C	**0x1CA**	**0x1DA**	**0x1EA**	**0x1FA**	**Taille** de la partition en **nombre** de secteurs de 512 octets	**4**

Manipulations du MBR :
Sous **MS-DOS** et les versions grand public de **Windows** jusqu'à Windows Millenium, il est possible de recréer la routine de boot du MBR sous DOS à l'aide de la commande **FDISK /MBR**. Le *Master Boot Record* est ainsi réécrit. Cela permet d'éliminer certains virus de boot (si la commande est exécutée depuis une disquette car les virus de boot détournent souvent l'interruption 13h), de restaurer un MBR endommagé (le PC ne démarre plus), ou de supprimer un chargeur de démarrage installé dans le MBR (lilo, GRUB, etc.).

Pour sauvegarder et restaurer le MBR sous **windows**, il faut utiliser le programme **debug**.

Sous **UNIX et Linux**, la commande **dd** (il faut évidemment prendre soin de sauvegarder et restaurer uniquement le premier secteur, soit les 512 premiers octets du disque).

Sous **Windows XP**, la commande à utiliser pour restaurer le MBR est **fixmbr** (fixboot). Elle est accessible depuis la console de récupération.

Sous **Windows Vista**, la commande à utiliser pour restaurer le MBR est **bootrec /FixMbr**. Elle est accessible depuis la console de récupération.

ATTENTION : il est très risqué de restaurer le MBR d'un disque dur sur un autre, car cela remplacerait la table des partitions du second disque par celle du premier ! La seule exception à cette règle est si les deux machines ont des configurations matérielles strictement identiques, notamment si les disques durs sont les mêmes ainsi que leur partitionnement (cas d'un parc de machines en entreprises).

Il est à remarquer que si le MBR initial pointait vers un chargeur d'amorçage de Linux (GRUB, LILO), celui-ci devient inaccessible après cet écrasement. Il faut alors démarrer sur un CD-ROM Linux (installation/restauration) et le restaurer par ce moyen.

Chargeur d'amorçage (bootloader)

Un chargeur d'amorçage (ou *bootloader*) est un logiciel permettant de lancer un système d'exploitation parmi plusieurs (*multi-boot*), c'est-à-dire qu'il permettra d'utiliser plusieurs systèmes, à des moments différents, sur la même machine.

Dans le cas le plus simple, il n'y a qu'une seule partition du disque de boot : le micrologiciel BIOS charge les 512 premiers octets de ce disque, ces 512 octets constituant le MBR. À partir des informations du MBR, il détermine l'emplacement de la routine d'amorçage.

Si le disque de boot a plusieurs partitions, le micrologiciel BIOS lit le MBR du disque, puis le VBR de la partition (*Volume Boot Record*). À partir de ces informations, il peut déterminer l'emplacement du chargeur d'amorçage et le lancer.

Si le support de boot est une disquette, c'est le VBR de cette disquette qui est utilisé pour déterminer l'emplacement du chargeur d'amorçage.

Sur certains PC actuels, c'est le micrologiciel EFI (et non pas le BIOS) qui est utilisé pour lancer le chargeur d'amorçage : l'EFI lit la GPT du disque (*GUID Partition Table*) pour déterminer l'emplacement de la routine d'amorçage.

Les chargeurs d'amorçage les plus usuels sont :

- Société Microsoft :
  - NTLDR (*NT LoaDeR* ou Chargeur d'amorçage de Windows NT) avec le BIOS. Sa configuration est stockée dans le fichier boot.ini.
  - IA86ldr.efi et IA64ldr.efi avec l'EFI
  - pour le système d'exploitation Vista : le chargeur d'amorçage est winload.exe et sa configuration est stockée dans une ruche du registre : BCD (*Boot Configuration Data*)
- Open source :
  - GRUB (*GRand Unified Bootloader*)
  - LILO (*Linux loader*) pour le BIOS et elilo pour EFI
  - IsoLinux de Syslinux pour booter à partir d'un DVD ISO 9660
  - PXELinux de Syslinux pour booter à partir d'une carte réseau
- Apple : Boot Camp

GUID (Globally Unique Identifier) et UUID (Universally Unique Identifier)

Un **GUID** (abréviation de l'anglais *Globally Unique IDentifier*) sert habituellement d'**identifiant unique pour un composant logiciel**, par exemple un plugin. Sa taille est de 16 octets, soit 128 bits (exemple : {3F2504E0-4F89-11D3-9A0C-0305E82C3301}).

Ce terme est utilisé à la fois dans le monde Microsoft et dans le monde Unix (voir RFC 4122). Microsoft utilise le terme de **CLSID** (de l'anglais *CLasS IDentifier*) pour désigner le GUID de la classe d'un objet OLE, et le terme IID (de l'anglais *Interface IDentifier*) pour les interfaces implémentées par ces classes.

Sous Unix/Linux, on utilise le **UUID**. UUID est l'abréviation du terme anglais *Universally Unique Identifier* (identifiant unique universel) utilisé en informatique. Il s'agit d'un standard défini initialement par l'OSF (*Open Software Foundation*). La dernière version de ce standard est définie par le RFC 4122, en 2005.

Ces identifiants uniques sont codés sur 128 bits et sont produits en utilisant des composantes pseudo-aléatoires ainsi que les caractéristiques d'un ordinateur (numéro de disque dur, adresse MAC, etc.). Un UUID se présente habituellement sous cette forme : 110E8400-E29B-11D4-A716-446655440000

Un UUID est conçu de manière à être unique dans le monde ; cependant, aussi infime qu'il soit, le risque existe que deux ordinateurs produisent un même identifiant. Les UUID sont destinés à l'identification de composants logiciels (plugins), des différents membres dans un système distribué ou d'autres applications nécessitant une identification sans ambiguïté. Les UUID peuvent être générés sur les systèmes Unix via la commande uuidgen. Sous Linux, le paquet e2fsprogs contient une commande /sbin/blkid qui permet d'afficher un UUID correspondant à certains périphériques.

```
# blkid
/dev/sda1: UUID="5ca2063f-5863-41be-a3c6-625957698b3c" SEC_TYPE="ext2" TYPE="ext3"
/dev/sda5: TYPE="swap" UUID="a1d386b8-dad8-46d7-b54f-ed46ee44f16b"
/dev/sda6: UUID="f31cca75-ce58-4e26-922f-ba3a5ddef95b" SEC_TYPE="ext2" TYPE="ext3"
/dev/sda7: UUID="8b38665e-1cc9-401a-a5dc-c8db046cb9c2" SEC_TYPE="ext2" TYPE="ext3"
/dev/sdb1: UUID="A034789234786D62" TYPE="ntfs"
/dev/sdb5: UUID="d7d1448d-6e20-48b5-a8cc-2bb6df3d2514" TYPE="ext3" SEC_TYPE="ext2"
/dev/sdb6: TYPE="swap" UUID="a9f13f98-6be6-11de-a414-dfdad8185f33"
/dev/sdc1: UUID="a8fcdd63-dc78-4230-8d27-0edce4a7d7a7" SEC_TYPE="ext2" TYPE="ext3"
/dev/sdc5: UUID="135a531a-3b21-4d2b-8826-7ea648b3d6d7" SEC_TYPE="ext2" TYPE="ext3"
/dev/sdd1: LABEL="Elements" UUID="A1FC-FC74" TYPE="vfat"
```

Systèmes de fichiers (*File System*)

Un système de fichiers définit l'organisation d'un disque (ou partition d'un disque). C'est donc une structure de données permettant de stocker les informations et de les organiser dans des fichiers sur ce que l'on appelle des mémoires secondaires (disque dur, disquette, CD-ROM, clé USB, disques SSD, etc.). Il offre à l'utilisateur une vue abstraite sur ses données et permet de les localiser à partir d'un chemin d'accès.
Pour rappel, le fichier est la plus petite entité logique de stockage sur un disque.

Formatage

Le formatage est l'action de formater, c'est-à-dire de préparer un support de données informatique (disquette, disque dur, etc.) en y inscrivant un système de fichiers, de façon à ce qu'il soit reconnu par le système d'exploitation de l'ordinateur. Il existe de nombreux systèmes de fichiers différents : FAT, NTFS, HFS, ext2, ext3, UFS, etc.

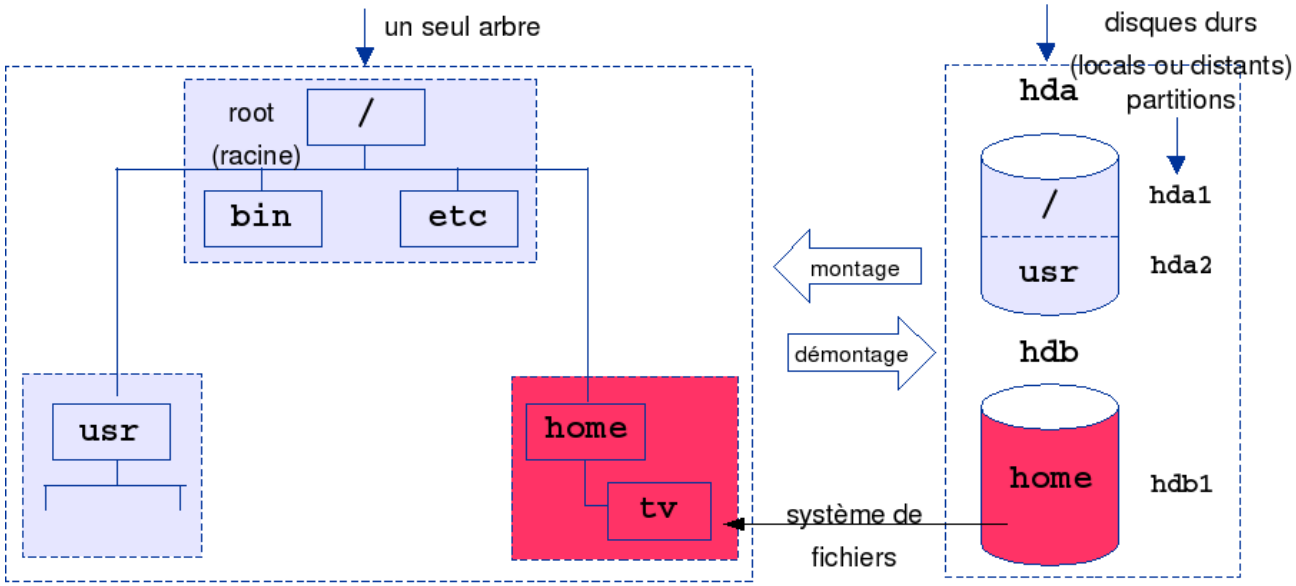
Les disques de grande capacité peuvent recevoir plusieurs systèmes de fichiers, divisés en partitions logiques ; on parle alors de partitionnement. En pratique, on partitionne surtout des disques durs.

Le formatage fait appel à deux processus différents connus sous les noms de **formatage de bas niveau** et **formatage de haut niveau**. Le formatage de bas niveau s'occupe de rendre la surface du disque conforme à ce qu'attend le contrôleur tandis que le formatage de haut niveau concerne les informations logicielles propres au système de fichiers (système d'exploitation).

Les fabriquants effectuent le formatage de bas niveau à la sortie d'usine. Il ne reste plus qu'à lancer un formatage de haut niveau pour inscrire le système de fichiers.

Arborescence

Sous Unix/Linux, les utilisateurs voient une arborescence de fichiers unique. Cet arbre est en fait l'unification de plusieurs systèmes de fichiers :



Dans un système Windows, les périphériques de stockage de données et les partitions sont affichés comme des lecteurs indépendants en haut de leur propre arborescence.

Si l'on considère par exemple un système comprenant :

- une partition de disque dur où est installé le système (Windows ou Unix) ;
- une partition de disque dur où se trouvent les données des utilisateurs ;
- un lecteur de disquette.
- un lecteur de cdrom.

Sous Windows, on accédera alors à ces données de manière séparée :

- **C:** : partition système du disque dur ;
- **D:** : partition utilisateur du disque dur ;
- **A:** : disquette (accès-type : A:\chemin\fichier).
- **E:** : cdrom (accès-type : E:\chemin\fichier).

Sous Unix, l'accès se fera à partir de la racine **/** :

- **/** : première partition système du disque dur ;
- **/home** : partition utilisateur du disque dur ;
- **/mnt/floppy** : disquette (accès-type : /mnt/floppy/chemin/fichier).
- **/mnt/cdrom** : cdrom (accès-type : /mnt/cdrom/chemin/fichier).

Le montage et démontage

Le **montage** d'un système de fichiers consiste à l'**attacher** à un répertoire (point de montage) d'un système de fichiers déjà actif (l'arbre racine). Cette opération est réalisée par la commande **mount**. Les fichiers et répertoires de ce système de fichiers sont donc accessibles aux utilisateurs (cp, rm, mv, ...).

L'opération de démontage, réalisée par la commande **umount**, **détache** le système de fichiers de l'arbre racine. Les fichiers et répertoires de ce système de fichiers ne sont donc plus accessibles aux utilisateurs.
Lors du démarrage, le disque qui contient le système de fichiers principal (arbre racine) doit être connu pour être monté automatiquement.

Un **point de montage est un répertoire** à partir duquel sont accessibles les données se trouvant sous forme d'un système de fichiers sur une partition de disque dur ou un périphérique.

Dans les systèmes Unix, le point de montage par défaut est /mnt ou /media. Par exemple, une disquette sera généralement montée en /mnt/fd0 et un cdrom en /mnt/cdrom ou /media/cdrom. Le point de montage par défaut des périphériques (au démarrage du système) est spécifié dans un fichier de configuration système sous Linux : **/etc/fstab** (/etc/vfstab sous Solaris).

Pour monter un périphérique ou une partition avec la commande **mount**, il faut indiquer :

- le type du système de fichiers par l'option -t
- le fichier spécial représentant le périphérique ou la partition (généralement /dev/*) ;
- le répertoire de montage.

Par exemple, la commande ci-dessous monte le périphérique /dev/cdrom (cédérom) sur /media/cdrom en indiquant que le système de fichier est ISO 9660 :

```
mount -t iso9660 /dev/cdrom /media/cdrom
```

Lorsque le montage a réussi, une mise à jour est effectuée dans un fichier système recensant les montages en cours (fichier **/etc/mtab** sous Linux ou /etc/mnttab sous Solaris)

Pour démonter une partition ou un périphérique, il faut utiliser la commande umount. Par exemple :

```
umount /media/cdrom
```

Le démontage ne marche que si la partition n'est pas utilisée, à savoir :

- aucun fichier n'est en train d'être lu ou écrit sur la partition ;
- aucun processus n'a son répertoire de travail sur la partition.

Si le démontage est refusé, on peut utiliser la commande fuser pour savoir quels processus l'utilisent. Par exemple (si le démontage de /media/cdrom est refusé) :

```
fuser -v -m /media/cdrom
```

Pour envoyer à chaque processus, un signal SIGTERM (qui priera le processus de se terminer proprement) :

```
fuser -k -TERM -v -m /media/cdrom
```

ou en envoyant à chaque processus un signal SIGKILL (qui le tuera d'autorité) :

```
fuser -k -v -m /media/cdrom
```

Lorsque le démontage a eu lieu, le fichier /etc/mtab est mis à jour.

Remarques :
On peut également sous les Unix modernes monter des fichiers qui constituent un système de fichiers à eux-seuls (*loopback*), grâce à l'option -loop sous Linux. Ceci est particulièrement utile dans le cas d'images (.iso) représentant des disquettes, CDROMs, DVDs. Les commandes dd et mkisofs peuvent aider à fabriquer de tels fichiers.

La structure d'un système de fichiers

Un système de fichiers définit l'organisation d'un disque (ou partition d'un disque). C'est donc une structure de données. Sa structure dépend du système de fichiers utilisé (FAT, NTFS, ext2, …).

Dans le cas d'un système de fichiers ext2 (ou ext3), on aura l'organisation suivante :

- le **super bloc** qui contient les informations clés concernant le FS
- la **table des inodes** (la table des descripteurs des fichiers). Chaque fichier physique est identifié de manière unique par un numéro d'inode.
- les **répertoires** qui assurent une correspondance entre un nom de fichier et un numéro d'inode.
- les **fichiers** identifié par un numéro d'inode. Un fichier est une suite non ordonnée d'octets.

Disque et système de fichiers

L'organisation physique d'un système informatique tient compte des principes suivants :

- Un disque peut contenir plusieurs partitions (commande **fdisk**).
- On peut créer au plus quatre partitions (soit quatre partitions primaires, soit de 1 à 3 partitions principales puis une partition étendue)
- Une partition étendue contiendra des partitions secondaires (4 max)
- Une partition ne peut contenir qu'un seul système de fichiers.
- Un système de fichiers ne peut s'étendre sur plusieurs disques ou partitions.
- Les disques amovibles possèdent aussi un système de fichiers.

Les différents types de système de fichiers

Il existe de nombreux systèmes de fichiers :

- mimix : le premier FS utilisé par Linux
- **ext2** : le FS standard du système Linux
- msdos : le FS FAT16 de MSDOS et Windows
- vfat : le FS FAT32 de Windows
- smb : le FS réseau utilisant le protocole SMB de Microsoft
- nfs : le FS réseau de Sun
- ntfs : le FS de Windows NT
- iso9660 : le FS utilisé par les CD-ROM

Et des systèmes de fichiers **journalisés** : **ext3**, **ReiserFS**, XFS, ...

Remarques : L'avantage d'un système de fichiers journalisé est de maintenir en permanence la cohérence des métadonnées (données relatives aux structures d'un système de fichiers: emplacement des fichiers, inodes, etc...). Au reboot après un crash (problème matériel, bug du noyau etc), le système va relire le journal, examiner les transactions non terminées, et remettre un système de fichiers sain. En revanche, la journalisation offre peu de garantie quant au contenu des fichiers eux-même. Certains FS supportent les transactions atomiques : une modification sur le disque est effectuée ou non, mais elle n'est jamais effectuée "à moitié".

D'autres systèmes de fichiers : CODA, dtfs, LFS, GFS, PVFS, MOSIX (cf. http://www.inforlad.net/info/computer/linux.htm)

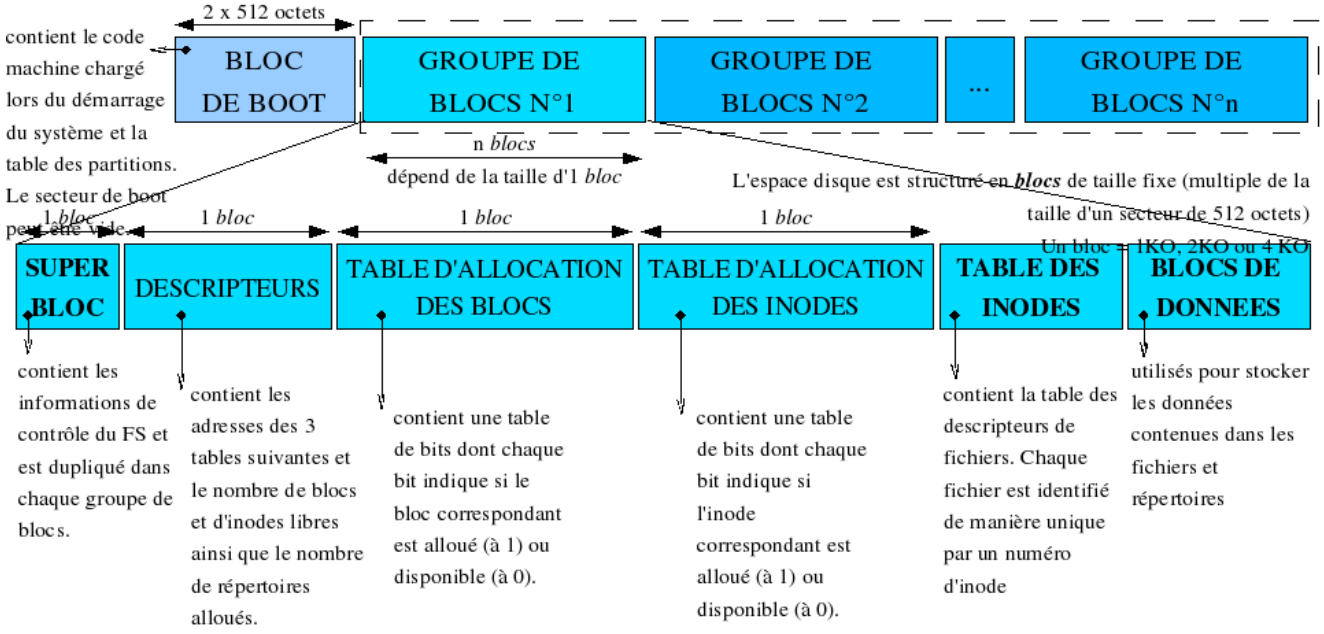
Comparaison des systèmes de fichiers :
http://fr.wikipedia.org/wiki/Comparaison_des_syst%C3%A8mes_de_fichiers

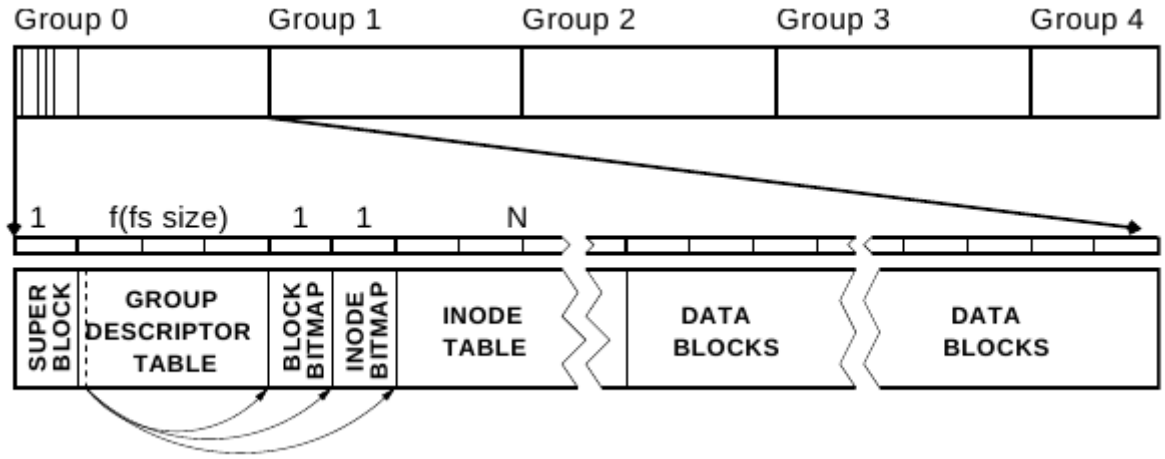
Le système de fichiers ext2

ext2 ou ext2fs (en anglais, *second extended file system*) est le système de fichiers historique de GNU/Linux. Il a été créé à l'origine par Rémy Card, un développeur français.

Structure du système de fichiers ext2

Un système de fichiers **ext2** doit être présent sur un périphérique physique (disquette, disque dur, ...) suivant la décomposition suivante : **`/usr/src/linux/include/linux/ext2_fs.h`**





Remarques :
La table des inodes a une taille fixée statiquement à la création du système de fichiers qui fixe donc le nombre maximal de fichiers qu'elle pourra contenir.

Cette table contient plusieurs entrées réservées :

- premier inode mémorise les « *bads blocs* » du système de fichiers
- second inode représente le répertoire racine du système de fichiers
- etc ... Le premier inode disponible est le 11.

Un bloc de données représente 1, 2 ou 4 Koctets (valeur fixée à la création du FS). Par défaut, **un bloc a une taille de 4096 octets** (4 KiO).

La taille maximale d'une partition ext2/ext3 dépend de la taille de bloc (16 Tio pour un bloc de 4 Kio).

La destruction d'un fichier consiste à libérer l'inode et à restituer les blocs d'adresses et de données au groupe des blocs libres.

La table de descripteurs de groupe (GDT ou *group descriptor table*) contient un ou plusieurs blocs qui contiennent des descripteurs de groupe. Un descripteur de groupe contient l'emplacement du block bitmap, inode bitmap, et le début de la table des inodes pour son groupe de bloc. Il contient également le nombre de blocs libres, inodes libres, et des répertoires attribué pour son groupe.

Il y a un descripteur de groupe pour chaque groupe dans le système de fichiers, de sorte que le nombre de blocs qui compose le descripteur de groupe dépend du nombre de groupes dans le système de fichiers, ce qui à son tour, dépend de la taille du système de fichiers.

Parce que la table de descripteurs de groupe est une donnée critique du système de fichiers, des copies de sauvegarde de la table des descripteurs de groupe sont placés dans les mêmes groupes que la sauvegarde des superblocs. Il existe des copies de sauvegarde du superbloc stockées dans les groupes de bloc avec des nombres entiers qui sont des puissances de 3, 5 et 7 (soit 1, 3, 5, 7, 9, 25, 27, 49, ...).

Reserved GDT blocks : des blocs ont été réservés afin de permettre un redimensionnement en ligne.

Exemple d'une partition ext3

```
# dumpe2fs /dev/sdc1 | more
dumpe2fs 1.41.4 (27-Jan-2009)
...
Filesystem OS type:       Linux
Inode count:              643376
Block count:              2572400
Reserved block count:     128620
Free blocks:              1136364
Free inodes:              399395
First block:              0
Block size:               4096
Reserved GDT blocks:      628
Blocks per group:         32768
Inodes per group:         8144
Inode blocks per group:   509
First inode:              11
Inode size:               256
Taille du journal:        128M

Groupe 0 : (Blocs 0-32767)
 superbloc Primaire à 0, Descripteurs de groupes à 1-1
  Blocs réservés GDT à 2-629
 Bitmap de blocs à 630 (+630), Bitmap d'i-noeuds à 631 (+631)
 Table d'i-noeuds à 632-1140 (+632)
  24120 blocs libres, 3021 i-noeuds libres, 614 répertoires
 Blocs libres : 6058-12287, 12821-14335, 14345-30719
 I-noeuds libres : 5124-8144
```

```
Groupe 1 : (Blocs 32768-65535)
 superbloc Secours à 32768, Descripteurs de groupes à 32769-32769
  Blocs réservés GDT à 32770-33397
 Bitmap de blocs à 33398 (+630), Bitmap d'i-noeuds à 33399 (+631)
 Table d'i-noeuds à 33400-33908 (+632)
  9230 blocs libres, 4955 i-noeuds libres, 720 répertoires
 Blocs libres : 53745, 54069-55295, 55387-57343, 57587, 57589-59391, 59552-61439, 61553-63487,
65111-65124, 65126-65131, 6513
8-65535
 I-noeuds libres : 11334-16288
Groupe 2 : (Blocs 65536-98303)
 Bitmap de blocs à 65536 (+0), Bitmap d'i-noeuds à 65537 (+1)
 Table d'i-noeuds à 65538-66046 (+2)
  31231 blocs libres, 5601 i-noeuds libres, 718 répertoires
 Blocs libres : 66105-75775, 75945-77823, 78612-90111, 90123-98303
 I-noeuds libres : 18832-24432
...
Groupe 78 : (Blocs 2555904-2572399)
 Bitmap de blocs à 2555904 (+0), Bitmap d'i-noeuds à 2555905 (+1)
 Table d'i-noeuds à 2555906-2556414 (+2)
  11809 blocs libres, 3057 i-noeuds libres, 562 répertoires
 Blocs libres : 2560494-2568191, 2568289-2572399
 I-noeuds libres : 640320-643376
```

Pour cette partition, **79 groupes** (de 0 à 78) ont été créés. On distingue trois types de groupe :

- 9 groupes complets avec le superbloc et la GDT avec 8 copies de sauvegarde (taille : 32768 blocs)
- 69 groupes complets (taille du groupe : 32768 blocs)
- 1 groupe (le dernier) incomplet (taille : 16496 blocs)

9 groupes	*Taille en octets*
superbloc	1
Descr. groupes	1
GDT	628
Bimap bloc	1
Bitmap inode	1
Table inode	509
Blocs	31627
1 groupe =	32768
9 x 1141 =	10269
9 x 31627 =	284643

69 groupes	*Taille en octets*
Bimap bloc	1
Bitmap inode	1
Table inode	509
blocs	32257
1 groupe =	32768
69 x 511 =	35259
69 x 32257 =	2225733

1 groupe	*Taille en octets*
Bimap bloc	1
Bitmap inode	1
Table inode	509
blocs	15985
1 groupe =	16496
1 x 511 =	511
1 x 15985 =	15985

Total blocs : (78 x 32768) + (1 x 16496) = **2 572 400** soit 10 536 550 400 octets (9,81 GiO)
Total blocs réservés (*file system*) : 10269 + 35259 + 511 = 46 039 soit 188 575 744 octets (179,8 moi) = 1,79%
Total blocs libres (*file*) : 284643 + 2225733 + 15985 = 2 526 631 soit 10 349 080 576 octets (9,6 GiO)

Le nombre de groupes de blocs ?
Réponse : **Inode count / Inodes per group** = 643 376 / 8 144 = **79** groupes

Le nombre de blocs de la table des inodes ?
Réponse : (**Inodes per group** x **Inode size) / Block size**
= (8 144 x 256) / 4 096 = **509** *blocs*

Le nombre total de fichiers possibles ?
Réponse : **Inode count - First inode =** 643 376 – 11 = **643 365** fichiers

Le nombre de blocs par groupe ?
Réponse : (**Blocks per group**) : Taille de la table d'allocation des blocs (= 1 bloc) x 8 (bits)
= 4096 octets x 8 = 32 768 bits soit **32 768** blocs par groupe

Le nombre d'inodes (fichiers) par groupe ?
Réponse : **Inode count / Nb de groupes =** 643 376 / 79 = **8 144** inodes

Le nombre de blocs du dernier groupe ?
Réponse : 2 572 400 % 32768 = **16 496 blocs**

La taille de ce système de fichiers ?
Réponse : **Block count x Block size** = 2 572 400 x 4096 = 10 536 550 400 octets (9,81 GiO)

Affichage avec df :

```
# df --block-size=4096 /dev/sdc1
Sys. de fich.         4K-blocs    Occupé  Disponible Capacité Monté sur
/dev/sdc1              2532013   1395649     1007744      59%      /usr
```

En fait, cet affichage peut être faux car il ne tient pas compte des *Reserved GDT blocks* :

- 9 groupes complets avec le superbloc (sans la GDT) avec 8 copies de sauvegarde (soit 31627+628 = 32255 blocs pour les fichiers)
- 69 groupes complets (soit 32257 blocs pour les fichiers)
- 1 groupe (le dernier) incomplet (soit 15985 blocs pour les fichiers)

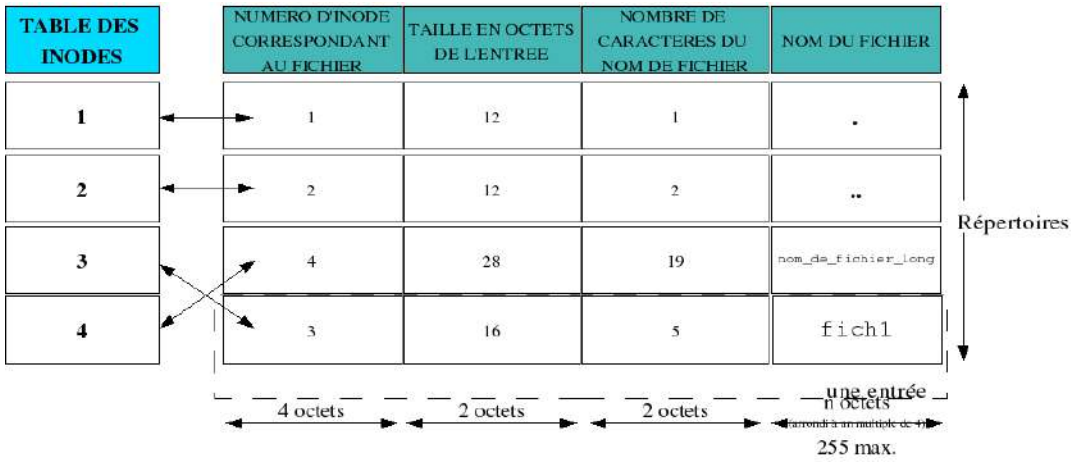
Ce qui fait un total de : (9 x 32 255) + (69 x 32 257) + (1 x 15985) = 2 532 013 blocs

En fait, il devrait afficher 2 526 361 blocs soit un différence de 5 652 blocs (soit 9 x 628).

Structure des entrées de répertoire

Un système de fichiers est organisé en fichiers et en répertoires.

Un **répertoire** est un fichier de type particulier, composés de blocs de données structurés en une suite d'**entrées** :



Remarque : la version **ext3** a apporté une légère modification aux entrées de répertoire. Dans /usr/src/linux/include/linux/ext3_fs.h :

```
/*
 * Structure of a directory entry
 */
#define EXT3_NAME_LEN 255

struct ext3_dir_entry_2 {
        __le32  inode;                  /* Inode number */
        __le16  rec_len;                /* Directory entry length */
        __u8    name_len;               /* Name length */
        __u8    file_type;
        char    name[EXT3_NAME_LEN];    /* File name */
};

/* soit 4 + 2 + 1 + 1 + 255 max octets */

/*
 * Ext3 directory file types.  Only the low 3 bits are used.  The
 * other bits are reserved for now.
 */
#define EXT3_FT_UNKNOWN         0
#define EXT3_FT_REG_FILE        1
#define EXT3_FT_DIR             2
#define EXT3_FT_CHRDEV          3
#define EXT3_FT_BLKDEV          4
#define EXT3_FT_FIFO            5
#define EXT3_FT_SOCK            6
#define EXT3_FT_SYMLINK         7
```

inode

Le terme ***inode*** désigne le descripteur d'un fichier. Les inodes (contraction de « index » et « node », en français : nœud d'index) sont des **structures de données** contenant des informations concernant les fichiers stockés dans certains systèmes de fichiers (notamment de type Linux/Unix). À chaque fichier correspond un numéro d'inode (*i-number*) dans le système de fichiers dans lequel il réside, unique au périphérique sur lequel il est situé.

Un inode contient :

- les attributs du fichier (affichés notamment pas un **ls -l** ou la commande **stat**) et
- une table d'accès aux blocs de données.

Le numéro d'inode est un **entier unique** pour le périphérique dans lequel il est stocké. Tous les fichiers, y compris les fichiers spéciaux, sont identifiés par un inode. Le numéro d'inode d'un fichier peut être affiché avec la commande :

```
$ ls -i un_fichier
8131 un_fichier
```

Le standard POSIX s'est basé sur les systèmes de fichiers traditionnels d'Unix. Cette norme impose donc que les fichiers réguliers aient les attributs suivants (« fiche d'identité » d'un fichier identifié par un numéro d'inode) :

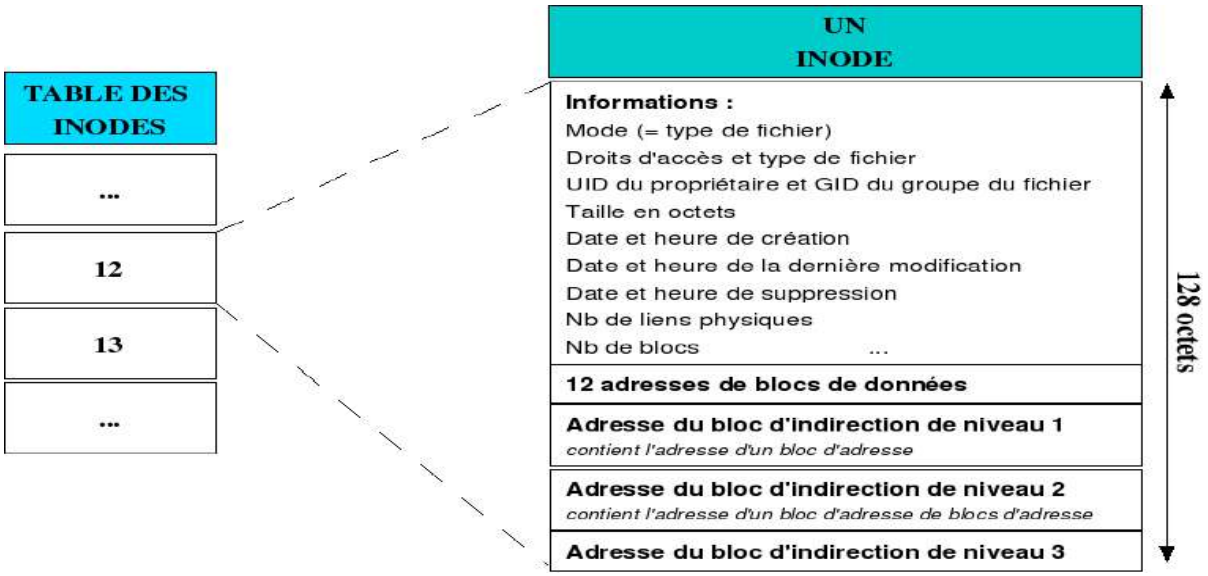
- La taille du fichier en octets
- Identifiant du périphérique contenant le fichier
- L'identifiant du propriétaire du fichier
- L'identifiant du groupe auquel appartient le fichier
- Le numéro d'inode qui identifie le fichier dans le système de fichier
- Le mode du fichier qui détermine quel utilisateur peut lire, écrire et exécuter ce fichier
- Horodatage (*timestamp*) pour :
  - La date de dernière modification *ctime* de l'inode (affichée par la commande stat ou par ls -lc)
  - La date de dernière modification du fichier *mtime* (affichée par le classique ls -l)
  - La date de dernier accès *atime* (affichée par la commande stat ou par ls -lu)
- Un compteur indiquant le nombre de liens physiques sur cet inode.

Les temps sont exprimés en secondes depuis le premier janvier 1970 à 0 h GMT

Remarque : les inodes ne contiennent pas les noms de fichier.

Structure d'un inode

Un inode occupera 128 ou 256 octets (taille définie à la création du système de fichiers).



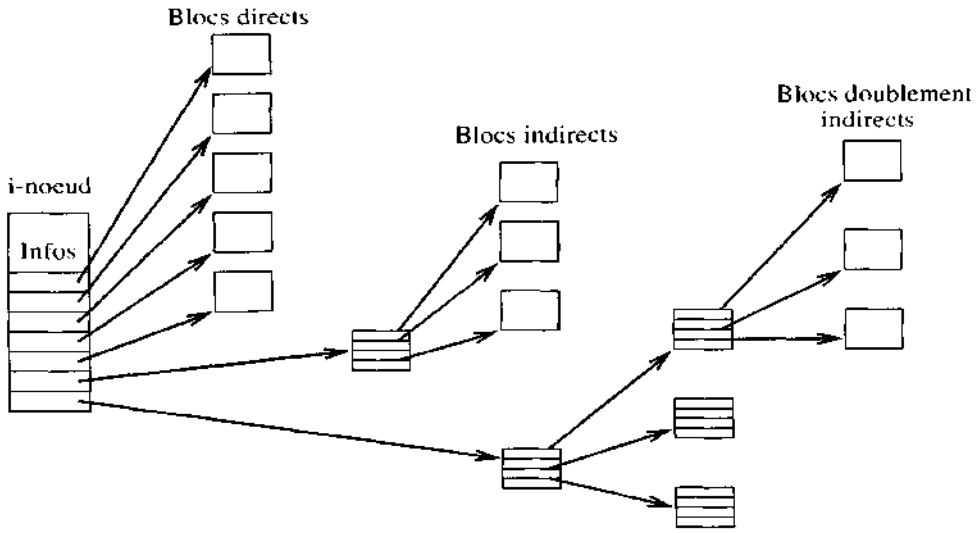
Mécanisme d'adressage des blocs

Par défaut, un bloc a une taille de **4096 octets** (4 KiO). Une adresse de bloc occupe **4 octets** (32 bits).

L'inode contient 15 adresses de blocs :

- les 12 premières sont les numéros des 12 premiers blocs de données du fichier soit 12 x 4 KO = **48 KO** au total ;
- la suivante indique le numéro d'un bloc indirection de niveau 1 : il adresse donc un bloc de 4 KO contenant (4096/4) 1024 adresses de blocs. On a donc maintenant : (48 KO) + (1024 x 4 KO) = **4 MO**.
- l'adresse du bloc indirection de niveau 2 : cette adresse pointe sur 1024 adresses de blocs d'indirection de niveau 1, donc (48 KO) + (1024 x 4 KO) + (1024 x 1024 x 4 KO) = **4 GO**.
- l'adresse du bloc indirection de niveau 3 : suivant le même principe, on a (48 KO) + (1024 x 4 KO) + (1024 x 1024 x 4 KO) + (1024 x 1024 x 1024 x 4 KO) = **4 TO**.

Le principe d'adressage, ici limité à 2 niveaux d'indirection pour des raisons de clarté :



Fragmentation

Un grand mythe à propos de ce système de fichier est qu'il ne fragmenterait pas (contrairement à FAT ou NTFS utilisés par Windows), sous prétexte qu'il rangerait ses données de manière « intelligente ». Il limite en effet la fragmentation (tout comme NTFS et la plupart des systèmes de fichiers récents) mais ne l'empêche en aucun cas. Celle-ci n'est pas toujours évitable, par exemple dans le cas où l'espace libre contigu n'est pas assez grand pour créer un fichier d'une taille donnée. Le fichier à créer est alors réparti dans les espaces libres disponibles, fragmentant ce fichier.

La fragmentation arrive le plus souvent sur des disques très plein (moins de 20% d'espace libre) en manipulant de gros fichiers (par exemple des images ISO).

Officiellement, il n'existe pas d'utilitaire de défragmentation qui travaille au niveau des bits sur une ext3. Des utilitaires de défragmentation existent mais requièrent un démontage puis une conversion vers une ext2. Ces utilitaires sont à utiliser avec une extrême prudence car ils peuvent détruire des données.

Un utilitaire de défragmentation travaille au niveau des bits : e2defrag et deux au niveau de la structure : Shake et defrag. Sinon, la seule méthode pour défragmenter des fichiers situés sur un disque ext2/ext3 consiste à les copier sur un autre support, effacer les fichiers d'origine puis recopier les fichiers à leur emplacement d'origine.

Ceci étant dit, la fragmentation reste généralement très faible sous Linux de part la méthode de fonctionnement d'ext2/ext3. Dans la grande majorité des cas, ext3 conserve une fragmentation inférieure à 3%.

La commande fsck (l'option -n est nécessaire pour interdire toute réparation) donnera en fin de rapport le % de fragmentation ("% non-contiguous") :

```
# fsck -n <partition>
```

Voir aussi : la commande **filefrag**

La journalisation ext3

La journalisation consiste à écrire sur le disque toutes les opérations en cours à chaque instant. Ainsi, lorsqu'un redémarrage intempestif se produit, le système peut ramener rapidement la structure de données du système de fichiers dans un état cohérent. La journalisation accroît donc encore à la fiabilité du système de fichiers.

Plus tard, une extension a été ajoutée, permettant de journaliser le système de fichiers ext2. Celle-ci a été nommée ext3. Un système qui ne connaît que l'ext2 est parfaitement capable de lire et d'écrire de l'ext3, mais il n'y aura pas alors de journalisation. La différence entre les deux systèmes réside dans l'adjonction d'une zone journal et la suppression des données, rendant la récupération de celles-ci impossible sur le système de fichier ext3. Il suffit de cocher une option dans son noyau et de le recompiler pour bénéficier du support de ext3.

Pour pouvoir passer sa partition ext2 en ext3, cela se fait à l'aide de la commande tune2fs (exemple de commande : tune2fs -j /dev/hda7).

Un élément important de sécurité d'ext2fs est la commande « **chattr** » qui lui est associée, et qui peut servir de garde-fou léger contre l'effacement accidentel même par root, **non possible en ext3**.

Dans un système de fichiers ext2, il et possible d'utiliser les attributs ext2 dans le but de protéger ses données. Ces attributs sont manipulés à l'aide de la commande **chattr**. Il y a un attribut « ajout seulement » (*append-only*) : il est possible d'ajouter des données à un fichier ayant cet attribut, mais pas de le supprimer, et le contenu du fichier ne peut pas être écrasé. Si un répertoire a cet attribut, tous les fichiers et répertoires qu'il contient peuvent être

normalement modifiés, mais aucun fichier ne peut être supprimé. Cet attribut peut être placé en tapant

```
$ chattr +a FICHIER...
```

Il existe aussi un attribut « immuable » (*immutable*), qui ne peut être placé ou retiré qu'en tant que root. Un fichier ou répertoire ayant cet attribut ne peut être ni modifié, ni supprimé, ni renommé, ni se faire ajouter un lien (physique). Il peut être placé comme suit :

```
# chattr +i FICHIER...
```

Ext2fs fournit également l'attribut « récupérable » (*undeletable*, option +u de chattr). Si un fichier ayant cet attribut est supprimé, mais pas réellemnt réutilisé, il est déplacé vers un « endroit sûr » afin d'être supprimé plus tard.

Modes de journalisation
Ext3 dispose de trois modes de journalisation: "data=writeback"; "data=ordered"; et "data=journal". Ces modes sont activables au montage de la partition. Par défaut, ext3 est monté avec le mode "data=ordered" qui convient pour le cas général. Dans ce mode, les métadonnées sont d'abord écrites dans le journal, ensuite sur le disque après l'écriture effective des données sur le disque. Le mode writeback, où les métadonnées sont journalisées mais peuvent être écrites sur le disque avant les données elles-mêmes, est plus rapide dans certaines circonstances, tout en assurant la cohérence du système de fichiers (c'est une garantie supplémentaire par rapport à ext2). C'est aussi le mode de fonctionnement de reiserfs, jfs et xfs. Enfin, le mode "data=journal", où les données et les métadonnées sont journalisées, est plus lent car les données sont écrites deux fois, mais offre aussi plus de garantie quant à l'intégrité des fichiers. Ce mode est, semble-t-il, plus adapté aux bases de données.

C'est l'endroit où sont stockés les logs d'activité de la journalisation. Si le journal a été créé par la commande tune2fs -j sur une partition démontée, ou par mke2fs -j sur une partition non formatée, alors il se matérialise sous la forme d'un inode invisible par le système de fichiers. Sinon, c'est un fichier caché ".journal" à la racine du système de fichiers, avec attribut "immuable" (man chattr). On peut imaginer ce fichier comme un fichier swap dédié à la journalisation. Son contenu change en permanence en fonction des activités du système de fichiers (création, modification, suppression de fichiers), mais le fichier lui-même garde la même taille et la même date de création/modification.

Quelques commandes utiles

mkfs et **fsck** : crée et vérifie un FS (commandes standards UNIX)
mount et **umount** : monte et démonte un FS
df : indique l'espace libre des disques et partitions
du : indique l'espace occupé par une arborescence
mke2fs, **e2fsck**, **tune2fs** : crée, vérifie et paramètre un FS ext2
dumpe2fs : infos sur le super bloc et les groupes de bloc
debugfs : débogue un FS ext2
lsattr et **chattr** : lister et changer les attributs des fichiers

Affichage de l'espace de chaque système de fichiers :

```
# df -Th
SysFichier  Type        Tail.       Util.       Disp.       Uti%        Monté sur
/dev/hda1   ext2        2.9G        469M        2.2G        17%         /
/dev/hda6   ext2        22G         1.4G        19G         7%          /home
/dev/hdd1   vfat        19G         1.4G        17G         8%          /mnt/windows
/dev/hda5   ext2        12G         4.2G        6.9G        38%         /usr
```

Affichage du *superbloc* :

```
# tune2fs -l /dev/hda1
tune2fs 1.26 (3-Feb-2002)
Filesystem UUID:          96d32da4-8ceb-11d6-94f5-9aef08a756f7
...
Filesystem OS type:       Linux
Inode count:              381696
Block count:              763079
Reserved block count:     38153
Free blocks:              631037
Free inodes:              352172
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         15904
Inode blocks per group:   497
Last mount time:          Sun May  4 18:31:57 2003
Last write time:          Sun May  4 19:26:26 2003
Mount count:              10
Maximum mount count:      30
Last checked:             Sun Apr  6 12:48:33 2003
Check interval:           15552000 (6 months)
Next check after:         Fri Oct  3 12:48:33 2003
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
```

Affichage des informations sur le *superbloc* et les groupes de blocs :

```
# dumpe2fs /dev/hda1
dumpe2fs 1.26 (3-Feb-2002)
...
Group 0: (Blocks 0-32767)
Primary Superblock at 0,  Group Descriptors at 1-1
Block bitmap at 2 (+2), Inode bitmap at 3 (+3)
Inode table at 4-500 (+4)
23736 free blocks, 15872 free inodes, 8 directories
Free blocks: 506-509, ...
Free inodes: 29, 31, 35-15904

Group 1: (Blocks 32768-65535)
Backup Superblock at 32768,  Group Descriptors at 32769-32769
Block bitmap at 32770 (+2), Inode bitmap at 32771 (+3)
Inode table at 32772-33268 (+4)
29292 free blocks, 15182 free inodes, 213 directories
Free blocks: 34416-34418, ...
Free inodes: 16553, ...
...

Group 23: (Blocks 753664-763078)
Block bitmap at 753664 (+0), Inode bitmap at 753665 (+1)
Inode table at 753668-754164 (+4)
8916 free blocks, 15904 free inodes, 0 directories
Free blocks: 753666-753667, 754165-763078
Free inodes: 365793-381696
```

Affiche les informations de base sur un fichier :

```
# ls -il un_fichier
8131 -rw-r--r-- 1 root root 19 2009-10-28 11:26 un_fichier
```

Affiche les informations contenues dans un inode :

```
# stat un_fichier
  File: `un_fichier'
  Size: 19              Blocks: 8          IO Block: 4096   fichier régulier
Device: 815h/2069d      Inode: 8131        Links: 1
Access: (0644/-rw-r--r--)  Uid: (    0/    root)   Gid: (    0/    root)
Access: 2009-10-28 11:26:30.000000000 +0100
Modify: 2009-10-28 11:26:41.000000000 +0100
Change: 2009-10-28 11:26:41.000000000 +0100
```

Affiche le périphérique, la partition et le type de système de fichiers d'un fichier :

```
# df -T un_fichier
Sys. de fich. Type     Tail. Occ. Disp. %Occ. Monté sur
/dev/sdb5     ext3    7,4G  662M  6,3G  10% /
```

Affiche les informations complètes contenues dans un inode :

```
# echo "stat <8131>" | debugfs /dev/sdb5
debugfs 1.41.4 (27-Jan-2009)
debugfs:  Inode: 8131   Type: regular    Mode:  0644   Flags: 0x0
Generation: 944415664    Version: 0x00000000
User:     0   Group:     0   Size: 19
File ACL: 0    Directory ACL: 0
Links: 1   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x4ae81c61 -- Wed Oct 28 11:26:41 2009
atime: 0x4ae81c56 -- Wed Oct 28 11:26:30 2009
mtime: 0x4ae81c61 -- Wed Oct 28 11:26:41 2009
Size of extra inode fields: 4
BLOCKS:
(0):57360
TOTAL: 1
```

Affiche (en hexa et en ASCII) les données contenues dans un bloc :

```
# dd if=/dev/sdb5 bs=4096 skip=57360 count=1 | hexdump -C
1+0 enregistrements lus
1+0 enregistrements écrits
4096 octets (4,1 kB) copiés, 0,0127792 s, 321 kB/s
00000000  4a 65 20 73 75 69 73 20  75 6e 20 66 69 63 68 69  |Je suis un fichi|
00000010  65 72 0a 00 00 00 00 00  00 00 00 00 00 00 00 00  |er..............|
```

Affiche en ASCII les données contenues dans un fichier :

```
# cat un_fichier
Je suis un fichier
```

Affiche (en hexa et en ASCII) les données contenues dans un fichier :

```
# hexdump -C un_fichier
00000000  4a 65 20 73 75 69 73 20  75 6e 20 66 69 63 68 69  |Je suis un fichi|
00000010  65 72 0a                                          |er.|
```

Affiche le numéro d'inode d'un répertoire :

```
# ls -id /tmp
8113 /tmp/
```

Le numéro d'inode est 8113 (0x1FB1).

Affiche les informations sur la partition contenant le répertoire spécifié :

```
# df -Th /tmp
Sys. de fich. Type     Tail. Occ. Disp. %Occ. Monté sur
/dev/sdb5     ext3    7,4G  664M  6,3G  10% /
```

Affiche les informations partielles du groupe 0 d'une partition :

```
# dumpe2fs /dev/sdb5 | grep -A 4 "Groupe 0"
dumpe2fs 1.41.4 (27-Jan-2009)
Groupe 0 : (Blocs 0-32767)
 superbloc Primaire à 0, Descripteurs de groupes à 1-1
  Blocs réservés GDT à 2-476
 Bitmap de blocs à 477 (+477), Bitmap d'i-noeuds à 478 (+478)
 Table d'i-noeuds à 479-985 (+479)
```

Affiche l'entrée de répertoire de la racine d'une partition :

```
# dd if=/dev/sdb5 bs=4096 skip=986 count=1 | hexdump -C
00000000  02 00 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |................|
00000010  0c 00 02 02 2e 2e 00 00  0b 00 00 00 14 00 0a 02  |................|
00000020  6c 6f 73 74 2b 66 6f 75  6e 64 00 00 91 57 03 00  |lost+found...W..|
00000030  0c 00 03 02 6d 6e 74 00  f1 96 03 00 0c 00 03 02  |....mnt.........|
00000040  64 65 76 00 21 b9 02 00  0c 00 03 02 65 74 63 00  |dev.!.......etc.|
00000050  01 fb 01 00 0c 00 04 02  68 6f 6d 65 b1 1f 00 00  |........home....|
00000060  0c 00 03 02 74 6d 70 00  f1 9b 01 00 0c 00 03 02  |....tmp.........|
...
```

Affiche les informations contenues dans l'inode du répertoire tmp :

```
# echo "stat <8113>" | debugfs /dev/sdb5
debugfs 1.41.4 (27-Jan-2009)
debugfs:  Inode: 8113   Type: directory    Mode:  0777   Flags: 0x0
Generation: 2329885321    Version: 0x00000000
User:     0   Group:     0   Size: 4096
File ACL: 0    Directory ACL: 0
Links: 10   Blockcount: 8
Fragment:  Address: 0    Number: 0    Size: 0
ctime: 0x4aeebe2a -- Mon Nov  2 12:10:34 2009
atime: 0x4aeeba4a -- Mon Nov  2 11:54:02 2009
mtime: 0x4aeebe2a -- Mon Nov  2 12:10:34 2009
Size of extra inode fields: 4
BLOCKS:
(0):36864
TOTAL: 1
```

Affiche l'entrée de répertoire du répertoire tmp :

```
# dd if=/dev/sdb5 bs=4096 skip=36864 count=1 | hexdump -C
00000000  b1 1f 00 00 0c 00 01 02  2e 00 00 00 02 00 00 00  |................|
00000010  0c 00 02 02 2e 2e 00 00  b2 1f 00 00 14 00 09 02  |................|
00000020  2e 49 43 45 2d 75 6e 69  78 00 00 00 b4 1f 00 00  |.ICE-unix.......|
00000030  14 00 09 02 2e 58 31 31  2d 75 6e 69 78 7a 73 35  |.....X11-unixzs5|
00000040  b3 1f 00 00 10 00 08 01  2e 58 30 2d 6c 6f 63 6b  |.........X0-lock|
00000050  b6 1f 00 00 14 00 08 02  2e 65 73 64 2d 35 30 30  |.........esd-500|
```

```
00000060  7a 69 32 6b ba 1f 00 00  14 00 0a 02 2e 75 72 70  |zi2k.........urp|
00000070  6d 69 2d 35 30 30 30 39  c7 1f 00 00 10 00 08 02  |mi-50009........|
00000080  6b 64 65 2d 72 6f 6f 74  c3 1f 00 00 14 00 0a 01  |kde-root........|
00000090  75 6e 5f 66 69 63 68 69  65 72 31 74 bc 1f 00 00  |un_fichier1t....|
000000a0  14 00 06 02 78 73 74 72  61 31 47 34 43 74 53 30  |....xstra1G4CtS0|
000000b0  bd 1f 00 00 14 00 09 02  2e 77 69 6e 65 2d 35 30  |.........wine-50|
000000c0  30 79 54 79 b9 1f 00 00  0c 00 04 02 70 65 61 72  |0yTy........pear|
000000d0  c2 1f 00 00 48 00 3d 06  4f 53 4c 5f 50 49 50 45  |....H.=.OSL_PIPE|
000000e0  5f 35 30 30 5f 53 69 6e  67 6c 65 4f 66 66 69 63  |_500_SingleOffic|
000000f0  65 49 50 43 5f 66 30 32  32 34 64 62 63 39 32 38  |eIPC_f0224dbc928|
00000100  64 65 36 34 35 35 39 32  31 32 37 66 63 31 34 38  |de645592127fc148|
00000110  34 62 35 66 64 6c 61 73  c4 1f 00 00 20 00 16 07  |4b5fdlas.... ...|
00000120  75 6e 5f 6c 69 65 6e 5f  73 75 72 5f 75 6e 5f 66  |un_lien_sur_un_f|
00000130  69 63 68 69 65 72 10 01  c3 1f 00 00 28 00 1f 01  |ichier......(...|
00000140  75 6e 5f 6c 69 65 6e 5f  70 68 79 73 69 71 75 65  |un_lien_physique|
00000150  5f 73 75 72 5f 75 6e 5f  66 69 63 68 69 65 72 00  |_sur_un_fichier.|
00000160  c5 1f 00 00 24 00 1c 07  75 6e 5f 61 75 74 72 65  |....$...un_autre|
00000170  5f 6c 69 65 6e 5f 73 75  72 5f 75 6e 5f 66 69 63  |_lien_sur_un_fic|
00000180  68 69 65 72 c6 1f 00 00  7c 0e 07 01 73 64 61 2e  |hier....|...sda.|
00000190  6f 75 74 62 c8 1f 00 00  6c 0e 0a 01 6d 61 6e 2e  |outb....l...man.|
000001a0  4a 4f 62 45 7a 62 73 2e  50 4c 79 69 65 37 69 63  |JObEzbs.PLyie7ic|
000001b0  c9 1f 00 00 50 0e 0f 01  62 72 5f 69 6e 70 75 74  |....P...br_input|
000001c0  2e 4d 71 68 4e 33 57 01  ca 1f 00 00 38 0e 0f 01  |.MqhN3W.....8...|
000001d0  62 72 5f 69 6e 70 75 74  2e 48 7a 71 62 6a 45 74  |br_input.HzqbjEt|
000001e0  2e 48 4e 6a 34 61 54 73  cf 1f 00 00 28 00 0b 01  |.HNj4aTs....(...|
000001f0  69 6e 69 74 2e 66 57 65  42 74 38 67 d0 1f 00 00  |init.fWeBt8g....|
00000200  14 00 0b 01 69 6e 69 74  2e 46 72 4c 77 62 57 4a  |....init.FrLwbWJ|
00000210  d1 1f 00 00 64 00 0b 01  69 6e 69 74 2e 6e 39 31  |....d...init.n91|
00000220  71 55 4a 44 d2 1f 00 00  50 00 0b 01 69 6e 69 74  |qUJD....P...init|
00000230  2e 6a 57 37 73 45 78 34  d3 1f 00 00 14 00 0b 01  |.jW7sEx4........|
00000240  69 6e 69 74 2e 64 77 64  44 70 6c 65 d4 1f 00 00  |init.dwdDple....|
00000250  14 00 0b 01 69 6e 69 74  2e 48 41 42 66 63 39 59  |....init.HABfc9Y|
00000260  d5 1f 00 00 14 00 0b 01  69 6e 69 74 2e 54 71 59  |........init.TqY|
00000270  32 5a 57 38 d6 1f 00 00  8c 0d 0b 01 69 6e 69 74  |2ZW8........init|
00000280  2e 72 75 69 57 4f 4b 6e  d7 1f 00 00 78 0d 0b 01  |.ruiWOKn....x...|
00000290  69 6e 69 74 2e 74 52 32  58 45 79 58 6f 45 48 74  |init.tR2XEyXoEHt|
000002a0  d8 1f 00 00 60 0d 0b 01  69 6e 69 74 2e 54 73 46  |....`...init.TsF|
000002b0  6f 4f 67 74 2e 68 4a 53  33 34 31 44 d9 1f 00 00  |oOgt.hJS341D....|
000002c0  44 0d 0b 01 69 6e 69 74  2e 48 38 6f 31 4b 35 74  |D...init.H8o1K5t|
000002d0  da 1f 00 00 30 0d 0b 01  69 6e 69 74 2e 56 42 42  |....0...init.VBB|
000002e0  34 72 39 00 00 00 00 00  00 00 00 00 00 00 00 00  |4r9.............|
000002f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
```

Affiche le contenu du répertoire tmp :

```
# ls -ail /tmp
8113 drwxrwxrwt 10 root root 4096 2009-11-02 12:10 ./
   2 drwxr-xr-x 20 root adm  4096 2009-11-02 11:56 ../
8118 drwx------  2 tv   tv   4096 2009-10-31 16:45 .esd-500/
8114 drwxrwxrwt  2 root root 4096 2009-10-25 17:07 .ICE-unix/
8135 drwx------  2 root root 4096 2009-10-25 17:07 kde-root/
8130 srwxrwxr-x  1 tv   tv      0 2009-11-02 08:17
OSL_PIPE_500_SingleOfficeIPC_f0224dbc928de645592127fc1484b5fd=
8121 drwxr-xr-x  4 root root 4096 2009-09-08 09:25 pear/
8134 -rw-r--r--  1 root root  416 2009-10-30 18:16 sda.out
8133 lrwxrwxrwx  1 root root   81 2009-10-29 11:23 un_autre_lien_sur_un_fichier ->
../usr/../usr/local/../local/lib/../lib/xorg/../xorg/modules/drivers/intel_drv.la*
8131 -rw-rw-r--  2 tv   tv     19 2009-10-28 11:26 un_fichier
```

```
8131 -rw-rw-r--  2 tv   tv     19 2009-10-28 11:26 un_lien_physique_sur_un_fichier
8132 lrwxrwxrwx  1 tv   tv     10 2009-10-28 11:50 un_lien_sur_un_fichier -> un_fichier
8122 drwxr-xr-x  4 tv   tv   4096 2009-07-18 22:41 .urpmi-500/
8125 drwx------  4 tv   tv   4096 2009-07-20 13:47 .wine-500/
8115 -r--r--r--  1 root root   11 2009-10-25 17:07 .X0-lock
8116 drwxrwxrwt  2 root root 4096 2009-10-25 17:07 .X11-unix/
8124 drwx------  2 tv   tv   4096 2009-10-28 17:39 xstra1/
```

Copier le MBR dans un fichier :

```
# dd if=/dev/sda of=/root/MBR.image bs=512 count=1
```

Afficher la table des partitions contenue dans le MBR d'un disque :

```
# hexdump -s 0x1BE -C /root/MBR.image
000001be  80 01 01 00 83 fe ff ff  3f 00 00 00 a7 c5 52 01
000001ce  00 fe ff ff 05 fe ff ff  e6 c5 52 01 9b 7f c9 1b
000001de  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001ee  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
000001fe  55 aa
```

ou : `# cfdisk -P r /dev/sda`

Le décodage de la table des partitions contenues dans le MBR peut se faire manuellement ou par un programme :

```
# cfdisk -P t /dev/sda
Table de partitions de /dev/sda

          ---Starting----      ----Ending-----    Start     Number of
 # Flags Head Sect  Cyl   ID  Head Sect  Cyl     Sector    Sectors
-- ----- ---- ---- ----- ---- ---- ---- ----- ----------- -----------
 1  0x80    1    1     0 0x83  254   63  1381          63    22201767
 2  0x00    0    1  1382 0x05  254   63 30400    22201830   466190235
 3  0x00    0    0     0 0x00    0    0     0           0           0
 4  0x00    0    0     0 0x00    0    0     0           0           0
 5  0x00    1    1  1382 0x82  254   63  1584          63     3261132
 6  0x00    1    1  1585 0x83  254   63  3588          63    32194197
 7  0x00    1    1  3589 0x83  254   63 30400          63   430734717
```

Lister la la table des partitions contenue dans le MBR d'un disque :

```
# fdisk -l -u /dev/sda

Disque /dev/sda: 250.0 Go, 250059350016 octets
255 heads, 63 sectors/track, 30401 cylinders, total 488397168 secteurs
Units = secteurs of 1 * 512 = 512 bytes
Disk identifier: 0x12b7c5c2

Périphérique Amorce    Début         Fin      Blocs    Id  Système
/dev/sda1   *          63    22201829    11100883+  83  Linux
/dev/sda2        22201830   488392064   233095117+   5  Extended
/dev/sda5        22201893    25463024     1630566   82  Linux swap / Solaris
/dev/sda6        25463088    57657284    16097098+  83  Linux
/dev/sda7        57657348   488392064   215367358+  83  Linux
```

```
# cfdisk -P s /dev/sda
Table de partitions de /dev/sda

               Premier     Dernier
 # Type       Secteur     Secteur  Offset    Longueur Sys.FichierType (ID) Fanions
-- ------- ----------- ----------- ------ ----------- -------------------- ----
 1 Primair           0    22201829     63    22201830 Linux (83)           Amorce
 2 Primair    22201830   488392064      0   466190235 Extended (05)        Aucun
 5 Logique    22201830    25463024     63     3261195 Linux swap / So (82) Aucun
 6 Logique    25463025    57657284     63    32194260 Linux (83)           Aucun
 7 Logique    57657285   488392064     63   430734780 Linux (83)           Aucun
```

Afficher le nombre d'inode utilisé et disponible pour une partition :

```
# df -iH /
Sys. de fich.          Inodes   IUtil.  ILib. %IUti. Monté sur
/dev/sdb5                487k     42k    446k    9% /
```

Le système de fichiers Microsoft FAT (File Allocation Table)

FAT, de *file allocation table* (table d'allocation de fichiers), est un système de fichiers relativement basique. Il a été conçu par Bill Gates et Marc McDonald pour le Microsoft Disk BASIC , puis réutilisé dans QDOS dont c'est le seul élément qui ne soit pas inspiré par CPM (QDOS est l'ancêtre de MS-DOS, écrit par Tim Paterson). Il fut ensuite utilisé sous MS-DOS puis dans la branche 9x de Windows. La branche NT utilisera elle le NTFS.

FAT a été conçu au départ pour des disquettes de 160 Ko, mais un de ses plus gros défauts était le nombre limité de caractères dans les noms des fichiers (le fameux *8.3*) : 1 à 8 caractères, un point et 0 à 3 caractères, sans distinction de la casse. Ce défaut a été résolu dans MS-DOS 7 (le MS-DOS de Windows 95) grâce à une astuce préservant la compatibilité : le système VFAT. Mais il ne permet l'accès aux noms longs qu'à l'intérieur de Windows 95. Les versions ultérieures de MS-DOS n'ont pas corrigé ce problème en raison de la prépondérance de Windows. VFAT permet aussi d'utiliser les lettres minuscules et les caractères unicode dans les noms de fichiers.

FAT a évolué en différentes versions, toutes supportées par les dernières versions de MS-DOS et les versions actuelles de Windows, qui ont été rendues nécessaires par l'évolution des capacités de stockage des disques :

- FAT12 : maximum de 2^{12}, soit 4 096 clusters de taille fixe (choisie au départ entre 512 octets et 4 Ko). Utilisé entre autres sur les disquettes.
- FAT16 : maximum de 2^{16}, soit 65 536 clusters de taille fixe (choisie au départ entre 2 Ko et 32 Ko).
- VFAT : Une évolution de la FAT, permettant de gérer les noms longs dans Windows 95 et les versions suivantes. Il s'applique à toutes les versions de FAT (FAT12, FAT16, FAT32...).
- FAT32 : apparu avec Windows 95 OSR2, il supporte un maximum de 2^{28} (268 millions !) de clusters de taille variable de 4 Ko à 32 Ko, avec reprise du système VFAT. Notons que la taille des fichiers ne peut dépasser 4 Go.

Malgré son manque de résistances aux pannes par rapport à ses concurrents, FAT reste aujourd'hui (2005) très utilisé, notamment sur les cartes mémoires pour appareils photo numériques en raison de la simplicité de son implémentation permettant l'utilisation dans des systèmes embarqués ainsi que la compatibilité assurée avec Windows et beaucoup d'autres systèmes d'exploitation.

Microsoft a tenté en décembre 2003 de déposer un brevet sur FAT, de façon à pouvoir percevoir des droits sur les licences qui auraient été accordées aux fabricants d'appareils électroniques. Mais la demande a été rejetée en septembre 2004. Elle a cependant été validée en janvier 2006, après que Microsoft eut complété son argumentaire.

Remarque :
Un ***cluster*** est un groupe de secteurs. Il sert d'unité d'allocation aux fichiers. Chaque cluster stocke donc les données d'un fichier. Pour un fichier de 9000 octets par exemple, sur un disque utilisant des clusters de 8192 octets (soit 16 secteurs de 512 octets), 2 clusters sont utilisés, dont le dernier n'utilise que 808 octets (9000 - 8192).

Aspect technique

Le système de fichier FAT est composé de 3 grandes sections :

1. Le "secteur de *boot*" ou "BPB" : c'est le premier secteur du disque.
2. Les "tables d'allocation" ou "FATs" : c'est une *carte* du disque.
3. Le "répertoire racine" ou "Root directory" : c'est une liste des fichiers présents à la *racine* du disque.

Le "secteur de *boot*" ou "BPB"

Le secteur de *boot* a le format suivant :

Position (octets)	**Taille (octets)**	**Description**
0	3	Saut vers un programme qui va charger le système d'exploitation
3	8	Nom du programme qui a formaté le disque ("MSWIN4.1" par exemple).
11	2	Nombre d'octets par secteur (512, 1024, 2048 ou 4096).
13	1	Nombre de secteurs par cluster (1, 2, 4, 8, 16, 32, 64 ou 128).
14	2	Nombre de secteur réservé en comptant le secteur de boot (32 par défaut pour FAT32, 1 par défaut pour FAT12/16).
16	1	Nombre de FATs sur le disque (2 par défaut)
17	2	Taille du répertoire racine (0 par défaut pour FAT32).
19	2	Nombre total de secteur 16-bit (0 par défaut pour FAT32).
21	1	Type de disque (0xF8 pour les disques durs, 0xF0 pour les disquettes).
22	2	Taille d'une FAT en secteurs (0 par défaut pour FAT32).
24	2	Nombre de secteurs par piste.
26	2	Nombre de têtes.
28	4	Secteurs cachés (0 par défaut si le disque n'est pas partitionné).
32	4	Nombre total de secteurs 32-bit (Contient une valeur si le nombre total de secteurs 16-bits est égale à 0)

Le tableau suivant explique la suite du BPB pour les disques FAT12/16 (pour le FAT32 voir plus bas) :

Position (octets)	**Taille (octets)**	**Description**
36	1	Identifiant du disque (à partir de 0x00 pour les disques amovibles et à partir de 0x80 pour les disques fixes).
37	1	Réservé pour usage ultérieur.
38	1	Signature (0x29 par défaut).
39	4	Numéro de série du disque.
43	11	Nom du disque sur 11 caractères ('NO NAME' si pas de nom).
54	8	Type de système de fichiers (FAT, FAT12, FAT16).

Le tableau suivant explique la suite du BPB pour les disques FAT32 (pour FAT12/16 voir plus haut) :

Position (octets)	**Taille (octets)**	**Description**
36	4	Taille d'une FAT en secteur (remplace l'équivalent cité au-dessus)
40	2	Attributs du disque.
42	1	Version majeur du système de fichier (0 par défaut).
43	1	Version mineur du système de fichier (0 par défaut).
44	4	Numéro du premier cluster du répertoire racine.
48	2	Informations supplémentaire sur le système de fichier (1 par défaut).
50	2	Numéro de secteur de la copie du secteur de boot.
52	12	Réservé pour des ajouts ultérieur (0 par défaut).
64	1	Identifiant du disque (à partir de 0x00 pour les disques amovibles et à partir de 0x80 pour les disques fixes).
65	1	Réservé pour usage ultérieur.
66	1	Signature (0x29 par défaut).
67	4	Numéro de série du disque.
71	11	Nom du disque sur 11 caractères ('NO NAME' si pas de nom).
82	8	Type de système de fichiers (FAT32).

Les "tables d'allocation" ou "FATs"

Une table d'allocation est une carte où chaque nombre représente un *cluster*. Cette table est un tableau de nombres, indexé par un numéro de cluster. Le tableau suivant donne les valeurs possibles pour ces nombres :

FAT12	**FAT16**	**FAT32**	**Description**
0x000	0x0000	0x?0000000	Cluster vide
0x001	0x0001	0x?0000001	Cluster réservé
0x002 - 0xFEF	0x0002 - 0xFFEF	0x?0000002 - 0x?FFFFFEF	Cluster utilisé, pointeur vers le cluster suivant du fichier
0xFF0 - 0xFF6	0xFFF0 - 0xFFF6	0x?FFFFFF0 - 0x?FFFFFF6	Valeurs réservées
0xFF7	0xFFF7	0x?FFFFFF7	"Mauvais cluster"
0xFF8 - 0xFFF	0xFFF8 - 0xFFFF	0x?FFFFFF8 - 0x?FFFFFFF	Cluster utilisé, dernier cluster d'un fichier

Exemple :

Numéro	**Cluster 1**	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Cluster 7**	**Cluster X**
La valeur...	0000	0003	0004	0006	0001	0007	FFFF	....
...correspond à	Vide	Utilisé	Utilisé	Utilisé	Réservé	Utilisé	Utilisé	

Remarque : Dans cet exemple un fichier utilise, dans l'ordre, les clusters 2, 3, 4, 6 et 7.

Le "répertoire racine" ou "Root directory"

Le répertoire racine est une liste d'entrée. Les entrées du répertoire racine ont le format suivant :

<table>
<tr><th>Offset</th><th>Taille</th><th>Description</th></tr>
<tr><td>0x00</td><td>8</td><td>Nom du fichier (rempli par des espaces)
Le premier octet peut avoir l'une des valeurs spéciales suivantes :
<table>
<tr><td>0x00</td><td>Entrée disponible, marque également la fin du répertoire</td></tr>
<tr><td>0x05</td><td>Le nom de fichier commence en fait par le caractère ASCII 0xE5 (valeur réservée)</td></tr>
<tr><td>0x2E</td><td>Entrée '.' ou '..'</td></tr>
<tr><td>0xE5</td><td>Entrée supprimée. Les outils de recourvrement de fichiers supprimés remplace ce caractère par un autre pour restituer le fichier supprimé.</td></tr>
</table></td></tr>
<tr><td>0x08</td><td>3</td><td>Extension (rempli par des espaces)</td></tr>
<tr><td>0x0b</td><td>1</td><td>Attributs du fichier :
<table>
<tr><td>Bit</td><td>Masque</td><td>Description</td></tr>
<tr><td>0</td><td>0x01</td><td>Lecture seule</td></tr>
<tr><td>1</td><td>0x02</td><td>Fichier caché</td></tr>
<tr><td>2</td><td>0x04</td><td>Fichier système</td></tr>
<tr><td>3</td><td>0x08</td><td>Nom du volume</td></tr>
<tr><td>4</td><td>0x10</td><td>Sous-répertoire</td></tr>
<tr><td>5</td><td>0x20</td><td>Archive</td></tr>
<tr><td>6</td><td>0x40</td><td>Device (utilisé en interne, jamais sur disque)</td></tr>
<tr><td>7</td><td>0x80</td><td>Inutilisé</td></tr>
</table>
La valeur d'attributs 0x0F est utilisé pour désigner un nom de fichier long.</td></tr>
<tr><td>0x0c</td><td>1</td><td>Reservé, utilisé par NT</td></tr>
<tr><td>0x0d</td><td>1</td><td>Heure de création: par unité de 10ms (0 à 199).</td></tr>
<tr><td>0x0e</td><td>2</td><td>Heure de création :
<table>
<tr><td>Bits</td><td>Description</td></tr>
<tr><td>15-11</td><td>Heures (0-23)</td></tr>
<tr><td>10-5</td><td>Minutes (0-59)</td></tr>
<tr><td>4-0</td><td>Secondes/2 (0-29)</td></tr>
</table></td></tr>
<tr><td>0x10</td><td>2</td><td>Date de création :
<table>
<tr><td>Bits</td><td>Description</td></tr>
<tr><td>15-9</td><td>Année - 1980 (0 = 1980, 127 = 2107)</td></tr>
<tr><td>8-5</td><td>Mois (1 = Janvier, 12 = Décembre)</td></tr>
<tr><td>4-0</td><td>Jour (1 - 31)</td></tr>
</table></td></tr>
<tr><td>0x12</td><td>2</td><td>Date du dernier accès ; voir offset 0x10 pour la description.</td></tr>
<tr><td>0x14</td><td>2</td><td>Index EA (utilisé par OS/2 et NT) pour FAT12 et FAT16 ; 2 octets de poids fort du numéro du premier cluster pour FAT32</td></tr>
<tr><td>0x16</td><td>2</td><td>Heure de dernière modification ; voir offset 0x0e pour la description.</td></tr>
<tr><td>0x18</td><td>2</td><td>Date de dernière modification ; voir offset 0x10 pour la description.</td></tr>
<tr><td>0x1a</td><td>2</td><td>Numéro du premier cluster du fichier (FAT12 et FAT16) ; 2 octets de poids faible de ce numéro (FAT32).</td></tr>
<tr><td>0x1c</td><td>4</td><td>Taille du fichier</td></tr>
</table>

Quand le fichier est un répertoire (voir bits attributs), le contenu est une structure identique au format précédent, et commence par les deux entrées '.' et '..'.

Le système de fichiers Microsoft NTFS (New Technology File System)

NTFS est un système de fichiers conçu pour Windows NT (et ses successeurs chez Microsoft) pour stocker des données sur disque dur. Il s'inspire d'HPFS, le système de fichiers conçu pour OS/2. Le sigle **NTFS** désigne en anglais *New Technology File System* (littéralement nouvelle technologie de système de fichiers). Ce système est arrivé avec la première version de Windows NT, en 1993.

NTFS permet de :

- mettre des droits très spécifiques (ACL) sur les fichiers et répertoires : lecture, écriture, exécution, appropriation, etc.
- chiffrer des fichiers avec EFS (*Encrypting File System*)
- compresser des fichiers
- d'établir des quotas par volume

Quelques caractéristiques :

Introduction	juillet 1993 (Windows NT 3.1)
Identificateur de partition	0x07 (MBR)
Contenu des répertoires	B+ tree (arbre-B)
Allocation de fichiers	B+ tree (arbre-B)
Mauvais blocs	B+ tree (arbre-B)
Taille max. de fichier	en pratique 16 TiB (en théorie 16 EiB) voir remarques
Nombre max. de fichiers	4 294 967 295 ($2^{32} - 1$)
Taille max. de nom de fichier	255 caractères (UTF-16)
Taille max. de volume	en pratique 256 TiB (en théorie 16 EiB) voir remarques
Caractères autorisés	dans les noms de fichiers Unicode (UTF-16), tout caractère à l'exception de "/"
Dates	Plage de dates 1er janvier 1601 - 28 mai 60056
Attributs	Lecture seule, caché, système, archive
Droits du système de fichier	ACLs (**Access Control List**, en français liste de contrôle d'accès)
Compression intégrée	Par fichier, LZ77 (à partir de Windows NT 3.51)
Chiffrement intégré	Par fichier, DES-X (à partir de Windows 2000), Triple DES (à partir de Windows XP), AES (Windows XP Service Pack 1, Windows 2003 et suivants)

Remarques:
En pratique, ces limites sont liées au nombre maximum de clusters (2^{32} - 1 pour la taille maximale de volume, et 2^{28} - 1 pour la taille maximale de fichier), combinée avec la taille maximale d'un cluster, actuellement fixée à 64 KiB, c'est-à-dire un regroupement de maximum 128 (2^7 secteurs physiques de 512 bytes). En théorie, soit 2^{64} clusters - 1, permis d'après l'architecture de NTFS.

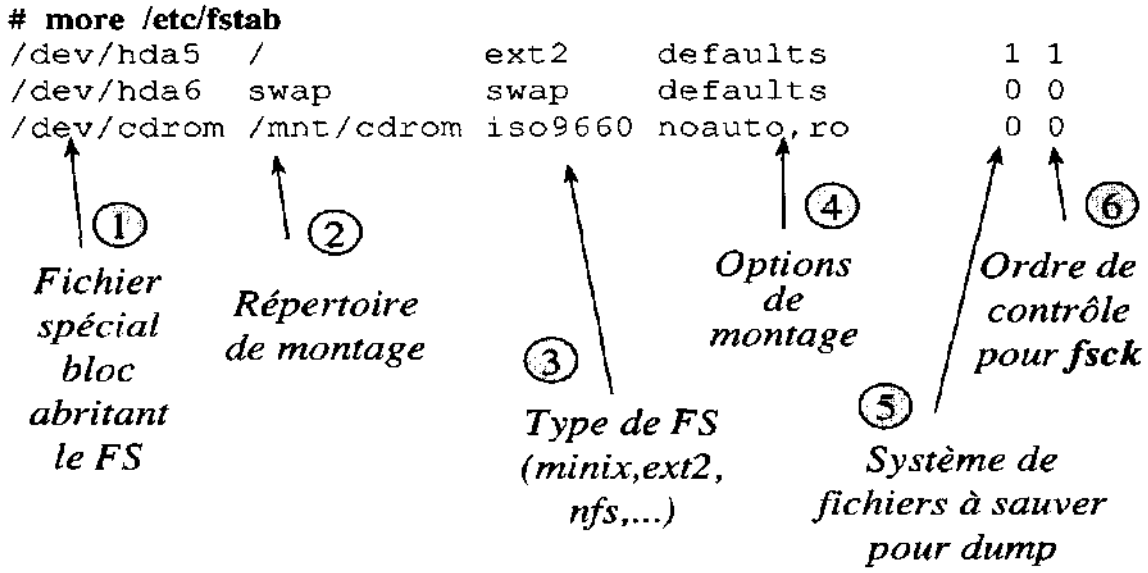
L'**exbibyte** est une unité d'information de stockage informatique. Il est une contracture de EXa BInary BYte. Son abréviation est EiB. 1 exbibyte = 260 bytes = 1,152,921,504,606,846,976 bytes = 1,024 pebibytes

Annexe 1

Les fichiers de configuration

- **/etc/lilo.conf** (pour **LILO**) : lors du démarrage du système, le disque qui contient le système de fichiers principal (la racine **/**) doit être connu pour être automatiquement monté. Ce paramètre de démarrage est défini par la variable **root** dans le fichier **/etc/lilo.conf**. En l'abscence de cette variable, c'est le disque indiqué dans l'image du noyau qui sera monté comme système de fichiers racine. Dans le fichier **/boot/grub/menu.lst** pour **GRUB**.
- **/etc/fstab** : ce fichier contient la liste des systèmes de fichiers à monter au démarrage du système. C'est la commande **mount -a**, exécutée par les scripts de démarrage, qui prend en compte le contenu du fichier **/etc/fstab**.
- **/etc/mtab** : ce fichier contient la liste des systèmes de fichiers actuellement montés sur le système. La liste peut aussi être visualisée par la commande **mount**.

Le fichier /etc/fstab



Le montage/démontage à la volée

On distingue 2 techniques : ***automount*** et ***supermount***.

Il faut avoir installé le paquetage **autofs** pour disposer du démon **automount**. C'est lui qui réalise automatiquement le **montage** d'un système fichiers si un processus manipule des fichiers situés en dessous du répertoire de montage et le **démontage** quand l'arborescence n'est plus utilisée.
L'automontage est notamment utilisé pour les disques amovibles et les systèmes de fichiers réseaux.
La configuration de ce service est réalisée par le fichier **/etc/auto.master** qui permet d'indiquer les répertoires en dessous desquels sont réalisés les montages et les noms des fichiers qui décrivent le détail des montages.

Le service **supermount** permet le montage automatique des disques amovibles. On ajoutera une demande de montage automatique en l'indiquant dans le fichier **/etc/fstab**.

Gestion de l'espace disque

La commande **df** indique l'espace libre des disques contenant des systèmes de fichiers montés. Les tailles sont affichés en Kilo octets. Les options les plus utilisés sont : **-i** (utilisation des inodes), **-T** (indique le type de FS).

La commande **du** affiche le nombre de blocs utilisés par une arborescence. Les options les plus utilisés sont : **-s** (affiche le total seulement) et **-h** (affichage « humain » en Kilo octets).

La commande **find** permet de rechercher des fichiers selon différents critères.

Les quotas

La mise en oeuvre des quotas va permettre à l'administrateur de limiter :

- le nombre de fichiers et / ou
- le nombre de blocs d'un utilisateur ou d'un groupe

Il existe deux types de limites :

- la limite ***hard*** qui est infranchissable
- la limite ***soft*** qui peut être franchie pendant un certain nombres de jours (7 par défaut). Après, le point atteint devient à sont our infranchissable jusqu'au retour à la normale.

Les commandes de gestion des quotas sont : **`edquota, quotaon, quotaoff, quota, repquota, quotacheck`**, ...

Les fichiers purgeables

Il existe des fichiers qui peuvent être détruits ou purgés (avec une périodicité qui varie selon les cas) :

- les fichiers **temporaires** créés par les applications dans les répertoires **/tmp** et **/var/tmp**
- les fichiers ***log*** qui enregistrent des informations relatives au suivi d'un service ou d'un logiciel. L'administrateur peut les purger dès qu'il a analysé leur contenu et les a archivés si besoin. Ils sont situés principalement dans **/var/log/**.
- les fichiers ***core*** qui sont générés lorsqu'un processus meurt suite à une anomalie de fonctionnement. Ils contiennent l'image du processus au moment de sa mort (« boite noire ») et sont destinés au programmeur pour effectuer un débuggage post-mortem.

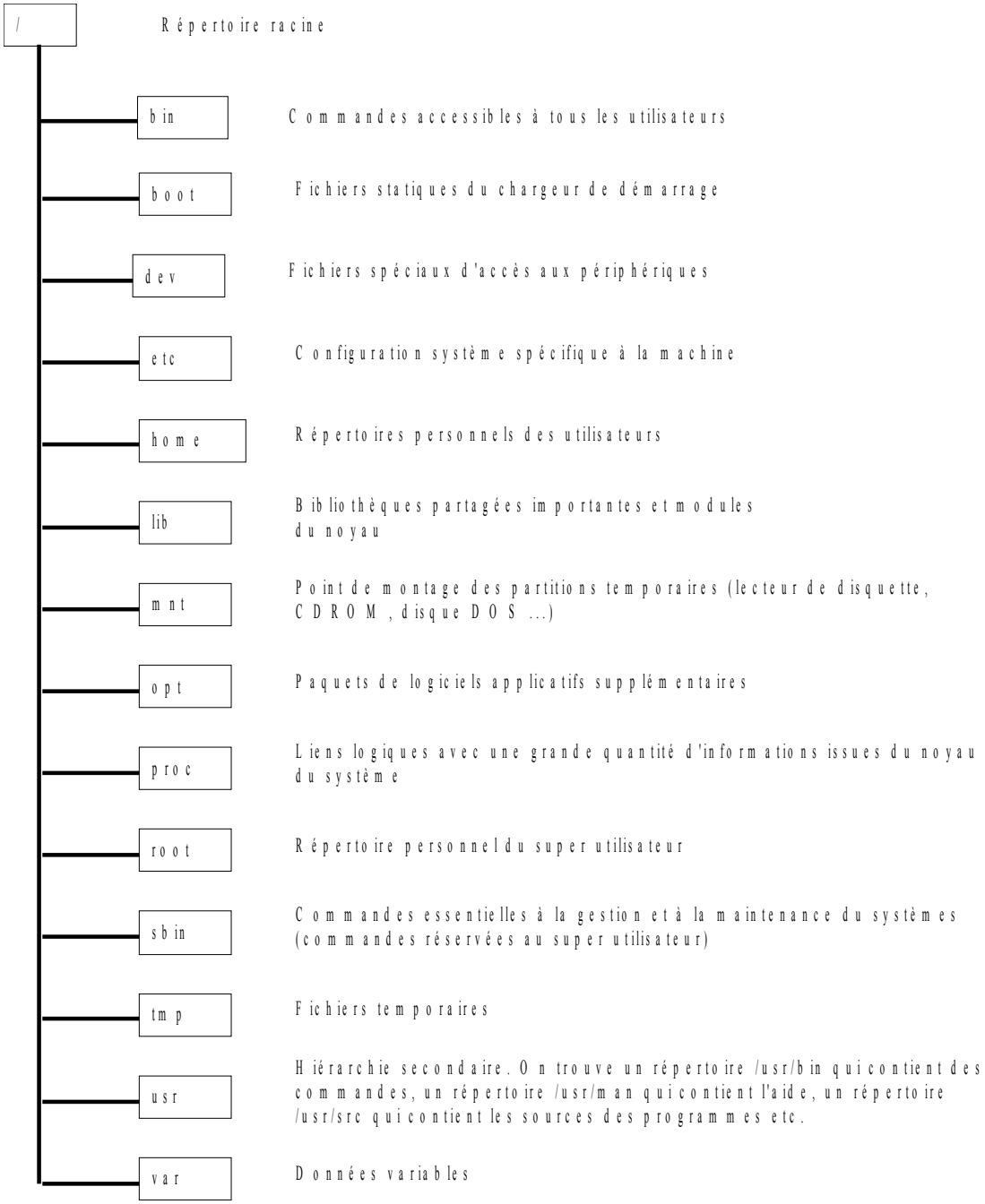
Pour vider un fichier : **`$ cp /dev/null nom_fichier`**

Les sparse files

Les systèmes de fichiers Unix sont également capables de prendre en charge ce que l'on appelle les fichiers « troués » (« *sparse files* » en anglais). Ces fichiers sont des fichiers contenant des données séparées par de grands espaces vides. On peut donc dire qu'ils sont « presque vides ».

Pour ces fichiers, il est évident qu'il est inutile de stocker les espaces vides sur le disque. Les systèmes de fichiers mémorisent donc tout simplement qu'ils contiennent des trous, et ils ne stockent que les données réelles et la position des trous avec leurs tailles. Cela constitue une économie de place non négligeable. Les applications classiques des fichiers presque vides sont les bases de données, qui utilisent souvent des fichiers structurés contenant relativement peu de données effectives, et les images disque pour installer des systèmes d'exploitation dans des machines virtuelles ou encore les fichiers en cours de téléchatgement dans les applications *peer-to-peer*.

Annexe 2 : l'arborescence standard de Linux



Pour obtenir de plus amples informations sur cette arborescence, il est possible de consulter la norme FHS (Filesystem Hierarchy Standard) disponible à http://www.pathname.come/fhs ou en traduction française à ftp://ftp.lip6.fr/pub/linux/french/docs/fhs-2.0.fr.tar.gz.

Des conseils pour le partitionnement sous Mandriva Linux : http://wiki.mandriva.com/fr/Partitionner

L'arborescence :

/	racine du système de fichiers
/bin	programmes nécessaires pour démarrer le système en mode mono-utilisateur.
/dev	(*device files*) points d'entrées vers des périphériques physiques
/etc	fichiers de configuration
/home	données propres à chaque utilisateur
/lib	(*shared libraries*) nécessaires aux programmes de démarrage (principalement des programmes placés dans /bin et /sbin).
/mnt	pour le montage pour les périphériques en mode block (cdrom, floppy ...)
/proc	infos sur l'état du système et les différents processus
/sbin	programmes nécessaires au fonctionnement du système
/tmp	fichiers temporaires
/usr	(usr = *Unix System Resources*) Données que les utilisateurs peuvent se partager
/var	fichiers qui sont susceptibles de changer fréquemment : logs, les files d'attentes pour les impressions, etc.

Remarque : Différence entre bin et sbin : bin=binaires, sbin=binaires système. Pour un utilisateur standard, bin peut être dans le PATH, mais sbin n'a aucune raison d'y être.

Les répertoires dans /usr :

X11R6	système Xwindow
X11R6/bin	exécutables pour le système Xwindow.
X11R6/lib	librairies pour les programmes dans /usr/X11R6/bin.
X11R6/lib/X11	librairies utiles au démarrage du serveur Xwindow.
X11R6/include/X11	fichiers d'entêtes des applications X11.
bin	programmes utilisables sur le système.
bin/X11	liens symboliques (et historiques) vers /usr/X11R6/bin
dict	dictionnaires utilisés par les vérificateurs orthographiques
etc	fichiers de configuration qui peuvent être partagés entre plusieurs machines.
include	fichiers entêtes (.h) pour le compilateur C
lib	librairies utilisées par les programmes des utilisateurs
local	ce qui est spécifique à la machine locale
local/bin	programmes
local/lib	librairies
local/doc	documentations
man	man pages
src	sources des diverses applications installées sur le système.
src/linux	sources du noyau.

Les répertoires dans /var :

lock	fichiers de verrouillage.
log	logs du système
spool	contient les files d'attentes d'impressions
spool/cron	entrées crontab pour l'automatisation des tâches.
spool/lpd	fichiers en attente d'impression.
spool/mail	boites à lettres (mailbox) et les messages (mails) des utilisateurs.