ENSEIGNEMENT DE PROMOTION SOCIALE

Cours de

STRUCTURE DES ORDINATEURS

- Le disque dur -

H. Schyns

Janvier 2004

Sommaire

3. Le disque dur

3.1. Introduction

Le disque dur (*ang.: hard disk* ou *HD*) est incontestablement la partie la plus précieuse de l'ordinateur.

Brûler une alimentation, plier une patte d'un processeur, briser une barrette de mémoire sont de pannes souvent gênantes, parfois coûteuses mais jamais irremplaçables.

Perdre un disque dur est presque toujours synonyme de catastrophe.

Selon certaines statistiques, 30% des sociétés qui ont dû faire face à une perte de disque dur sans avoir de copie de sauvegarde récente tombent en faillite dans les trois mois qui suivent l'accident !

Il est donc capital de bien comprendre le fonctionnement de cet organe.

3.2. Structure physique

Dans un PC, le disque dur se présente sous la forme d'un boîtier en alliage d'aluminium d'environ 1.5 cm d'épaisseur, 10 cm de large et 15 cm de long. Il est généralement fixé dans un berceau placé sur la partie antérieure du chassis du PC. Il est relié à la carte mère par un large ruban de connexion et reçoit son alimentation électrique par une fiche à 4 fils. Ce boitier ne doit ***jamais*** être ouvert : qu'une simple poussière microscopique pénètre à l'intérieur et c'est la mort du disque dur.

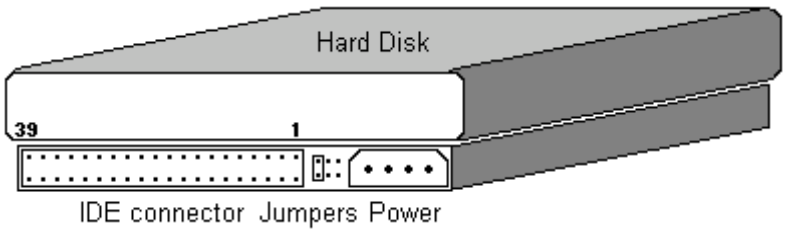


Figure 3.1 Schéma d'un disque dur

La face inférieure du disque dur comporte un circuit imprimé qui contrôle les échanges entre le disque et la carte mère : le contrôleur IDE.

Figure 3.2 Vue du contrôleur IDE du disque dur sur la face inférieure

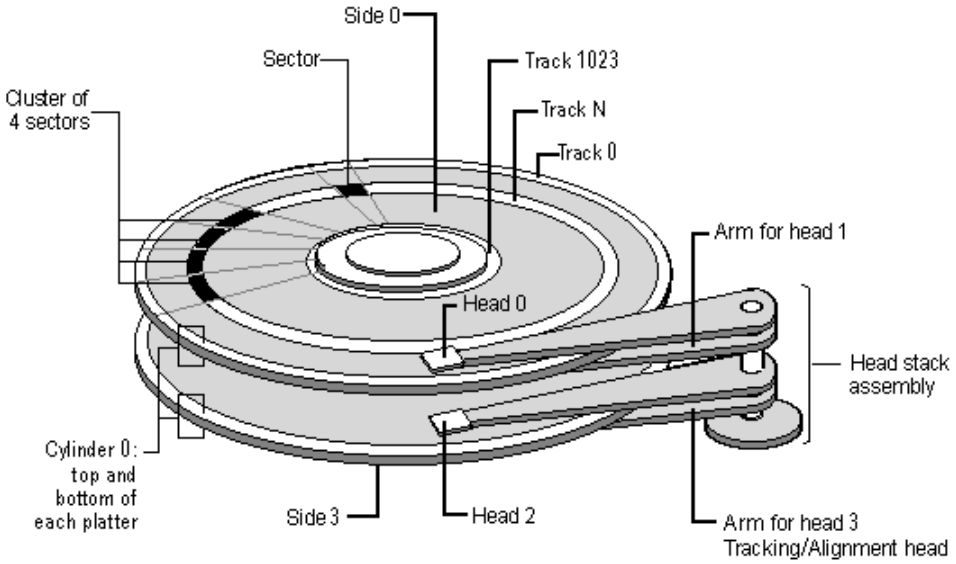


Figure 3.3 Schéma de principe d'un empilage de disques

A l'intérieur du boîtier se trouvent un ou plusieurs **disques** empilés, faits d'un polymère très résistant. Les deux **faces** (*ang.: side*) de chaque disque sont complètement recouvertes d'une substance magnétique comparable à celle que l'on trouve sur les bandes vidéo. Les disques tournent tous ensemble, à vitesse constante, autour de leur axe commun. Les vitesses de rotation vont de 3 000 à plus de 10 000 tours/min, selon les modèles.

A chaque face correspond une ou plusieurs **têtes de lecture** magnétique (*ang.: head*) placée au bout d'un bras (*ang.: arm*) à la manière de nos anciennes platines HIFI. Tous les bras sont solidaires et se déplacent en même temps (*head stack assembly*). Les faces et les têtes de lecture sont numérotées à partir de 0 ([1]).

Avant de quitter l'usine, le revêtement magnétique est virtuellement structuré en **pistes** (*ang.: tracks*) circulaires et concentriques. C'est le formatage de bas niveau encore appelé **formatage physique**. Les pistes de chaque face sont numérotées à partir de 0 en commençant par l'extérieur.

Comme tous les disques de l'empilage sont identiques, les pistes qui possèdent le même numéro se trouvent à la verticale les unes des autres. Elles forment un **cylindre** virtuel (*ang.: cylinder*). Structurer les données en cylindres réduit considérablement leur temps d'accès. En effet, le mouvement des têtes de lecture d'un cylindre à l'autre prend plus de temps que la rotation d'un cylindre sous une même tête.

Les pistes sont elles-mêmes fractionnées en **secteurs** (*ang.: sectors*). Les secteurs sont séparés les uns des autres par des zones spéciales contenant des informations de contrôle. Chaque secteur est précédé d'une zone d'en-tête (*ang.: header*) et suivi d'une zone de fin de secteur (*ang.: trailer*). On a donc la structure suivante :

… | header j | sector j | trailer j | header j+1 | sector j+1 | trailer j+1 | …

Chaque secteur a une capacité de stockage de **512 bytes**. Cette valeur est une norme internationale.

Dans les premiers disques durs, toutes les pistes contenaient le même nombre de secteurs. Cette contrainte entraînait un gaspillage important de la surface magnétique, surtout sur les pistes extérieures qui, étant plus longues, auraient pu

1 S'il y a n disques, et k têtes de lecture par face, les têtes de lecture seront numérotées de 0 à 2kn-1.

supporter des secteurs supplémentaires (zones grisées dans la figure ci-dessous, à gauche).

Le problème a été résolu en utilisant le **formatage par zone** (figure ci-dessous à droite). Le cylindre le plus proche du centre est divisé en un certain nombre de secteurs (p.ex. 5). Il en va de même pour les cylindres voisins (zone 1), tant que la circonférence n'est pas assez grande pour intercaler un secteur de plus. Dès que c'est le cas, on crée une nouvelle zone (p.ex. 6 secteurs, zone 2) et on reprend le processus jusqu'à atteindre le cylindre extérieur.

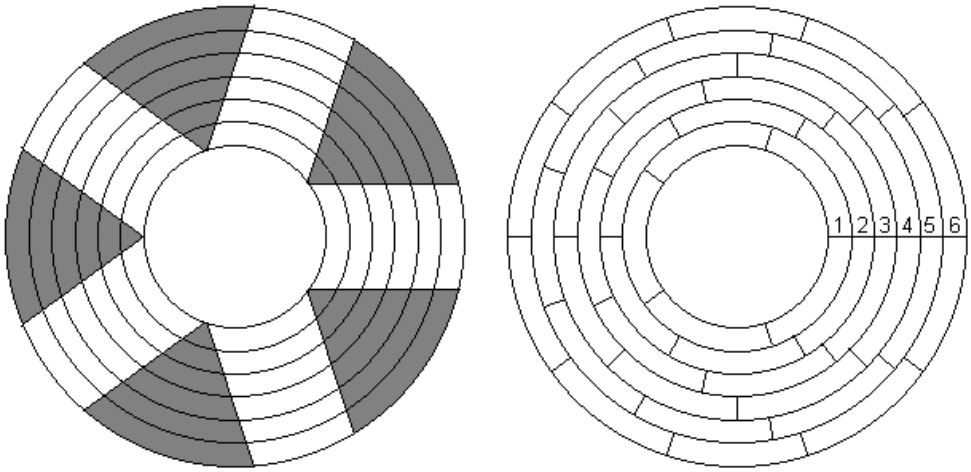


Figure 3.4 Formatage simple et formatage par zones

Sur le boitier du disque dur, le constructeur colle un étiquette qui reprend ses caractéristiques physiques : nombre de cylindres, nombre de secteurs par cylindre, nombre de têtes de lecture.

Evidemment, ceci pose un problème dans le cas du formatage par zones car le nombre de secteurs par cylindre n'est pas constant. On calculera donc un nombre moyen de secteurs par cylindre. Une table de conversion, stockée dans un chip du disque dur contiendra la table de conversion entre la position logique du secteur telle que vue par le PC et la position réelle du secteur telle que connue du disque dur. Cette technique se nomme LBA (Logical Block Adressing).

A titre d'exemple, imaginons que le PC, qui ignore tout de le disposition réelle des secteurs sur le disque, demande de renvoyer le contenu du secteur 6 du cylindre 4. Le contrôleur de disque, consultant sa table interne, verra, par exemple que ce secteur correspond physiquement au secteur 8 du cylindre 3.

La capacité *brute* de stockage d'un disque dur est donnée par le produit des trois caractéristiques :

Capacité (bytes) = têtes • cylindres • secteurs • 512

Comme 1 kb = 1024 b et que 1 Mb = 1024 kb, la capacité *en Mb* est donnée par

Capacité (Mb) = têtes • cylindres • secteurs / 2048

La capacité réelle dépendra du type de formatage et sera toujours légèrement inférieure à la capacité théorique.

Le diamètre de la piste "moyenne" d'un disque est de l'ordre de 6 cm. Par ailleurs, une piste est souvent segmentée en 64 secteurs. On en conclut que la longueur d'un secteur ne dépasse pas 3 mm.

D'un autre côté, on sait que le rayon du disque est de l'ordre de 4 cm et qu'il possède une pastille centrale, d'un rayon de 1 cm, qui sert à son entraînement mécanique. Il reste donc 3 cm pour tracer les 2000 à 4000 cylindres du disque. On voit que la largeur d'une piste ne peut être supérieure à 10 microns environ.

Conclusion : un secteur occupe moins d'espace que le signe - et contient 512 bytes !

3.3. Composants de la structure logique

Lorsqu'il quitte le fabricant, le disque dur est semblable à un vaste entrepôt immensément vide.

Chacun sait que disposer d'une *surface* au sol n'est pas suffisant. Il faut structurer cette surface afin de gérer correctement l'*espace* disponible. Nous allons donc définir une **structure logique** du disque. C'est la raison d'être du système de fichier.

Le **système de fichiers** définit les structures nécessaires au stockage et à la gestion des données. Il a trois fonctions principales :

- contrôler l'espace alloué et inutilisé,
- gérer les répertoires et les fichiers,
- repérer l'emplacement physique des fichiers sur le disque.

3.3.1. L'unité d'allocation

Le disque dur et ses secteurs peuvent être comparés à un hôtel de type "Formule 1" dans lequel toutes les chambres sont identiques.

Dans un hôtel il n'est pas possible de réserver seulement quelques mètres carrés de moquette pour se coucher. Il faut réserver une ou plusieurs chambres entières. Il en va de même dans notre PC : quand un système d'exploitation désire réserver de l'espace sur le disque dur, il doit réserver un nombre entier de secteurs (soit n•512 bytes). Il n'est pas possible de réserver seulement quelques bytes. On dit que **le secteur** est l'unité d'allocation minimale.

Dès lors, un texte qui contient un seul caractère, soit 1 byte, bloque en fait un secteur de 512 bytes et diminue d'autant la capacité disponible du disque dur.

3.3.2. La FAT

Dans un hôtel, toutes les chambres sont numérotées. Le préposé à la réception dispose d'un tableau auquel sont accrochées toutes les clés. Chaque clou du tableau correspond à une chambre. Si la clé est au tableau, c'est que la chambre est disponible; si la clé n'est pas accrochée au tableau, le préposé en déduit que la chambre est occupée ([1]).

Le disque dur dispose d'un système similaire appelé FAT (*ang.: **F**ile **A**llocation **T**able; fr.: Table d'Allocation des Fichiers*).

Comme les chambres de l'hôtel, les secteurs du disque sont numérotés. La FAT est **un tableau** (*ang.: array*) qui contient autant d'éléments qu'il y a de secteurs sur le disque. Si un élément de la FAT contient 0, c'est que le secteur correspondant est libre; s'il contient une valeur différente de 0 (p.ex. 1), c'est qu'il est occupé. Tant qu'il s'agit simplement de savoir si le secteur est libre ou non, un seul bit suffit par secteur.

1 Oui, je sais, ce n'est pas nécessairement vrai. Pour les besoins de la comparaison, nous supposerons donc que les voyageurs restent bouclés dans leur chambre depuis l'instant de leur arrivée jusquà ce qu'ils quittent l'hôtel.

3.3.3. Les répertoires

Savoir que telle ou telle chambre est occupée ne suffit pas. Notre hôtelier doit aussi savoir *par qui* elle est occupée. Il doit donc disposer d'une liste des clients et des chambres qu'ils occupent.

De même, le système d'exploitation doit disposer de la liste des noms des fichiers inscrits sur le disque et du numéro de secteur qu'ils occupent. Une telle liste porte le nom de **répertoire** (*ang.: directory*). Nous y reviendrons au point 3.5.

Attention ! Il ne faut pas confondre. La FAT est une simple liste d'occupation qui ne contient pas d'informations sur les fichiers. Le répertoire contient l'information sur le nom des fichiers mais aucun détail sur la manière dont les secteurs sont organisés.

3.3.4. La table de partition

Un disque dur peut être partagé en plusieurs morceaux appelés **partitions**. Du point de vue de l'utilisateur, chaque partition apparaît comme un disque dur distinct. Cette technique permet de décomposer **un disque dur "physique"** en deux, trois ou **quatre disques "logiques"**. Ainsi, un disque C: de 20 Gb peut être décomposé en quatre disques C:, D:, E:, F: dont les capacités seraient par exemple de 2, 5, 6 et 7 Gb respectivement.

Comment est-ce possible ?

Nous avons vu au point 3.2 qu'un disque dur est structuré en pistes concentriques (*ang.: tracks*) qui forment des cylindres virtuels. Ces cylindres sont numérotés à partir de 0 en commençant par l'extérieur du disque et en se dirigeant vers l'intérieur.

Chaque partition comprend un certain nombre de cylindres consécutifs : les cylindres 0 à N_1 constituent la première partition, les cylindres N_1+1 à N_2 forment la deuxième, N_2+1 à N_3, la troisième et ainsi de suite jusqu'au dernier cylindre. Cette information est stockée dans la **table de partition** (*ang.: partition table*) située dans les tout premiers secteurs du disque dur.

Puisque chaque partition agit comme un disque dur isolé, chaque partition est gérée de manière indépendante à l'aide de sa propre FAT et de ses propres répertoires.

3.3.5. La zone d'amorçage (MBR)

Lorsque l'ordinateur est mis sous tension, l'unité centrale exécute immédiatement un petit programme encodé "en dur" dans la mémoire morte de l'ordinateur (BIOS).

Ce petit programme effectue un certain nombre de tests puis lit la mémoire CMOS qui contient la description du système : date et heure, présence d'un lecteur de disquettes, présence et structure d'un ou de plusieurs disques durs, etc. Le CMOS contient aussi la référence du disque "physique" qui doit être utilisé pour le démarrage du système.

Le tout *premier secteur* de ce disque de démarrage est *obligatoirement* un secteur d'amorce (*ang.: Master Boot Record* ou *MBR*). Tout comme le BIOS, le MBR contient un petit programme de démarrage qui prend la direction des opérations ([1]).

Ce petit programme lit la table de partitions, recherche la partition active, lit le secteur d'amorçage de la partition et cède la main au programme d'amorçage de la partition.

1 En fait, le MBR contient une zone de code exécutable **et** la table de partition.

Ce dernier recherche les références des fichiers nécessaires au démarrage du système (tel que **CONFIG.SYS** et **COMMAND.COM**) puis lance le système proprement dit.

En résumé, la séquence de démarrage (boot) est la suivante :

- boot BIOS;
- boot Disk;
- boot Partition;
- boot System.

La zone amorce de la partition occupe soit **1**, soit **2** secteurs selon le système installé. Il faut donc être prudent en transférant un système sur un disque : faire entrer un "boot sector" de 2 secteurs dans une partition qui n'avait réservé qu'un seul secteur à cet effet peut conduire à la perte de tout le disque.

3.4. Fonctionnement de la FAT

3.4.1. Principe général

La FAT est donc un de tableau de réservation et d'occupation des secteurs du disque dur.

Quand le système a besoin d'un secteur pour écrire des données, il lui suffit d'examiner la FAT à la recherche du premier élément contenant un 0. Ceci fait, il réserve le secteur en remplaçant le 0 de la FAT par une autre valeur puis inscrit les données dans le secteur correspondant.

Quand le système veut libérer un secteur, par exemple quand un fichier est supprimé, il lui suffit d'écrire un 0 à l'endroit correspondant de la FAT. *Notez bien qu'il n'est pas nécessaire d'effacer physiquement le secteur* : la prochaine écriture dans ce secteur écrasera d'office les anciennes informations pour les remplacer par les nouvelles. C'est un gain de temps appréciable ([1]).

Ça, c'est pour le principe. Hélas, les choses ne vont pas tarder à se compliquer…

3.4.2. Bad sectors

Dans notre hôtel, il arrive que certaines chambres soient mises hors service : parce que le lavabo est bouché, parce que les ouvriers sont en train d'y changer la moquette ou parce qu'elle a été dévastée par un incendie. L'hôtelier ne peut pas considérer que cette chambre est libre. Il ne peut pas non plus la noter comme occupée car il chercherait en vain le nom du client dans sa liste. Il doit donc apposer un signe distinctif sur son tableau.

Un phénomène similaire apparaît sur le disque dur. Suite à un défaut de la couche magnétique, suite au dépôt d'une micro poussière, certains secteur sont défectueux (*ang.: bad sector*) : ce qu'on y écrit s'écrit mal et, de ce fait, ce qu'on y lit ne correspond pas à ce qu'on a voulu y écrire. De tels secteurs doivent être retirés de la circulation. Ils seront signalés dans la FAT par un nombre particulier conventionnel.

1 Cette technique a des avantages et des inconvénients. Avantage : un fichier qui vient d'être effacé par erreur peut être récupéré tant que le secteur libéré n'a pas été réutilisé. Inconvénient : il ne suffit pas d'effacer un fichier confidentiel pour faire disparaître son contenu; il faut s'assurer que le secteur a bien été réutilisé.

3.4.3. Fragmentation

On comprend qu'à force d'écrire et de supprimer des fichiers, le disque dur devienne bientôt une mosaïque de secteurs libres et de secteurs attribués. Ce phénomène est connu sous le nom de **fragmentation** du disque dur. Il est ***impossible*** de l'éviter. Il provoque une dégradation importante des performances du disque dur.

Heureusement, il existe des utilitaires de défragmentation qui vont réorganiser tout ça. Nous aurons l'occasion d'y revenir.

3.4.4. Allocation de gros fichiers

Le système marche à merveille tant que les données à inscrire occupent moins de 512 bytes. Mais que fait-on quand un rapport demande un ou deux mégabytes ?

Imaginez un car de touristes arrivant dans notre hôtel. Le patron arrivera sans doute à loger tout le monde, mais certainement pas dans des chambres contiguës… à moins de déplacer tous les clients déjà installés, ce qui risque de prendre du temps et de provoquer une belle pagaille.

De même, pour stocker notre rapport, le système arrivera sans doute à trouver le nombre de secteurs libres nécessaire mais ceux-ci ne seront certainement pas contigus. Nous devons donc trouver un truc pour expliquer au système quels sont les secteurs qui contiennent des morceaux du rapport. Nous devons aussi lui expliquer dans quel ***ordre*** il doit lire ces secteurs pour reconstituer le document initial.

Le problème est résolu au moyen d'une **liste chaînée**. Le principe est représenté ci-dessous.

FAT									
-01- **09**	*-02-* **08**	*-03-* **FF**	*-04-* 0	*-05-* 0	*-06-* **FF**	*-07-* 0	*-08-* **03**	*-09-* **FF**	*-10-* 0
-11- 0	*-12-* 0	*-13-* 0	*-14-* **FF**	*-15-* 0	*-16-* 0	*-17-* 0	*-18-* 14	*-19-* 0	*-20-* 0

SECTEURS du DISQUE (512 bytes)									
-01-	*-02-*	*-03-*	*-04-*	*-05-*	*-06-*	*-07-*	*-08-*	*-09-*	*-10-*
-11-	*-12-*	*-13-*	*-14-*	*-15-*	*-16-*	*-17-*	*-18-*	*-19-*	*-20-*

Figure 3.5

Admettons que notre disque contienne 20 secteurs numérotés de 01 à 20. La FAT contient donc 20 éléments.

Supposons que tout notre disque soit libre sauf les secteurs 01, 02, 03, 06, 08, 09, 14 et 18. Tous les éléments de la FAT contiennent la valeur 0 sauf ceux qui correspondent aux secteurs occupés.

Arrive maintenant un fichier de 2800 bytes. Pour stocker ce fichier sur le disque nous avons besoin de 6 secteurs ($5 \bullet 512 < 2800 \leq 6 \bullet 512$).

Le système explore la FAT et découvre que les secteurs 04, 05, 07, 10, 11 et 12 sont libres. Il peut donc les remplir l'un après l'autre avec le contenu du fichier (représenté par les paquets U, V, W, X, Y, Z).

FAT									
-01- 09	*-02-* 08	*-03-* FF	*-04-* **05**	*-05-* **07**	*-06-* FF	*-07-* **10**	*-08-* 03	*-09-* FF	*-10-* **11**
-11- **12**	*-12-* **FF**	*-13-* 0	*-14-* FF	*-15-* 0	*-16-* 0	*-17-* 0	*-18-* 14	*-19-* 0	*-20-* 0

SECTEURS du DISQUE (512 bytes)									
-01-	*-02-*	*-03-*	*-04-* **U**	*-05-* **V**	*-06-*	*-07-* **W**	*-09-*		*-10-* **X**
-11- **Y**	*-12-* **Z**	*-13-*	*-14-*	*-15-*	*-16-*	*-17-*	*-18-*	*-19-*	*-20-*

Figure 3.6

Ceci fait, dans la FAT, il change les "0" qui correspondent aux blocs utilisés. Mais au lieu d'y écrire n'importe quoi, le système va y noter le numéro du secteur dans lequel se trouve la suite du fichier.

Rien ne change au niveau du répertoire racine : il suffit d'y inscrire le nom du fichier et le numéro du *premier* secteur qu'il occupe. Le chaînage de la FAT fera le reste.

Notons au passage qu'un secteur contient le début d'un fichier si deux conditions son remplies :

- il est marqué non libre dans la FAT,
- son numéro n'est jamais repris dans le chaînage de la FAT.

On voit que la FAT joue un rôle essentiel dans la sauvegarde et la récupération des fichiers : effacer la FAT par erreur, par mauvaise manipulation ou par un virus, c'est perdre le disque. C'est pourquoi le système conserve toujours **deux copies** de la FAT et veille à ce qu'elles soient toujours identiques. Certains utilitaires comme "mirror" font une copie supplémentaire de la FAT dans les derniers secteurs du disque dur.

3.4.5. Suppression de gros fichiers

Nous avons vu qu'il est simple de supprimer un fichier qui n'occupe qu'un seul secteur : il suffit de lire le répertoire racine pour trouver la référence du premier secteur puis d'inscrire 0 dans l'élément correspondant de la FAT.

Supprimer un gros fichier n'est guère plus compliqué :

- lire le répertoire racine pour trouver la référence du premier secteur,
- lire dans la FAT la référence au secteur suivant s'il y en a une,
- inscrire 0 dans l'élément de la FAT,
- passer à l'élément suivant dans la chaîne,
- effacer le nom du fichier dans le répertoire.

On notera que les secteurs libérés conservent leur contenu jusqu'à ce qu'ils soient à nouveau attribués à un fichier.

3.4.6. FAT 16 et clustering

Puisque chaque élément de la FAT est susceptible de contenir le numéro d'un secteur, il faut prévoir la place nécessaire.

Nous avons vu au chapitre sur le calcul binaire qu'un byte (8 bits) permet d'encoder les nombres de 0 à 255 (0 à 11111111_b, soit 0_h à FF_h). Or, le nombre 0 est déjà retenu pour indiquer qu'un secteur est libre et le nombre FF est déjà retenu pour indiquer que le secteur est le dernier de la chaîne. Dès lors, une FAT dont les éléments ne pourraient accueillir qu'un byte ne pourrait gérer qu'un disque de 254 secteurs, soit 127 kb. Autant dire rien !

En numérotant les secteurs sur 2 bytes (**16 bits**), nous disposons de nombres allant de 0 à $65\,535_d$ (de 0 à $FFFF_h$). Cette option, connue sous le nom de **FAT 16** fut retenue au début de l'histoire des PC. La liste des nombres "réservés" fut un peu étendue :

- 0000_h signifie que le secteur est libre
- $FFF0_h$ à $FFF6_h$ sont réservés pour de futurs développement
- $FFF7_h$ signale un "bad sector"
- $FFF8_h$ à $FFFF_h$ marquent la fin de fichier (*ang.:* ***EOF*** = *End Of File*)
- les nombres de 1 à 65519 (de 0001_h à $FFEF_h$) sont disponibles pour numéroter les secteurs.

Ce système permet donc d'adresser 65 519 secteurs de 512 bytes, soit un disque de 31.99 Mb. N'oublions pas que pour s'installer sur le disque, la FAT elle-même a besoin de 2 bytes par secteur adressable, soit un maximum de 131 038 bytes ou 256 secteurs de 512 bytes (128 kb).

La solution était donc très satisfaisante pour les possibilités techniques de l'époque mais l'arrivée de disques de plus de 32 Mb allait évidemment poser un problème.

Le problème fut résolu très simplement : puisqu'on ne peut plus numéroter tous les secteurs d'un disque, considérons des groupes de 2 secteurs indissociables (soit une plage de 1024 bytes). Un peu comme si le gérant de notre hôtel décidait que toutes les chambres contiendraient dorénavant deux lits au lieu d'un.

Dorénavant, l'allocation et la libération de l'espace disque se feront par groupe de 2 secteurs appelé **cluster**. La FAT n'indiquera plus si un *secteur* est libre, occupé ou défectueux mais bien si un *cluster* est libre, occupé ou défectueux.

Un fichier de 2800 bytes qui occupait 6 secteurs occupe maintenant 3 clusters de 2 secteurs. Malheureusement, un fichier qui occupait 1 byte et qui bloquait 1 secteur de 512 bytes bloque maintenant un cluster de 1024 bytes. On voit que le système marche bien pour les gros fichiers mais qu'il conduit à un gaspillage de place pour les petits. Idem si un secteur est défectueux : c'est la totalité du cluster qui est marquée "bad".

Qu'importe, puisqu'il permet d'utiliser des disques de 65 519 clusters de 1024 bytes, soit 63.98 Mb. De toutes façons, les programmes sont de plus en plus gros et leurs fichiers aussi.

La capacité des disques augmentant sans cesse, on a poursuivi la technique du "clustering" en passant successivement à des clusters de 4, 8, 16, 32 et 64 secteurs (voir tableau). Notons au passage qu'il faudra bien noter quelque part – sur le disque lui-même, dans la table de partition - le nombre de secteurs compris dans un cluster.

Fort bien, mais avec des clusters de 64 secteurs, l'écriture d'un simple byte sur le disque peut bloquer 32 kb et un simple secteur défectueux provoque la mise à l'écart de tout le cluster. Bref, le gaspillage ne fait qu'augmenter. *Il est en moyenne de 16 kb pour chaque fichier du disque dur*.

FAT 16		
Secteurs par cluster	Taille du cluster (kilobytes)	Capacité du disque (mégabytes)
1	0,5	31.99
2	1	63.98
4	2	127.97
8	4	255.93
16	8	511.86
32	16	1023.73
64	32	2047.46

Figure 3.7

On peut théoriquement poursuivre de la sorte avec des clusters de 128 et 256 secteurs, ce qui permettrait d'utiliser une FAT 16 pour des disques de 8 Mb. Toutefois, au-delà de 64 secteurs par cluster, les problèmes commencent à apparaître : le gaspillage devient vraiment considérable, la numérotation des cylindres demande quelques astuces et surtout, bon nombre de programmes n'arrivent pas à lire des blocs de plus de 32 kb ([1]). Faites le test avec l'éditeur Blocnote.exe (*ang.: notepad.exe*) : il vous demandera si vous ne préférez pas utiliser Wordpad.

3.4.7. FAT 32

Pour dépasser le barrière des 2 Gb, il y avait deux possibilités :

- découper le disque en zones; nous en parlerons au chapitre consacré au partitionnement,
- changer la manière de numéroter les clusters, ce que nous allons voir ci-après.

Au lieu de numéroter les clusters sur 2 bytes, passons à 4 bytes et créons une **FAT 32**. Nous avons ainsi accès aux nombres allant de 0_h à FFFF $FFFF_h$, soit de 0 à 4 294 967 295_d ! Même en se limitant à des clusters d'un seul secteur, l'espace adressable atteint 2048 Gigabytes, soit 2 Térabytes (Tb) !

Un seul problème, sur un tel disque, une telle FAT 32 occuperait à elle seule 16 Gigabytes… et il y a toujours au moins deux copies de la FAT. Le problème n'est plus tellement de lire le disque mais bien de lire la FAT et d'en faire une copie en mémoire centrale.

On peut cependant trouver un compromis : si on fait des clusters de 2 secteurs, la taille de la FAT diminue de moitié sans grand gaspillage. Hé hé ! Le clustering n'a pas encore dit son dernier mot… mais plus pour les mêmes raisons.

De fait, *par défaut, la FAT 32 utilise des clusters de 8 secteurs (4 kb)*.

Un coup d'œil au tableau précédent nous montre que la FAT 32 n'est pas intéressante sur des disques de moins de 256 Mb. Pourquoi ?

1 Il ne suffit pas de lire un cluster et de le mettre en mémoire centrale. Il faut aussi pouvoir utiliser l'information lue, c'est-à-dire atteindre chacun des caractères du cluster, ce qui impose de les numéroter. Or la numérotation utilise généralement un entier et on sait qu'un entier non signé (2 bytes) ne peut compter que de 0 à 65535, un entier signé ne peut compter que de -32768 à 32767, soit 32 kb

Parce qu'un disque de 256 Mb utilise des clusters de 4 kb aussi bien en FAT 16 qu'en FAT 32. Par contre, une FAT 16 prend deux fois moins de place qu'une FAT 32. La FAT 16 laisse donc plus d'espace utile.

La FAT 32 ne devient intéressante que pour les disques de 512 Mb et au-delà !

Le tableau ci-dessous reprend le clustering utilisé par la FAT 32 en fonction de la capacité du disque.

FAT 32		
Secteurs par cluster	Taille du cluster (kilobytes)	Capacité *minimale* du disque
8	4	256 Mb
16	8	8 Gb
32	16	60 Gb
64	32	2 Tb

Tableau 3-1

Faut-il pour autant convertir tous les disques de plus de 512 Mb de FAT 16 à FAT 32 ? Pas si sûr !

- si vous convertissez un disque FAT 16 en FAT 32, il n'est plus possible de revenir en arrière; sauf si on re-partitionne et reformate le disque. Des utilitaires tels que Partition Magic font du bon travail mais ne sont pas sûrs à 100% dans le sens 32→16;
- si votre disque est un modèle amovible qui doit fonctionner sur plusieurs machines, il est préférable de le laisser en FAT 16.
- certains utilitaires "disque" qui fonctionnaient en FAT 16 ne fonctionnent pas toujours en FAT 32.
- le temps d'accès à un secteur d'un disque FAT 32 est légèrement supérieur (2%) à celui d'un disque FAT 16.
- la défragmentation d'un disque FAT 32 peut prendre beaucoup plus de temps que celle d'un disque FAT 16 car il y a plus de secteurs à déplacer.

Bref, réfléchissez à deux fois avant de vous lancer et lisez d'abord le chapitre consacré au partitionnement.

3.5. Rôle des répertoires

3.5.1. Principe

Nous avons déjà touché un mot du répertoire au paragraphe 3.3.3. Nous l'avons comparé au registre des clients de l'hôtel ([1]).

Un répertoire (*ang.: directory*) n'est rien d'autre qu'un fichier qui contient un certain nombre de lignes ou enregistrements (*ang.: records*). En FAT16 comme en FAT32, toutes les lignes ont la même structure et une longueur de 32 bytes. Chaque ligne reprend les caractéristiques d'un et un seul fichier :

1 Depuis Windows 95, la notion de répertoire a été remplacée par celle de dossier (*ang.: folder*) ou de classeur. Les développeurs s'interrogent encore sur les raisons de ce changement de terminologie.

- nom du fichier (8 bytes)
- extension du fichier (3 bytes)
- attributs (1 byte)
- heure de création du fichier (2 bytes)
- date de création (2 bytes)
- date du dernier accès (2 bytes)
- heure de la dernière modification (2 bytes)
- date de la dernière modification (2 bytes)
- numéro du premier cluster occupé par ce fichier (2 bytes)
- taille du fichier (en bytes) (4 bytes)

soit un total de 28 bytes utilisés auxquels il faut ajouter 4 bytes gardés en réserve à toutes fins utiles.

Les **attributs** des fichiers sont une série de bits qui définissent les caractéristiques d'un fichier et l'usage que l'on peut en faire :

- lecture seule (*ang.: read only*)
- fichier système (*ang.: system file*)
- fichier caché (*ang.: hidden file*)
- fichier archivé (*ang.: archive*)
- répertoire (*ang.: directory*)
- long nom (*ang.: long filename*)
- etc.

Seuls les quatre premiers attributs peuvent être modifiés par l'utilisateur ([1]). Il convient cependant d'être vigilant quand on modifie ces attributs : un fichier marqué "système" + "caché" + "lecture seule" est certainement un fichier très important pour le bon fonctionnement du PC !

3.5.2. Arborescence

Un répertoire est un fichier comme un autre. Il peut avoir une taille quelconque, se trouver n'importe où sur le disque et contenir un nombre pratiquement illimité de références de fichiers.

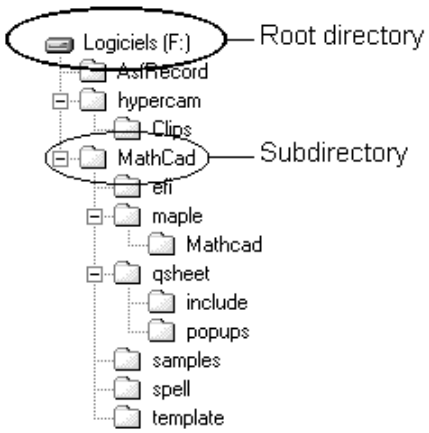


Figure 3.8 Arorescence de répertoires

1 Il est fréquent qu'un fichier transféré d'un CD-ROM vers un disque dur garde l'attribut "lecture seule". Pour y accéder, l'utilisateur devra modifier cet attribut à l'aide de la fonction "Propriétés" de l'explorateur Windows, soit à l'aide de la fonction ATTRIB du DOS.

Un répertoire peut lui-même contenir les références d'un ou plusieurs autres répertoires. On arrive ainsi à une structure en arbre avec un tronc unique appelé **répertoire racine** (*ang.: root directory*) sur lequel se greffent plusieurs branches qui elles-mêmes supportent d'autres branches appelées sous-répertoires (*ang.: subdirectory*).

3.5.3. Différences FAT16 / FAT32 et DOS / WIN

La gestion des répertoires en FAT16 et en FAT32 est fort semblable. Il n'y a qu'une petite différence.

En **FAT 16**, le système d'exploitation se réserve 32 secteurs juste derrière les deux copies de la FAT afin d'héberger le répertoire racine (*ang.: root directory*). La taille fixe de cette zone fait que le répertoire racine est limité à 512 entrées. *Autrement dit, on peut mettre un* ***maximum de 512 fichiers ou dossiers*** *dans le répertoire* ***C:*** même si le disque dispose de la place nécessaire.

En **FAT 32**, le répertoire racine n'a pas de place bien déterminée. Il peut se trouver n'importe où sur le disque en fonction des besoins du système d'exploitation. Il est donc géré (presque) comme n'importe quel autre répertoire et *peut contenir* ***plus de 512 fichiers***.

La gestion des répertoires en DOS et en Windows est aussi fort semblable. Ici aussi, il n'y a qu'une différence. Elle concerne les **longs noms de fichiers** (*ang.: long filenames*).

Depuis Windows 95, l'utilisateur peut utiliser jusqu'à 255 caractères pour nommer ses fichiers et répertoires. La contrainte 8.3 du DOS a été levée. Les noms peuvent aussi contenir des espaces, des signes de ponctuation, etc.

Toute cette "nouvelle" information est aussi conservée dans le répertoire décrit plus haut. Mais comment conserver un nom de 255 caractères dans une structure qui n'admet que 32 bytes par record ? Tout simplement en utilisant plusieurs records successifs.

Chaque fois qu'un nom de fichier compte plus de 8 caractères, par exemple "`lettre à belle-maman pour annoncer notre divorce.doc`", le système crée ([1])

- un record "normal" avec un nom DOS de 8 caractères forgé à partir du nom long. Par exemple : "`lettre~1.doc`".
- un record de 32 bytes avec un caractère spécial et le début du nom long
  Par exemple : "`#lettre à belle-maman pour annon`"
- autant de records que nécessaire pour la suite du nom long, toujours avec un caractère spécial. Par exemple : "`#cer notre divorce`"

Avec cette astuce, les répertoires conçus par Win95 (et suivants) restent lisibles en DOS : les programmes Windows utiliseront le nom "long"; par contre, la plupart des programmes DOS trouveront le nom DOS et, étant incapables d'interpréter les records suivants, ils les ignoreront simplement. Il reste néanmoins possible d'écrire un programme DOS capable de traiter les noms long ([2]).

1 Attention, il s'agit ici du principe. La réalité est un peu différente notamment en ce qui concerne la valeur et la position du caractère spécial ainsi que le codage du nom long.

2 C'est notamment le cas de DOSPRMPT.EXE qui ouvre une fenêtre DOS sous Windows. Quand on y tape la commande DIR, la programme affiche le nom DOS 8.3 et le nom long "Windows"

Le principal inconvénient est qu'une ancienne version DOS de SCANDISK.EXE détectera des lignes incompréhensibles dans les fichiers-répertoires. En corrigeant cette "erreur" il supprimera les noms longs. Il en ira de même avec une ancienne version DOS de DEFRAG.EXE. Donc n'utilisez pas ces utilitaires sur des disques qui contiennent des noms longs ([1]).

Notez bien que cette gestion des noms longs ***ne dépend pas*** de la structure de la FAT mais uniquement du système d'exploitation.

3.5.4. Création et suppression de fichiers

Lorsque l'on crée un fichier sur le disque, nous avons vu que le système explore la FAT et attribue les secteurs. Ensuite, il remplit la structure décrite ci-dessus et ajoute cette ligne dans le fichier-répertoire ad hoc.

A l'inverse, lorsque l'on supprime un fichier, nous savons que le système se contente d'inscrire un 0 dans les éléments correspondants de la FAT mais qu'il n'efface pas les secteurs qui étaient occupés par le fichier effacé.

De manière similaire, le système n'efface pas les informations contenues dans le répertoire : il se contente de remplacer le premier caractère du nom de fichier par la valeur ***conventionnelle*** $E5_h$ (caractère ASCII "σ").

Ceci permet de récupérer un fichier que l'on vient d'effacer par erreur à l'aide d'un utilitaire tel que UNDELETE. Les chances de succès sont excellentes si on agit sans tarder et si le disque n'est pas trop fragmenté ([2]).

3.6. Autres systèmes de fichiers

Nous avons principalement parlé des systèmes de fichiers FAT16 et FAT32 utilisés par les systèmes d'exploitation DOS, Windows 95 / 98 / Me / 2000 / NT et OS2.

Les informaticiens, rarement à court d'imagination, en ont inventé d'autres que nous présentons brièvement ci-dessous. Il n'est malheureusement pas possible d'entrer dans les détails et nous conseillons au lecteur de faire ses propres recherches sur Internet pour en savoir plus.

3.6.1. VFAT

La Virtual FAT est une version plus sécurisée de la FAT. Ce système est utilisé par Windows 95. La différence principale est le support des longs noms de fichiers qui ne sont plus limités au format 8.3.

3.6.2. NTFS

NTFS (New Technology File System) est utilisé par **Windows NT et 2000**. Sa structure principale est la "table de fichiers maître" (*ang.: Master File Table* ou *MFT*), d'une robustesse industrielle, mais qui utilise beaucoup d'espace disque. NTFS conserve plusieurs copies de la partie la plus importante de la MFT, par mesure de sécurité en cas de corruption ou de perte de données.

---

1 Pour éviter tout problème, sous Win 95 et suivants, les fichiers indispensables au systeme sont encore toujours nommés selon la convention 8.3.

2 Il ne faut pas confondre l'envoi d'un fichier dans la corbeille et sa suppression. L'envoi dans la corbeille est un simple changement de répertoire qui ne modifie ni la FAT ni l'espace disponible sur le disque tandis que la suppression (vider la corbeille) libère les secteurs et augmente l'espace disponible.

Le système vérifie en permanence le bon état du disque dur, détecte automatiquement les secteurs défectueux et déplace leur contenu vers des secteurs sains. Il est (théoriquement) capable de gérer des fichiers de plusieurs millions de gigabytes.

La taille des clusters NTFS est beaucoup plus petite qu'en FAT. A titre d'exemple, un disque de 2 Gb formaté en NTFS utilise des clusters de 2 kb contre 4 kb en FAT 32 et 32 kb en FAT 16. L'utilisation de petits clusters réduit non seulement la quantité d'espace disque gaspillé, mais aussi la fragmentation des fichiers. Grâce à sa capacité à utiliser des petits clusters, NTFS est très performant sur les disques très volumineux.

Contrairement aux systèmes FAT qui utilisent une structure linéaire des répertoires, NTFS utilise un arbre binaire, beaucoup plus efficace quand il s'agit de retrouver un fichier.

Chaque fichier d'un disque NTFS est géré comme un objet : il se voit attribuer une série d'attributs tels que les données qu'il contient, les informations de sécurité, le nom du fichier, l'icône, etc. En particulier, NTFS permet de définir des ressources (fichiers, hardware,…) qui dont l'accès est limité à certains utilisateurs en fonction du groupe auquel ils appartiennent : certains n'ont aucun accès à la ressource, d'autres peuvent lire le document mais pas l'éditer, d'autres encore ont les droits de lecture, d'édition et d'effacement (CRUD = create, read, update, delete).

On voit que NTFS est surtout conçu pour des systèmes à usage intensif tels que des serveurs de fichiers fort sollicités. Il n'est certainement pas recommandé pour des disques de moins de 400 Mb car ce système utilise une grande quantité d'espace pour la structure du système.

3.6.3. HPFS

HPFS (High Performance File System) est utilisé par OS/2 (IBM) et les anciennes versions de Windows NT.

Ce système est rapide, bien organisé et attribue plus judicieusement l'espace disque. L'allocation se fait secteur par secteur. Les secteurs ne sont pas organisés en clusters mais bien en **bandes**, c'est-à-dire en paquets de 8 Mb situés sur une même piste. Les tables d'allocation sont des zones de 2 kb situées entre les bandes. De ce fait, les têtes de lecture ne doivent pas (ou presque pas) se déplacer pour retrouver la position d'un fichier et lire son chaînage. C'est beaucoup plus efficace.

3.6.4. Unix & Linux

Unix et Linux, son dérivé pour PC, sont des systèmes d'exploitation qui mériteraient un cours à eux seuls. Nous ne donnerons ici qu'un très bref résumé de leur système de fichiers.

Le système de fichiers d'Unix s'articule en plusieurs niveaux.

Au niveau supérieur, on trouve les **disques**.

Les disques sont segmentés en **partitions** de tailles différentes. Chaque partition a une fonction spécifique définie en fonction des besoins de l'administrateur.

Chaque partition contient le **système de fichier**, c'est à dire les répertoires (*ang.: directories*), sous-répertoires (*ang.: subdirectories*) et fichiers (*ang.: files*). Jusque là, rien de bien différent de ce que nous avons rencontré en FAT 16 et 32.

Les informations relatives à chaque fichier sont contenues dans les **inodes**. On y trouve notamment la taille du fichier, la référence du début de fichier et les autorisations. Les inodes sont en relation avec les **directories**, qui, elles, conservent les noms des fichiers.

Les informations concernant les caractéristiques physiques du disque, l'adresse de inodes, la liste des blocs libres sont stockées dans les **superblocks**.

Quant aux fichiers, ils peuvent être de plusieurs types, chaque type ayant une fonction particulière : fichiers "normaux", hard links, symbolic links, sockets, named pipes, character devices, block devices.

Nous invitons le lecteur intéressé par ce sujet à consulter un ouvrage spécialisé.

Exercices du chapitre 3

♦ Exercice 3.1

Calculez la capacité du disque qui a les caractéristiques suivantes :

16 têtes, 63 secteurs, 1024 cylindres.

♦ Exercice 3.2

En examinant la FAT de la Figure 3.5 du paragraphe 3.4.4, que pouvez-vous déduire au sujet des secteurs hachurés ?

Après écriture du fichier UVWXYZ, comment feriez-vous pour stocker un fichier de 2096 bytes dans les secteurs restants ?

♦ Exercice 3.3

Si vous formatez le disque de l'exercice 3.1 en FAT 16, quelle sera la taille des clusters ?

Quelle sera la taille de la FAT ?

Est-il intéressant de le formater en FAT 32 ?

♦ Exercice 3.4

Vérifiez que la taille maximale de la FAT 32 d'un disque de 2 Tb est de 16 Gb.

♦ Exercice 3.5

A l'aide d'un petit programme VB ou C, essayez de créer ou de copier plus de 512 fichiers dans le répertoire racine d'un disque FAT 16.

Recommencez l'essai dans un sous répertoire.

Refaites l'expérience avec un disque FAT 32.

♦ Exercice 3.6

Détaillez la suite des opérations effectuées par le système d'exploitation (lecture de la table de partition, de la FAT, du répertoire racine, etc.) pour accéder au fichier

```
D:\Cours\PC\Supports\syllabus.doc
```

qui occupe 189 kb sur un disque de 800 Mb formaté en FAT 32 ?

Application du chapitre 3 : installer un disque dur

♦ Objectifs

- Identifier les connecteurs d'un disque dur et les connecteurs correspondants d'une carte mère.
- Connecter correctement un disque dur en primary master.
- Définir le disque dur au niveau de l'architecture de la machine (setup).
- Booter sur une disquette système afin de tester le fonctionnement et l'accès au disque dur
- Ajouter un disque dur à un système existant (primary slave, secondary master ou secondary slave).

♦ Ingrédients

- Un disque dur de type IDE, d'une capacité de 200 Mb à 1200 Mb que l'on peut se procurer à bon compte dans une brocante informatique.
- Un ruban de connexion IDE.
- Un PC équipé au moins d'un lecteur de disquette.
- Une disquette DOS bootable contenant les principaux utilitaires.
- Un tournevis cruciforme.

♦ Branchement

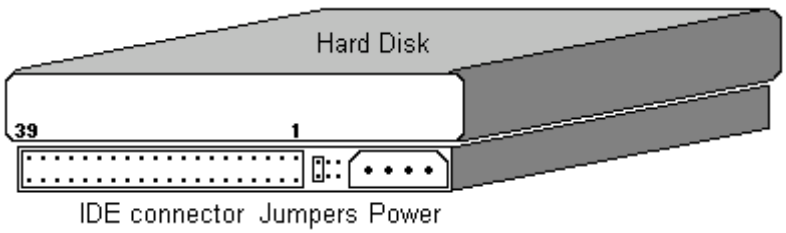
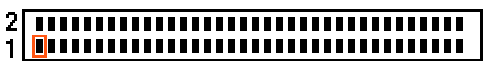


Figure 3.9

- Repérer les "**jumpers**" du disque dur. Les jumpers sont de petits cavaliers rectangulaires dont la position définit le type de branchement : maître (*ang.: master*) ou esclave (*ang.: slave*). Ils sont généralement placés à côté du connecteur IDE, parfois de l'autre côté du disque, plus rarement sur la face inférieure du disque, sur le circuit imprimé.

  Examiner soigneusement l'étiquette collée sur la face supérieure du disque dur. Elle explique comment placer le ou les jumpers. Cette position varie selon les constructeurs et les modèles de disque. Placez-les en position "**master**".
- Repérer le **connecteur IDE** du disque dur. Le connecteur IDE est formé de 40 broches (*ang.: pins*). Par convention, la broche [1] est (presque) toujours située du côté de l'alimentation (*ang.: power*). Parfois, le chiffre 1 apparaît sur le circuit imprimé qui forme la face inférieure du disque.



- Repérer le fil rouge du ruban de connexion. Le ruban de connexion est formé d'une nappe de 40 fils. Le premier (ou dernier) fil est marqué en rouge.

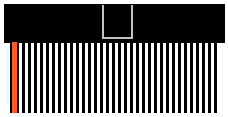


Figure 3.10 Connecteur IDE

Insérez le dans le connecteur de manière à placer le fil rouge du côté [1].

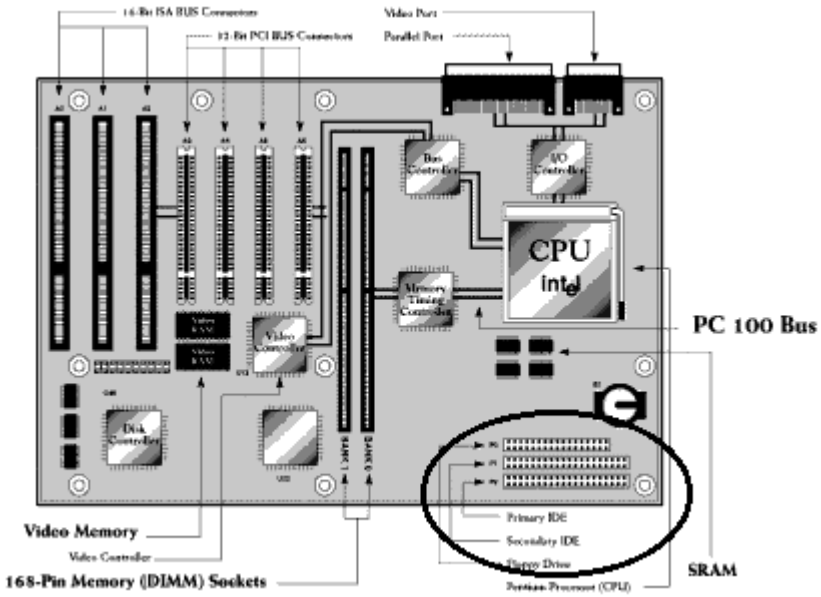


Figure 3.11 Schéma de la carte mère

- Arrêtez le PC, enlevez son capot et couchez le PC sur le côté de manière à accéder facilement à la carte mère.
- Repérer les **connecteurs IDE de la carte mère**. Il y a généralement trois connecteurs : deux connecteurs de 40 broches destinés aux disques durs et un plus petit, de 34 broches, destiné au lecteur de disquette.

  Les connecteurs IDE sont nommés "**Primary IDE**" ou "IDE1" et "**Secondary IDE**" ou "IDE2". Chaque connecteur peut se raccorder à deux disques durs, lecteur ou graveur de CD.

  Repérer la broche marquée [1]. Si aucun chiffre n'apparaît sur le circuit imprimé, repérez la broche dont la soudure a une base carrée.

  Si d'autres disques durs sont déjà branché, repérez le côté du fil rouge puis déconnectez-les tous (mais pas le lecteur de disquette) afin d'éviter de les endommager par une mauvaise manœuvre.

  Insérez le câble venant du nouveau disque dans le connecteur "Primary master" de manière à placer le fil rouge du côté de la broche [1].
- Récupérez un **câble d'alimentation interne**. Les câbles d'alimentation internes sont des câbles à 4 fils (jaune / noir / noir / rouge) qui se terminent par une fiche mâle ou femelle à 4 broches.

  Insérez la fiche d'alimentation dans la broche correspondante du disque dur. Du fait de sa forme, la fiche ne s'insère que dans un seul sens.
- Posez le disque dur sur le chassis du PC. Au besoin, isolez-le avec une feuille de papier. Vous le fixerez définitivement quand vous aurez vérifié son bon fonctionnement.

♦ Test du disque

- Insérer la disquette bootable **DOS** dans le lecteur.
- Démarrer le PC. Lorsqu'une ligne du type

  `Hit DEL to enter SETUP` ou **`Hit F1 to enter SETUP`**

  apparaît, tapez la touche ad hoc pour entrer dans le programme de configuration (SETUP). Vous trouverez plus de détails sur le **SETUP** dans un autre chapitre du cours.
- Le setup diffère d'un constructeur à l'autre. Les deux grandes familles sont **AWARD** et **AMI**. Malgré une présentation différente, on y retrouve les mêmes fonctions. L'exemple ci-dessous est repris d'un setup Award.

  Sélectionnez l'option qui permet de définir le disque dur :

  `STANDARD CMOS SETUP`

et introduisez les caractéristiques du disque que vous venez de connecter.

Si le setup le permet, utilisez la commande d'autodétection de disque :

IDE HDD AUTO DETECTION

- Toujours dans le setup, indiquez au système qu'il doit booter sur la disquette **A:** et non sur le disque dur **C:**. On trouve ce paramètre dans

  BIOS FEATURES SETUP

- Quitter le setup en sauvant les informations.
- Laisser au système le temps de redémarrer.

  Vérifiez qu'il détecte bien le disque que vous venez de connecter. Il affiche un message du type

  Detecting HDD primary master : WDC2420

  avec la référence du disque dur. Ensuite il explore la disquette, démarre le système DOS et affiche le prompt DOS

  A:\>

- Lancez **FDISK**.

  A:\> FDISK

- Vérifiez la présence du nouveau disque avec l'option 4

Si tout a bien fonctionné, vous pouvez à présent réaliser les application suivantes afin de partitionner et formater ce nouveau disque. Sinon, passez à la section dépannage.

♦ Dépannage

Reprenez la procédure en vérifiant chaque étape :

- Le câble IDE est-il bien enfoncé dans ses connecteurs ?
- Est-il connecté dans le bon sens au disque et à la carte mère ?

  Essayez éventuellement avec un autre câble ou en permutant les extrémités disque et carte mère.

- Les jumpers sont-ils positionnés correctement ?

  Essayez éventuellement la position "slave"

- Les jumpers ont-ils bien une lamelle métallique intérieure ?
- Tous les autres disques et lecteurs CD sont-ils bien débranchés ?
- Le disque est-il alimenté correctement ?

  Essayez éventuellement de reprendre l'alimentation d'un des disques ou lecteurs de CD que vous avez déconnecté.

- Le setup a-t-il été bien défini et sauvegardé ?

  Essayez la définition manuelle si vous aviez choisi la définition automatique ou vice-versa.

  Vérifiez que les paramètres encodés correspondent bien à ceux de l'étiquette du disque.

- La pile du CMOS est-elle toujours bonne ?

  Quand la pile est à plat, le système oublie la date et l'heure entre deux redémarrages. S'il oublie la date, il risque aussi d'oublier la configuration du système. Changez éventuellement la pile.

- Le setup est-il configuré pour booter sur la disquette au lieu du disque dur ?
- La disquette est-elle bien bootable ?

- S'agit-il bien d'une disquette bootable "DOS" et non d'une disquette de démarrage "Windows" ?

  Vérifiez en bootant un autre PC.

- **FDISK** est-il bien présent sur la disquette ?

Si le disque n'est toujours pas détecté, reprenez la manipulation avec l'un des disques qui se trouvaient déjà sur le PC.