System 1ere année 2007 – 2008

1 Introduction : L'ordinateur et ses composants.

OS (Operating System) : ensemble de programme permettant la gestion des différentes ressources de l'ordinateur.

Un ordinateur permet d'exécuter **UN** programme (une instruction a la fois.)

Le pc est composé de :

- 1 RAM :
  - Ensemble de « Mots » adressables.
  - On peut y lire un mot à une adresse donnée.
  - Contiens des instructions et des données que l'on peut utiliser.
  - Mémoire volatile.
  - le temps d'exécution est la même peu importe ou tu est.

RAM et CPU utilisent un bus d'adresse et un bus de données pour communiquer.

- 2 CPU : Interprète et exécute une instruction a chaque cycle.

- Dans le CPU :

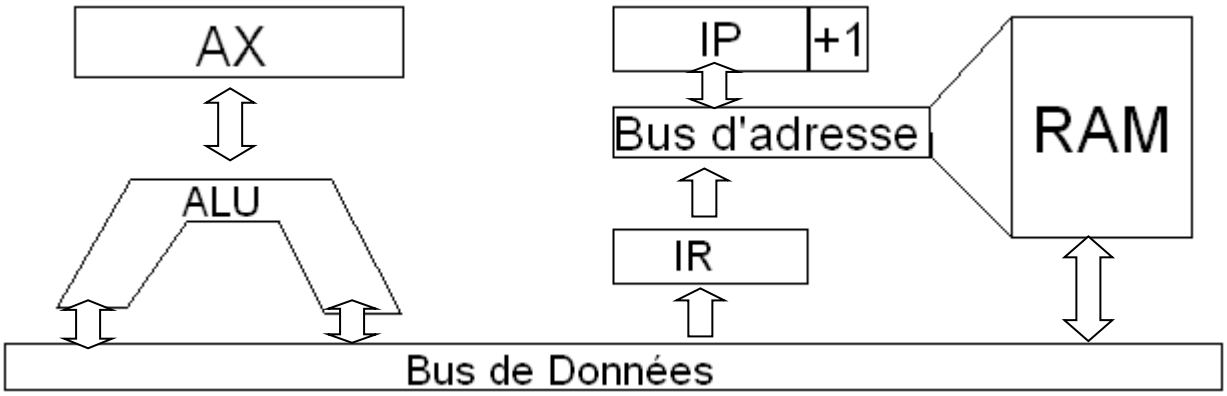
Registres : Zone de stockage (très petite) inclue dans le CPU et très rapide.

Unité de commande : Amène et demande aux ALU l'instruction.

- ALU : (Arithmetic logic unit) : s'occupe des opérations logique et arithmétique (calcul)
- UC : (Unité Centrale) : Elle est chargé d'extraire de la mémoire l'instruction courante, la décodé et l'exécuter (faisant appelle a ALU)
  - Registre d'instruction (RI) : Instruction à exécuter.
  - Registre IP (Instruction pointeur ou compteur cordial (CO)) : Contiens l'adresse mémoire de l'instruction suivante a exécuté.
  - Décodeur d'instruction : détermine l'opération à exécuter et les opérandes
  - Séquenceur : Synchronise les différentes parties du CPU.
  - Horloge : Synchronise les actions de L' UC.

http://fr.wikipedia.org/wiki/Processeur

Schéma



Principe d'exécution d'une instruction.

Top 1 : L'adresse contenue dans IP passe dans le bus d'adresse et va en RAM.

Top 2 : IP reçoit l'adresse +1 et l'instruction, puis l'instruction va dans RI via le bus de données.

Top 3 et suite : Exécution de l'instruction.

MOV AX, [100h]

Cette instruction MOV met dans l'AX le contenus de l'adresse 100h de la RAM

Hypothèse de départ

- IP vaut 300h
- En 300h se trouve l'instruction MOV AX, [100h] (codé : 8B 0100h)
- En 100h se trouve les caractères : ABCD.

Top 1 : 300h est envoyé sur le bus d'adresse vers la RAM.

Top 2 : IP reçoit 301h et « 8B 0100h » va dans RI via le bus de donnée. 8B est traduit par le décodeur donc RI enclenche top 3.

Top 3 : 100h est envoyé sur le bus d'adresse vers la RAM.

Top 4 : La RAM renvois ABCD sur le bus de donnée vers L'ALU qui l'envoi dans AX.

Instruction JMP

Hypothèse de départ.

- IP vaut 400h
- En 400h se trouve l'instruction « JMP 4567H » (codé par exemple B9 4567h)
- En 4567h il doit y avoir une instruction !

Top 1 : 400h est envoyé à la RAM via le bus d'adresse.

Top 2 : IP reçoit 401h et B9 4567h est envoyer a RI par le bus de donnée. B9 est traduit en : « JMP »

Top 3 : 4567h va dans IP par le bus d'adresse. IP est bien mise pour exécuter l'instruction 4567h.

Le chargeur.

Le chargeur (loader) charge le code binaire d'un programme en mémoire.

Le chargeur est un programme… mais qui charge le chargeur ?

- Le bootstrap ! (se trouve dans la ROM de la carte mère ?)

Vrais au faux :

- On peut exécuter un programme qui ne réside pas en mémoire. F
Il doit d'abord être chargé par le chargeur pour être exécuté.

- Le registre IP contient toujours l'instruction à exécuter. F
Il contient l'adresse de la prochaine instruction à exécuter.

- Exécuter l'instruction MOV AX [100h] qui met dans AX le contenu de l'adresse 100 demande de lire 2x en RAM. V

- JMP 500h, à l'adresse 500h peut se trouver la donnée ABCD. F
En 500h dois se trouvé une instruction et non une donnée.

JMP 400h, a l'adresse 400h peut se trouvé l'instruction JMP 500h. V

2 Emergence et évolution des system d'exploitation.

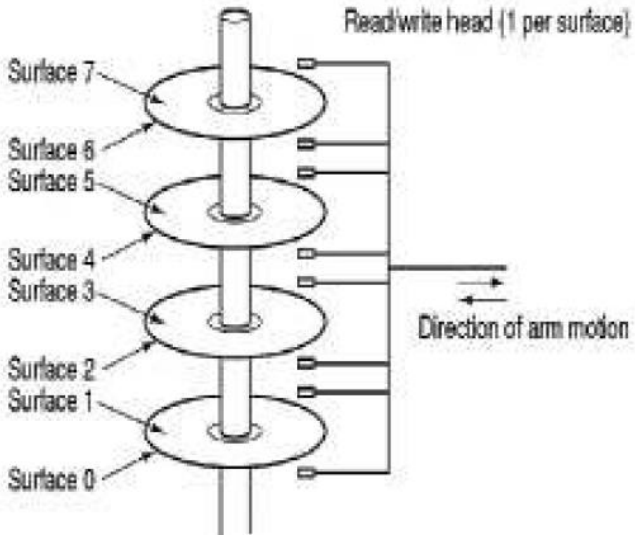
Au début (Année 50) : il n'y avait pas de OS (que des circuits et puces exécutant la même tache.)

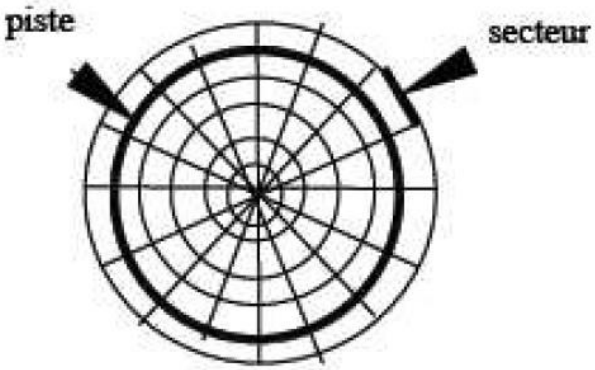
De nos jours : Ordinateur = Machine virtuelle.

Le disque dur :

Le disque dur est composé de plateaux divisé en Cylindre. Ses cylindres sont divisés en Pistes elle même divisé en Secteur.

Schéma :





Piste : Ligne fictive tracé sur une bande ou un disque magnétique par une tête de lecture-écriture.

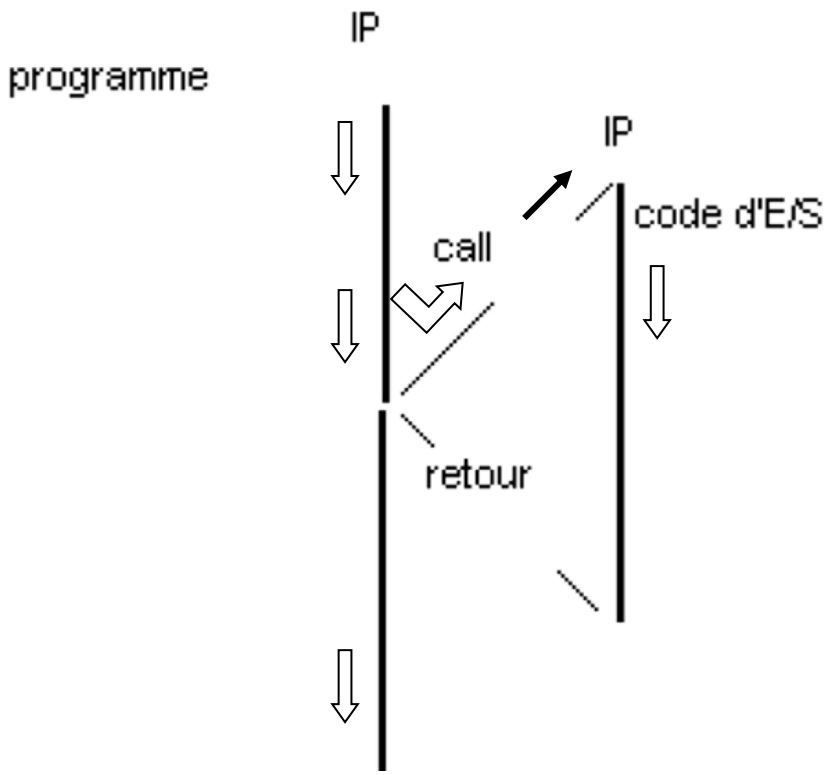
Secteur : Enregistrement physique constitué d'un ensemble contigu de mots situé sur une piste du disque.

Cylindre : dans une pille de disque, ensemble de pistes qui peuvent être lues sans que intervienne un mouvement de tête de lecture-écriture.

Instruction CALL

Permet d'exécuter un bout de code situé ailleurs et de revenir ensuite au code appelant (appel de fonction).

Schéma :



L'instruction RET permet de retourné au programme en cour.

Hypothèse de départ

- IP vaut 432h.
- En 432h se trouve CALL 724h.

(Idem que le JMP sauf que IP est sauvé dans AR et donc a l'exécution de la commande RET → IP revient a 433h.)

Après exécution du CALL :
- AR vaut 433h (IP<-IP+1)
- IP vaut 724h
Les instructions en 724h et suivantes (bout de code) sont exécutées jusqu'à l'instruction **RET** :
- IP <- AR (=433h)
Après exécution du RET : IP vaut 433h
Le programme continue de s'exécuter et récupère la donnée lue

Un code système peut en utiliser un autre

Le mécanisme décrit plus haut ne permet qu'un niveau d'appel (un seul registre AR)
Sur Intel, pour permettre plusieurs niveaux d'appels : on dispose d'une : **pile** des adresses de retour elle se trouve dans une zone de mémoire pointée par le registre SP (Stack Pointer) **pointeur de pile**

Monoprogrammation.

Mode d'exploitation d'un ordinateur dans lequel UN et UN seul programme est exécuté jusque au bout dans la mémoire principale. ➔ Enchainement rapide des programmations.

Processeur canal *(DMA : Direct Memory Access).*

Ce sont des petits processeurs en dehors du CPU (northbridg, soutbridg,…)

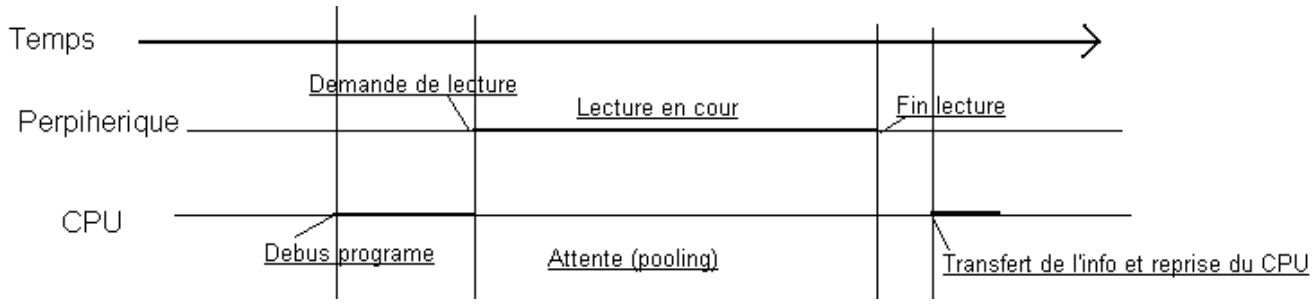
Canal : Processeur d'entrée-sortie mettant en relation la mémoire d'un systeme informatique et un ou plusieurs organes périphériques.

Des processeurs secondaire (Canaux) vont décharger le CPU des E/S (Entrées / Sorties) et ils ne s'occupent que de ça. Le processeur canal est connecté aux bus, à la RAM et au CPU. Il peut écrire l'info des périphériques ou on lui dit de l'écrire : « lis autant et met le a tel adresse. »
Donc on diminue le temps CPU dépenser a géré les E/S et on peut lire plus de chose car la RAM est bien plus grande que la mémoire tampon.

Travail batch : permet d'enchainé les programmes plus rapidement : on écrit un petit programme qui dit quel programme doit être lancé dans la suite.

La vitesse des périphériques est beaucoup plus lente que celle du CPU donc quand les périphériques font une lecture ou une écriture, le CPU est ralentis ou en attente.

Schéma : (slide 28)



Les interruptions :

- Une alternative au polling → on utilise une interruption pour signaler la fin d'une E/S.
- L'interruption arête le CPU des que un événement spécial intervient.
- C'est une interruption hardware. Une interruption intervient quand il y a du courant sur un fil spécial.

Lors d'une interruption :

- Arrêter le programme en cours
    1. Sauvegarder l'état du CPU
    2. Exécuter le code de service de l'interruption.

- Rétablir l'état de la machine.
- Reprendre l'exécution du programme interrompu (si possible).

[IP + registre]

1) Le CPU sauvegarde IP sur la pile (mémoire).
2) Le CPU change IP à la valeur de l'adresse du code du traitement de l'interruption (software).
3) Traitement de l'instruction.
4) La dernière instruction de ce code restore la valeur d'IP (RET).

(Comme un CALL)

1) le programme fait une interruption INT (logiciel).
2) Le processeur canal est activé dans le code de l'interruption.
3) Le programme boucle jusqu'à ce que le processeur canal finisse.
4) Le E/S finit → l'interruption continue de ce déroulé → fin de l'interruption → le programme continue.

Questions :

- un appel système provoque toujours une interruption. V
- une interruption empêche toujours le processeur de terminer l'instruction en cours. F

Car on exécute toujours une instruction jusqu'au bout.

Mode privilégié :

Quand une interruption est lancée, le flag est changé et l'interruption passe en mode privilégié. Il repassera en mode non-privilégie à la commande IRET.

La multi programmation.

- Le CPU est libre pendant que il y a une lecture ou écriture sur un périphérique d'entrée / sortie (il boucle en attendant la fin du périphérique) on va donc donner « la main » a d'autre programmes.

Processus : programme chargé en mémoire en attente de son exécution.

Le problème c'es le partage de la mémoire :
Il ne faut pas que le 1 programme est touché aux données du 2emme.
Les registre AR s'occupes de sécurisé la mémoire.

Partage du CPU :
L'ordonnanceur s'occupe de donné la main au programme en attente.

Il donne la main à l'autre programme soit quand le programme en cour fait une E/S ou quand le temps alloué est dépassé.

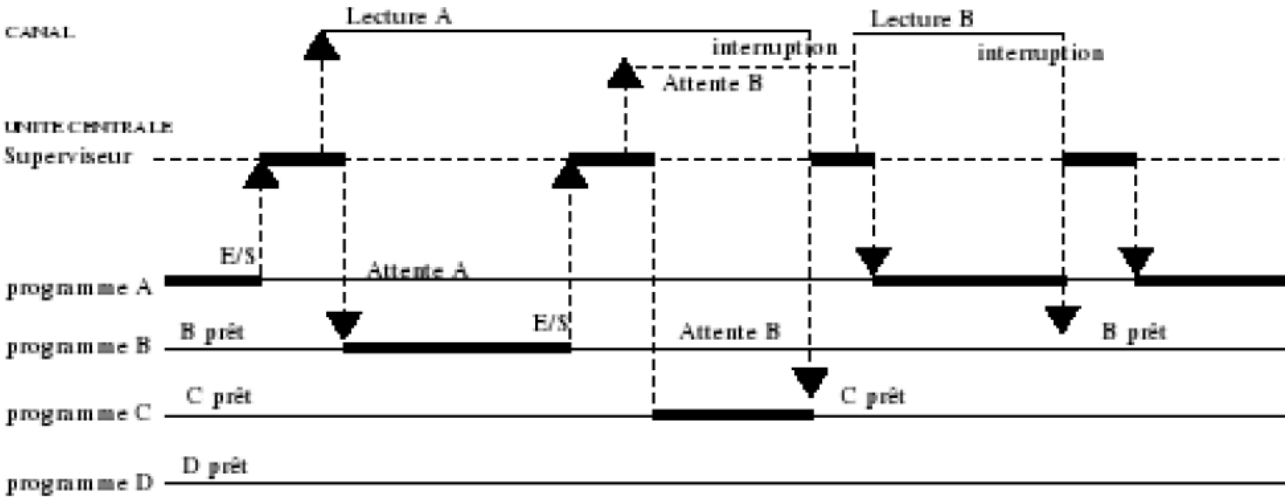
Les programmes (processus) peuvent être dans différents états :

1. prêt : attend le CPU.
2. élu : utilise le CPU.
3. bloqué : attend un événement externe (lecture de donnée par exemple)

EX :

- P1 est élu, P2 est en prêt.
- P1 fait une interruption pour demander une lecture sur périphérique
- Le processeur exécute le code de l'interruption voit que c'est une lecture et lance le canal → on sauve IP de P1 sur la pile.
- P1 passe a bloqué et donne la main a ordonnanceur.
- L'ordonnanceur donne la main à P2 qui passe élu.
- Le CPU exécute les intrusions de P2 pendant que le canal fait une lecture pour P1.

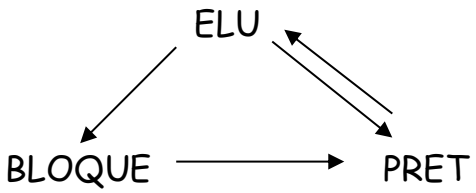
EX 2 :



Time slicing :

Pour éviter qu'un processus qui ne fait pas d'E/S et donc garde le CPU indéfiniment, après un temps T, le system d'exploitation lance une interruption pour obliger le programme à prendre la main.
L'ordonnanceur est donc appelé par l'interruption horloge et a le choix du prochain programme à exécuter.
Çà minimise les temps de réponses de l'ensemble des programmes mais gérer les changements de contextes occupent du temps CPU.

Transition possible



Gestion des ressources :

Ressources : un « truc (cd-rom, RAM,…) » dont un processus a besoin pour s'exécuter.

L'inter - blocage.

Concerne l'utilisation de ressources pour éviter les inter-blocages, il y a des ressources partageables et non partageable.

Partageables :

1. Ecran
2. Disque dur
3. Baffles.

4. …

Non - partageables :

1. CPU
2. Imprimante
3. Graveur
4. …

Une ressource qui ne peut être retiré a un processus tant que il n'a pas finit de l'utilisé est dite non – préemptible.

Préemptible : on peut retirer la ressource au processus car il prend trop de temps.

Pour que un inter - blocage ait lieu, il faut :

- Plusieurs ressources. .
- Des ressources non partageables.
- Des ressources non – préemptibles.

Exemple d'inter-blocage :

- P1 grave un CD et P2 imprime un document.
- P2 a besoin de donnée se trouvant sur le graveur, il attend que P1 ait finit.
- P1 a besoin d'imprimé avent de terminé sa gravure P1 attend que P2 ait finit l'impression.

- Il y a donc un inter-blocage car les 2 processus attende chacun la fin de autre.

Pour prévenir un inter-blocage sur imprimante, on utilise le spooler qui est un processus system qui envoie les fichiers en entier à l'imprimante.
Ses fichiers sont stockés dans des buffers propres à l'imprimante.

3 System de fichier.

Répertoire : Fichier qui permet de retrouvé tous les autres qu'il contient. (/home/mba/texte). C'est l'opération open qui ouvre le fichier et mémorise son chemin lu dans la table descripteur.

Table descripteur : Contient le début du fichier ouvert, la position courante du fichier ouvert et sa longueur. Elle se trouve dans la RAM du processus (cache).

Allocation de l'espace disque au fichier.

Il y a 2 types d'allocation :

1. Allocation contiguë.
2. Allocation non contiguë (sur base d'un index ou enchaînement de blocs).

Allocation contiguë : on retrouve un fichier, mais en connaissent son début et sa longueur.

Le problème :

On veut agrandir un fichier, mais si il est entouré de part et d'autre part d'autres fichiers, il faut déplacer des fichiers pour créer de la place : opération lourde (à cause de la fragmentation externe).

Fragmentation externe : on coupe un fichier en plusieurs parties et on le place dans les trous.

Fragmentation interne : les fichiers sont plus petits que les clusters.

Allocation non contiguë :

1. Par bloc et index : Les blocs sont numéroté et une table d'index permet de reconstitué les fichiers.
   parcourir un fichier = parcourir les blocs indiquer par la table, à partir du premier bloc du fichier.

   Les blocs : c'est la taille minimum d'un bloc de donnée alloués sur le disque (Bloc = Cluster).

2. Par blocs chainés : au lieu d'utiliser une table d'index, les blocs sont enchainés entre eux [7] → [8] → [9] → [21] → [1]
   0 … 7 8 9 … 21

   Avantage : pas de table d'index dans la RAM mais il faut savoir où se trouve le 1.

Les allocations par blocs (non contigus) finissent par générer des fichiers forts fragmentés.

Fonction principale d'un répertoire :

Fournir les informations pour trouver les blocs de disque. Il fait le lien entre le nom du fichier et son emplacement. (En Windows, les données d'un répertoire sont stockées sous forme d'un fichier)

Il traduit le nom du fichier en son emplacement.

Fragmentation externe :

Apparait sur le disque quand on supprime des fichiers, il y apparait des trous.
Schéma

Fragmentation interne :

Apparait quand la taille des blocs est trop grande et qu'ils ne sont jamais remplis jusqu'au bout → perte de place.

3. FAT : File Allocation Table

La fat c'es une zone chargé au debus en ram qui contien l'adresse des clusters des fichiers.

Allocation de l'espace par blocs avec index

- Blocs = clusters numérotés à partir de 0
- La table d'index = la FAT, là ou l'emplacement des clusters se trouvent
- La taille des secteurs et des clusters est définie au formatage.
  Un cluster = n secteurs

Schéma : Structure d'une partition FAT

1) Zone réservée (boot sector 1secteur pour FAT 12-16, 32 pour FAT 32)
2) Zone FAT (fat et copie de la fat)
3) Zone du répertoire racine (pas pour FAT 32)
4) Zone pour les fichiers et répertoires

Le Boot Sector :

Il est tout d'abord mis en RAM (par le chargeur) et éxécuté.
L'info qu'il contient :

- Instruction de saut au programme d'amorce
- Taille d'un secteur
- Nombre de secteurs d'un cluster
- Nombre de FAT
- Nombre d'entrées dans le répertoire racine (nombre de fichier sur le disc) (pas pour FAT 32)
- Nombre de secteurs dans la FAT
- Nombre total de secteurs

La FAT : Table

La FAT et sa copie éventuelle

- Décrit

Manque un cours je pense

Les entrées de la FAT ont une taille fixe :

- FAT 12-12 bits (par cluster) → MAX 2^12 = 4096 clusters
- FAT 16-16 bits (par cluster) → MAX 2^16 65536 clusters
- FAT 32-28 bits utiles (par cluster) → MAX jusqu'à 2^{28-1} clusters

La FAT est chargée en RAM par des raisons de performances.

Appel Système de la FAT :

Schéma

Pour effectuer un OPEN, il faut passer tous les paramètres d'ouverture par registres → il faut donc les vider, c'est donc un appel système comme un INT.

Appel système READ :

Pour lire 271 bytes du fichiers n°7,le programme doit :

- Compléter le registre 1 avec le numéro du fichier (soit 7).
- Compléter le registre 2 avec l'adresse de la mémoire ou les 271 bytes sont écrits, soit un nom de variable.
- Compléter le registre 3 avec le nombre de bytes à lire, soit 271.
- INT → interruption avec le processeur CANAL.
- …
- Une fois réclu : lire le nombre de bytes réellement lus dans le registre 4 (au cas où on tomberait sur la fin du fichier).

4. NTFS

Par rapport à la FAT, on travaille avec des blocs plus réduits

VCN (Virtual Cluster Number): numéro d'ordre dans le fichier. (Premier cluster du fichier, deuxième…)
LCN (Logical Cluster Number): numéro d'ordre du cluster au sein de la partition. Numéro secteur ou démarre un cluster sur la partition.

LCN→ LCN * n°sect. / cluster
VCN→ adresse spécifique située dans le DATA RUN (information stockée et qui indique le chaînage du fichier)

Structure d'une partition NTFS

En NTFS, toute information est stockée sous forme de fichier. Les métadonnées du système de fichiers sous forme de fichiers système.

- Fichiers des blocs libres
- Fichiers des blocs défectueux
- Fichier « répertoire racine »
- Fichier MFT : le plus important (une entrée par fichier du système)

→NTFS retrouve les données des fichiers via la MFT
Schéma

Le fichier BOOT :

Seul fichier à emplacement fixe : secteur de démarrage, 0
16 secteurs d'informations

1) $MFT
2) $MFT ???
3) . racine
4)
5)
6)

7)
8)

Zone MFT :

Sa taille peut être configurée au départ.
Son but est de maintenir le fichier $MFT non fragmenté

Chaque fois que le reste du disque se remplit → sa taille diminue de moitié. La place que prends la MFT = 12,5 % de la taille du disque

Exemple
100 Gb → zone MFT de 12,5 Gb

Si on a plein de petits fichiers → vaut mieux prendre une grande MFT (pour tout stocker dedans).
Si on a de grands fichiers → petite MFT pour la liaison de la place dans la zone fichier.

$MFT est le plus important des fichiers de métadonnées entrée de taille variable : une entrée pour chaque fichier.

…. ?

L'entrée de la MFT pour un fichier contient :

- Une partie des métadonnées (sous forme d'attributs).
- Parfois les données du fichier (petits fichiers : attribut data).
- A défaut, leur localisation sur clé disque (LCN : attribut data non résident)

Les répertoires permettent de localiser les fichiers en renseignant leur entrée MFT.

La MFT n'est pas stockée en RAM

Les entrées de la MFT + schéma

Chaque enregistrement à une taille fixe. Un enregistrement se compose comme ceci :
Schéma

Exemple :

[EN TETE] Standa…. ???

En tête :

- Flug type d'enregistrement
- Flag d'enregistrement libre
-
-

Les attributs : chacun a une taille variable → pour s'y retrouver, il faut que chaque attribut spécifie sa propre taille dans l'en-tête qui lui est propre.

- Si le fichier est petit : $DATA le contient et le flag « résident » est mis à 1 (résident = le fichier se trouve en MFT).

- Si le fichier est grand : $DATA ne contient que son en-tête, le flag « résident » sera à 0 et un chainage de blocs sera indispensable (qui sera dans le DATA-RUN).

DATA-RUN : liste de blocs

Appel système : interruption logicielle : fait sur demande du prog : demande d'E/S