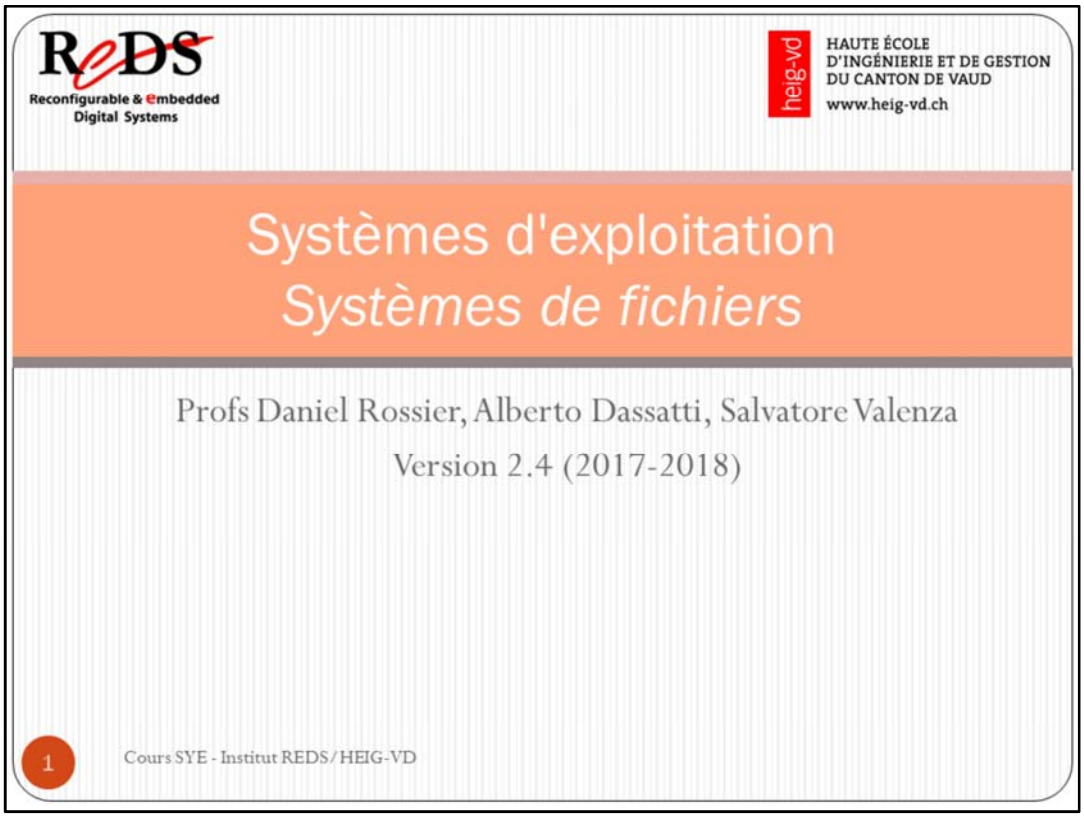
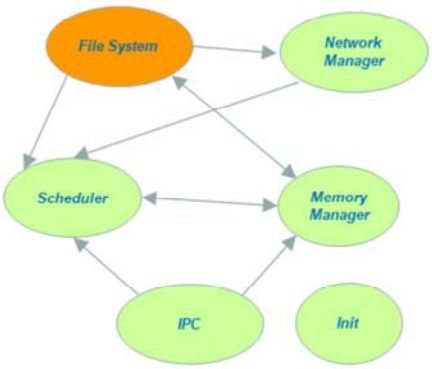
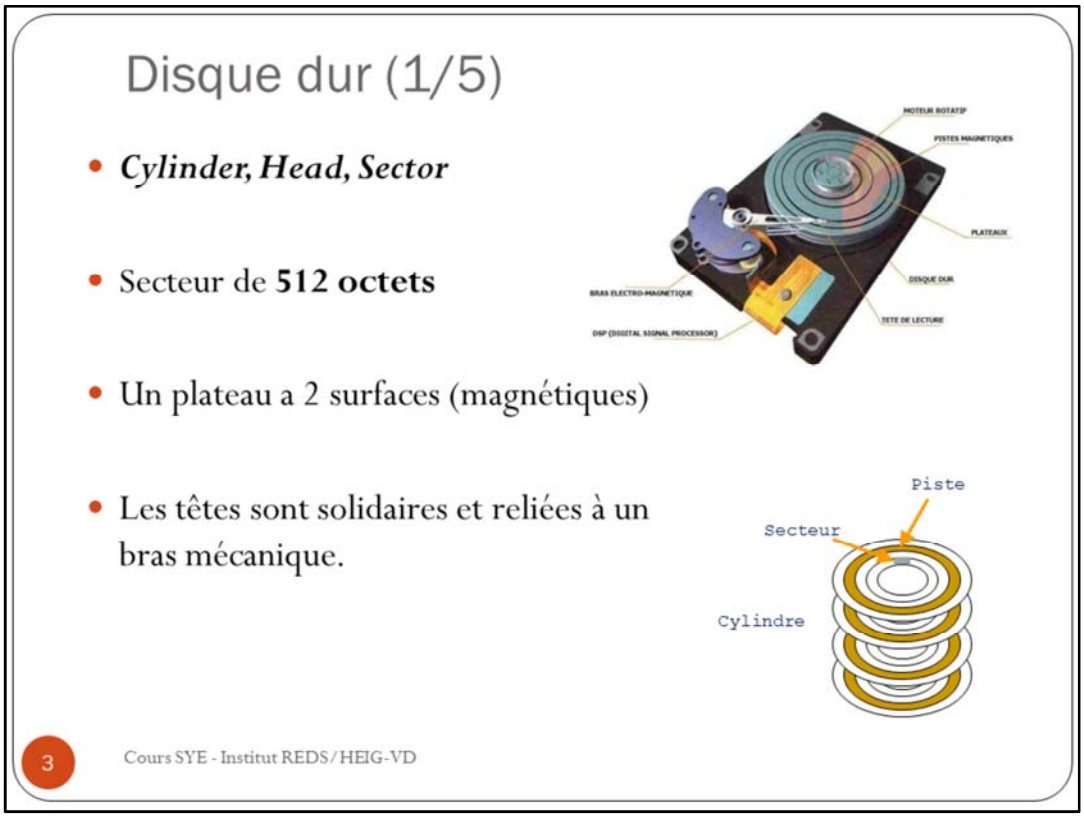

ReDS
Reconfigurable & embedded Digital Systems
heig-vd
HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch
Systèmes d'exploitation
Systèmes de fichiers
Profs Daniel Rossier, Alberto Dassatti, Salvatore Valenza
Version 2.4 (2017-2018)
1
Cours SYE - Institut REDS/HEIG-VD

Plan

- Disque dur
- Architecture et partitions
- Allocation contiguë
- Allocation par listes chaînées
- Allocation indexée





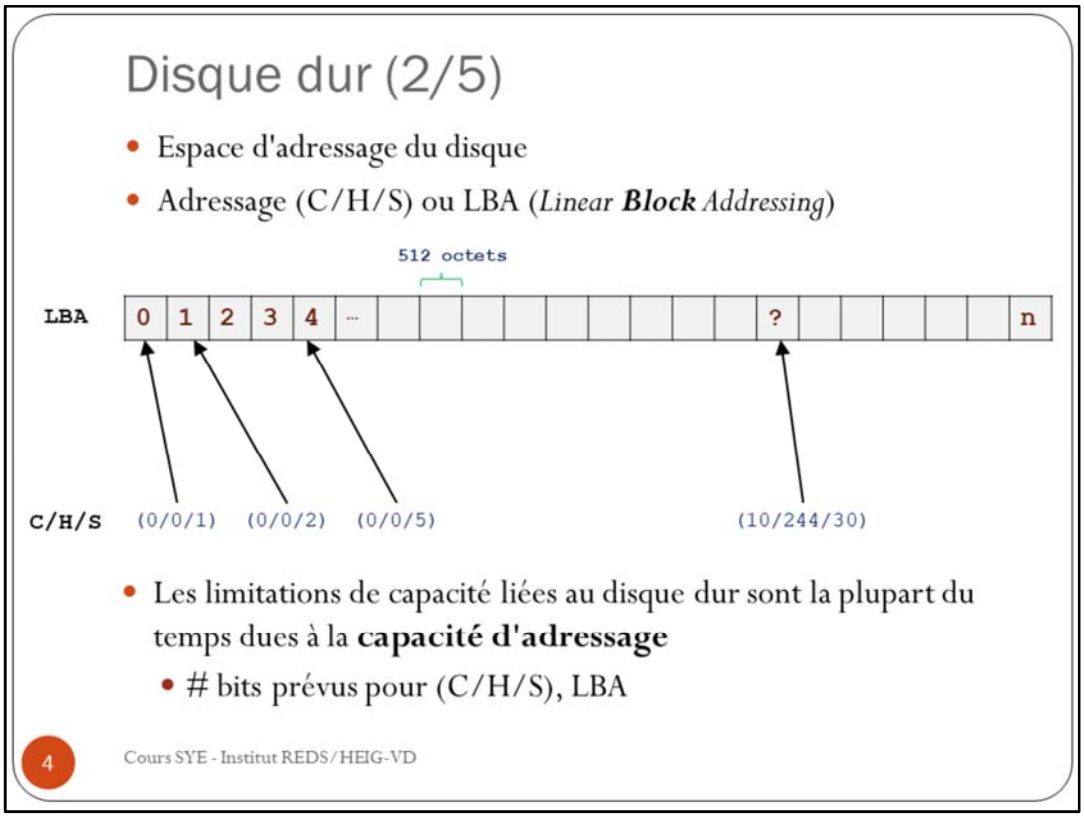
Un disque dur est une collection de plateaux – aujourd'hui typiquement entre 2 et 4 pour un disque 3"½ – sur lesquels apparaissent deux surfaces magnétiques (une dessus, une dessous). Les plateaux sont solidaires et tournent autour d'un même axe de rotation à une certaine vitesse (7'200 tours/minute par exemple).

La surface magnétique est balayée par une tête de lecture/écriture, et l'ensemble des têtes sur chaque surface se déplace de manière *solidaire*.

Une surface magnétique contient un ensemble de **secteurs** disposés le long d'une piste. Un secteur est un bloc de 512 octets et constitue la granularité la plus fine au niveau de l'adressage d'un disque. Une surface comporte donc plusieurs pistes concentriques dont le nombre de secteurs varie en fonction du rayon de la piste. Sur chaque surface magnétique, on retrouve la même disposition des pistes et des secteurs.

Si l'on considère l'ensemble des pistes de même rayon sur toutes les surfaces magnétiques (des différents plateaux), on obtient un **cylindre**.

Un disque dur sera donc caractérisé par sa ***géométrie physique***: un ensemble de cylindre (**C**), un ensemble de têtes (*heads*) (**H**), et un ensemble de secteurs par piste (**S**). La géométrie physique (**C/H/S**) n'a également un sens **que si le nombre de secteurs par piste** est identique. On verra par la suite que cela n'est aujourd'hui plus le cas.



La géométrie physique d'un disque dur a conditionné la manière dont on adresse les octets sur celui-ci. Comme vu précédemment, il n'est pas possible de référencer un et un seul octet à la fois, mais seulement **512 octets** via la notion de **secteur**. Il s'agit donc de savoir comment identifier les différents secteurs d'un disque dur.

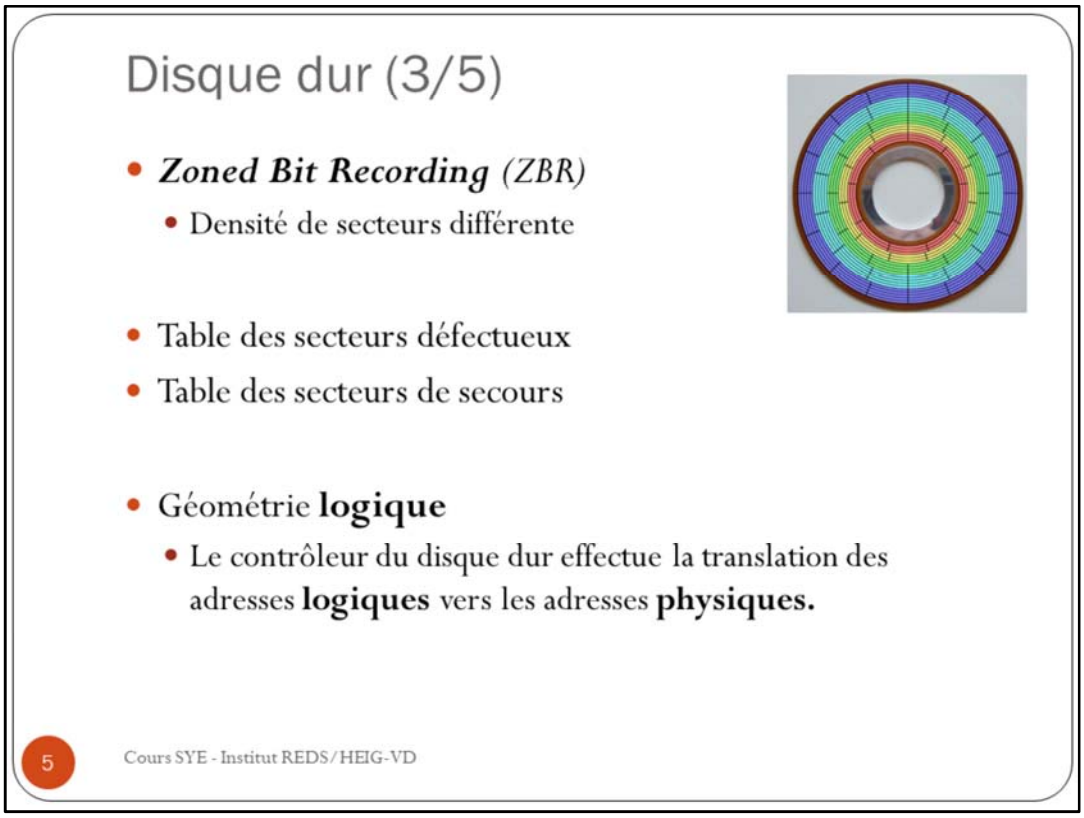
L'adressage d'un secteur passe d'abord par l'identification du ***cylindre***. Chaque cylindre – dont le rayon varie - est numéroté. Un cylindre représentera une collection importante de secteurs, puisqu'il s'agit de l'ensemble des pistes d'un même rayon sur tous les plateaux, i.e. sur toutes les surfaces magnétiques.

Une fois le cylindre identifié, il s'agit de déterminer la piste contenant le secteur référencé. Cela est possible en "désignant" la ***tête*** de lecture/écriture qui balaye la piste concernée. En identifiant le numéro de tête, la piste est donc sélectionnée.

Finalement, il reste à identifier le numéro de ***secteur*** à l'intérieur de la piste.

Grâce à la combinaison (**cylindre, tête, secteur**) – ou (**c, h, s**) chaque secteur d'un disque dur devient *référençable*: on parle d'adressage de **type (c/h/s)**.

Une autre manière d'adresser les secteurs d'un disque dur est de faire abstraction de sa géométrie, et de les considérer de manière **linéaire**. Ainsi les secteurs sont numérotés (de manière unique) de 0 à **n,** n représentant le dernier secteur adressable. Ce type d'adressage est le type **LBA** (*Linear Block Addressing*).



Le passage d'une adresse de type *(c/h/s)* et une adresse de type *LBA* s'effectue simplement. Il nécessite en revanche de connaître la géométrie du disque (C/H/S).

Conversion (c/h/s) en LBA

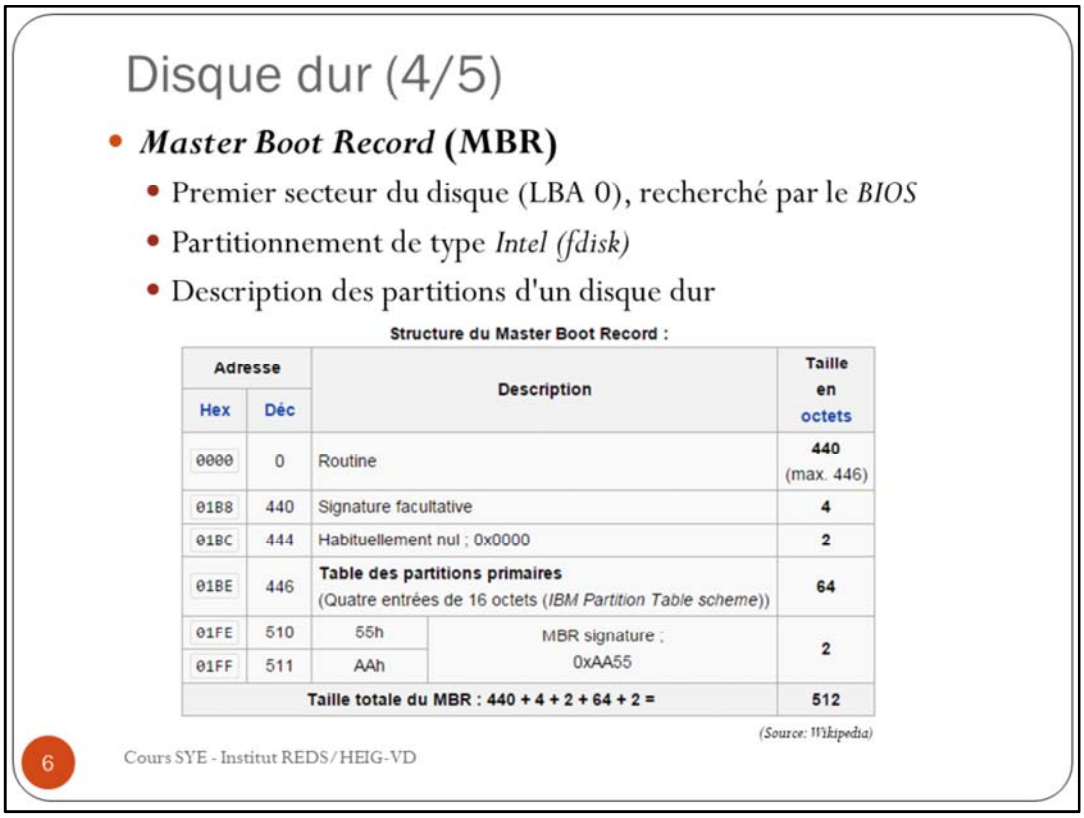
- LBA = $c * H * S + h * S + s - 1$

Conversion LBA en (c/h/s)

- $s = LBA \,\%\, S + 1$
- $h = ((LBA - s + 1) / S) \,\%\, H$
- $c = (LBA - s + 1) / S) \,\mathrm{div}\, H$

Le passage d'un type d'adresse à l'autre montre bien qu'il y a une équivalence entre les deux: l'adressage de type LBA ne résout donc aucun problème lié à une quelconque limite d'adressage; c'est simplement un type d'adressage simplifié.
La variation de densité de secteurs en fonction du rayon des pistes rend l'utilisation de la géométrie physique difficile. C'est pourquoi, on considère aujourd'hui une géométrie ***logique***; c'est le contrôleur disque qui se chargera ensuite de faire la traduction d'une adresse logique en une adresse physique.

Par convention, la géométrie logique d'un disque s'exprime en fixant le nombre de têtes à **255**, et le nombre de secteur à **63**. Seul le nombre de cylindre varie. Par exemple, une géométrie de (*12'300*/**255**/**63**) correspond à un disque dur d'une capacité de 12'300 * 255 * 63 * 512 octets.



Disque dur (4/5)

- ***Master Boot Record* (MBR)**
  - Premier secteur du disque (LBA 0), recherché par le *BIOS*
  - Partitionnement de type *Intel (fdisk)*
  - Description des partitions d'un disque dur

Structure du Master Boot Record :

Adresse		Description		Taille en octets
Hex	Déc			
0000	0	Routine		**440** (max. 446)
01B8	440	Signature facultative		**4**
01BC	444	Habituellement nul ; 0x0000		**2**
01BE	446	**Table des partitions primaires** (Quatre entrées de 16 octets (*IBM Partition Table scheme*))		**64**
01FE	510	55h	MBR signature ; 0xAA55	**2**
01FF	511	AAh		
Taille totale du MBR : 440 + 4 + 2 + 64 + 2 =				**512**

(Source: Wikipedia)

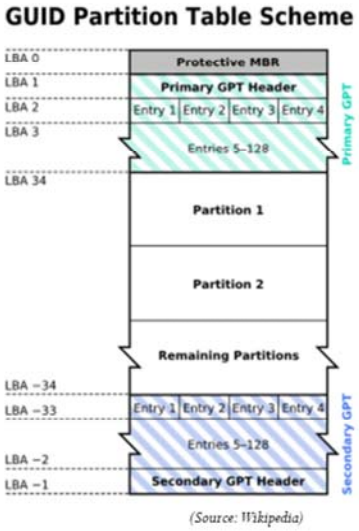
6 Cours SYE - Institut REDS/HEIG-VD

Le *Master Boot Record* est le premier secteur d'un disque dur qui contient les informations relatives au partitionnement de celui-ci. Le format du *MBR* a été spécifié par *Intel* et rend possible un partitionnement avec quatre partition au maximum (quatre partitions primaires ou trois partitions primaires et une partition étendue permettant de définir des *sous-partitions*).

Le *MBR* doit également contenir le premier code d'amorçage du système exécuté par le *BIOS.* Le code doit tenir dans 440 octets au maximum; c'est très peu. C'est la raison pour laquelle, ce code réalise un branchement vers un code plus évolué pouvant se trouver à l'intérieure ou à l'extérieure d'une partition.

Disque dur (5/5)

- ***GUID Partition Table* (GPT)**
- Standard UEFI *(Unified Extensible Firmware Interface)* proposé par *Intel*
- Nombres de partitions (primaires) très élevées
- Entrée protectrice de MBR

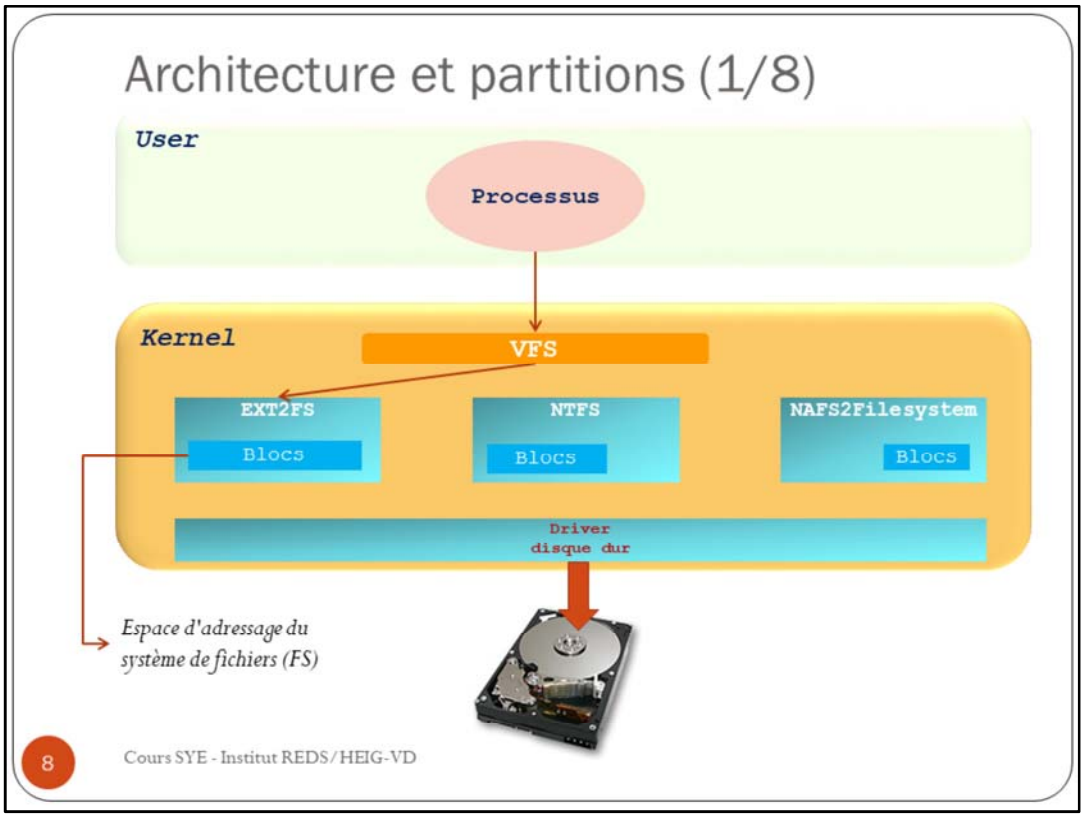


7 Cours SYE - Institut REDS/HEIG-VD

Les limitations du *MBR* ont conduit *Intel* à spécifier un nouveau standard associé à un nouveau *firmware* en remplacement du *BIOS* : le *Unified Extensible Firmware Interface (UEFI)*. Ce standard définit une nouvelle approche pour décrire l'ensemble des partitions sur un disque dur en offrant la possibilité de créer un nombre bien plus élevé de partitions (128 partitions gérables par *Windows* par exemple).

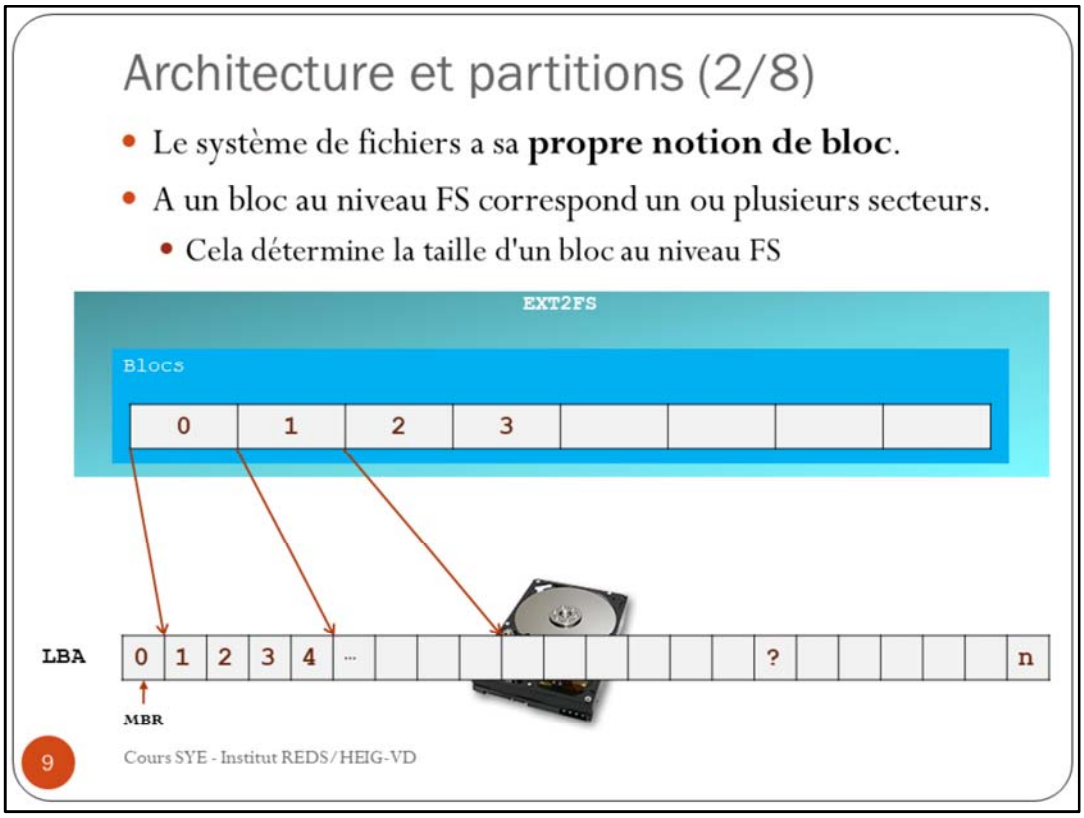
GPT gère les disques durs et partitions jusqu'à 9,4 Zo (9,4 x 10^{21} octets ou 9,4 trilliards d'octets soit 9,4 * 10^{9} To ou **2^{73} octets**)

Le premier secteur est appelé *Protective MBR* et contient un MBR valide spécifiant une seule partition pour la totalité du disque, mais au max. 2.2 To, pour les *BIOS* qui ne reconnaîtraient pas un partitionnement GPT.



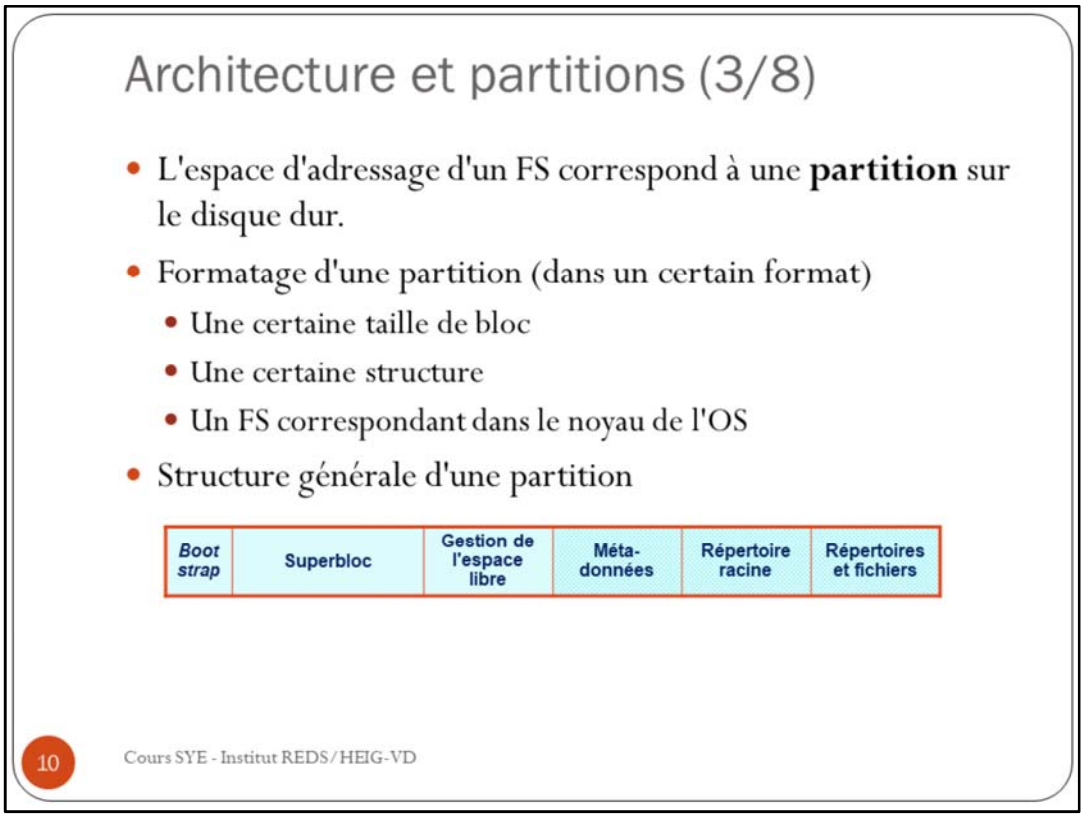
La couche *VFS (Virtual File System*) permet de s'abstraire du type de système de fichiers, du point de vue d'un processus. Ce dernier peut alors utiliser les appels systèmes habituels (*open(), write(), read(), etc.*) indépendamment du type de stockage.

La couche *VFS* peut ainsi gérer plusieurs types de systèmes de fichiers. L'opération de montage permettra alors d'accéder au contenu géré par un système de fichier. Grâce à ce mécanisme, plusieurs systèmes de fichiers peuvent coexister simultanément et le processeur peut accéder aux fichiers de l'un ou l'autre de manière totalement transparente.



La notion de bloc au niveau du système de fichiers (bloc FS) n'est pas équivalente à celle de secteur ou bloc LBA d'un disque dur. Le bloc FS est généralement supérieur à 512 octets et représente la plus petite unité adressable au niveau du système de fichiers, à l'instar de la notion de page mémoire dans un système paginé. Cette comparaison avec le gestionnaire mémoire amène naturellement à considérer des **blocs FS de 4 Ko**; ainsi, la taille d'une page mémoire et celle d'un bloc FS étant identique, le transfert de données entre mémoire et système de fichiers (et donc le disque dur) s'en trouvera grandement optimisé. La taille du bloc est déterminée lors du formatage de la partition (formatage de haut niveau) et ne change plus. Les transferts entre disque dur et système de fichiers s'effectueront généralement par tranche de 4 Ko; l'utilisation d'antémémoire dans le contrôleur disque permet de lire et d'écrire plusieurs secteurs de 512 octets à la fois.

Si des transferts de blocs de 4 Ko peut s'avérer optimaux dans la plupart des cas, il peut arriver toutefois qu'une taille de bloc supérieur à 4 Ko soit plus avantageuse; c'est le cas, par exemple, pour le traitement de très gros fichiers de type multimédia. Dans ce cas, il s'avère préférable de stocker ces fichiers dans une partition formatée avec des blocs de plus grande taille (32 Ko ou 64 Ko); les systèmes de fichiers de type FAT se prêtent bien à ce type d'utilisation comme nous le verrons plus loin.



La partition représente l'espace d'adressage tel que perçu par le système de fichiers. La plus petite unité adressable est le bloc, comme vu précédemment.

Une partition est généralement organisée en différentes zones comme le montre la figure ci-dessus.

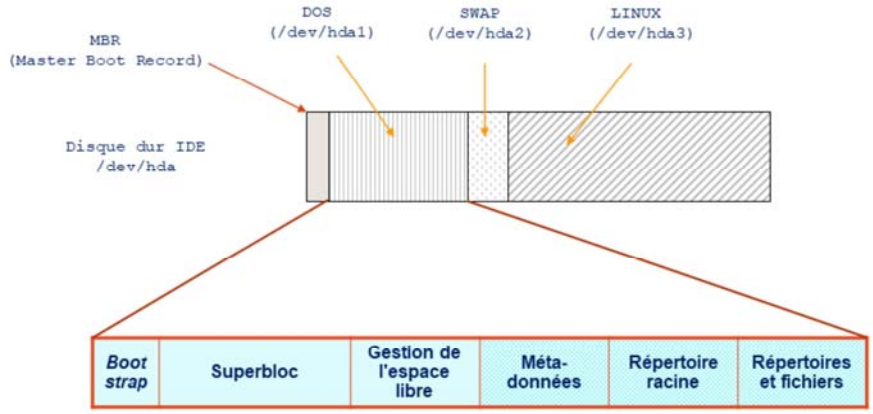
Le superbloc contient les informations de type *administrative* sur la partition (taille, début/fin de secteur, type, pointeur vers le répertoire racine, etc.).

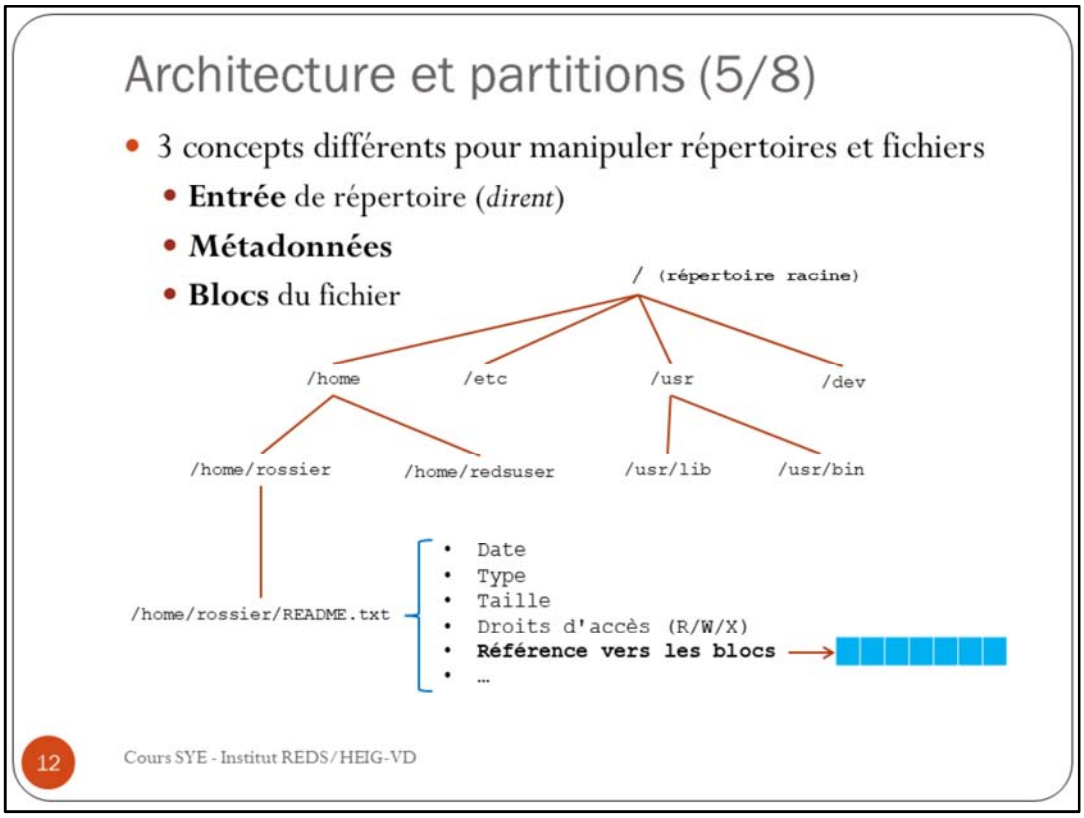
La gestion de l'espace libre consiste à stocker de manière permanente une structure de données permettant de garder la trace des **blocs libres** de la partition. Cette structure est typiquement un tableau de bit (*bitmap*) ou une liste chaînée, à l'instar de ce qu'utilise le gestionnaire mémoire pour la gestion des pages libres.
La structure de données utilisée pour gérer les blocs libres représente une zone de données critiques car elle renseigne sur la place disponible d'une partition. Une corruption de cette zone retournera des informations erronées, en faisant croire à l'utilisateur, à tort, qu'il dispose encore de la place ou au contraire que le la partition est pleine. C'est pourquoi, les outils de vérification et réparation de la cohérence de la partition sont généralement capables de régénérer une nouvelle structure en *scannant* l'intégralité de la partition et en examinant les différentes tables d'allocation.

Le répertoire racine constitue le point d'entrée de l'arborescence de fichiers.

Architecture et partitions (4/8)

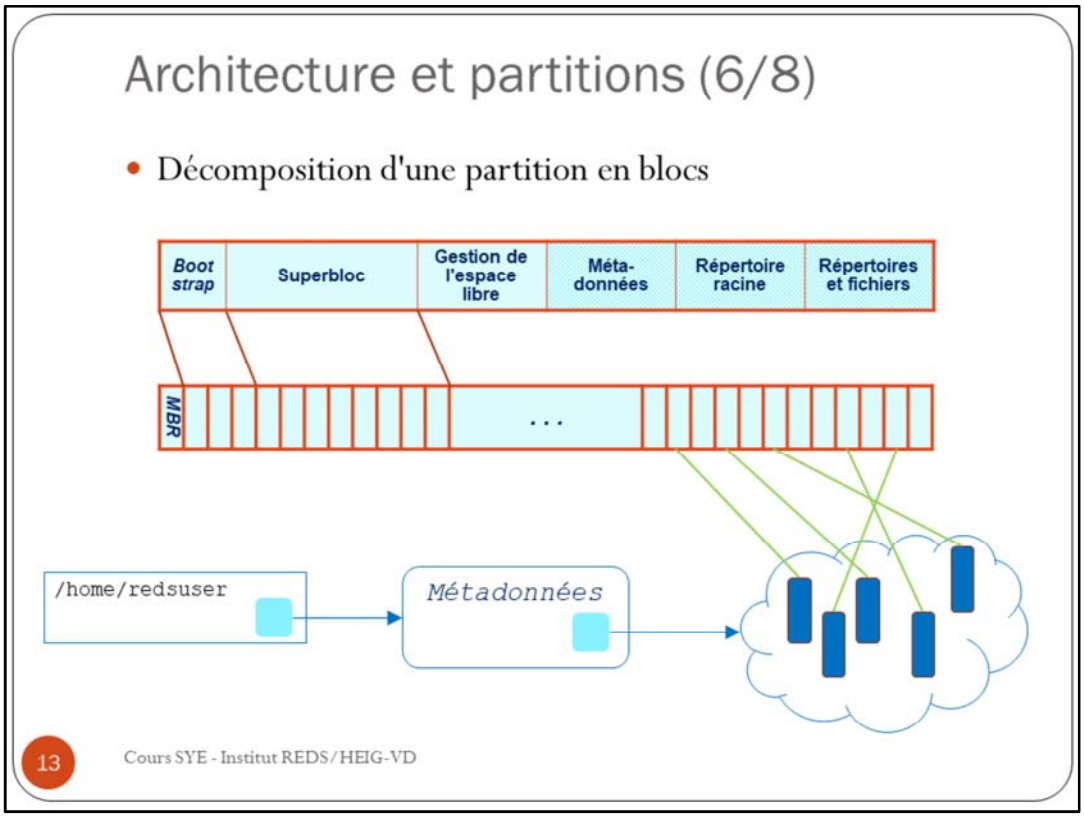
- Le disque peut stocker plusieurs partitions
  - Plusieurs systèmes de fichiers





Une entrée de répertoire consiste en une chaîne de caractères (ASCII) permettant l'identification d'un fichier, à partir de laquelle il sera possible de retrouver l'ensemble de ses données (métadonnées et données utilisateur).

Sous une forme générale, l'entrée de répertoire contient une référence vers les métadonnées du fichier qui, à leur tour, possèdent les informations nécessaires à l'identification des blocs de la partition appartenant à ce fichier.



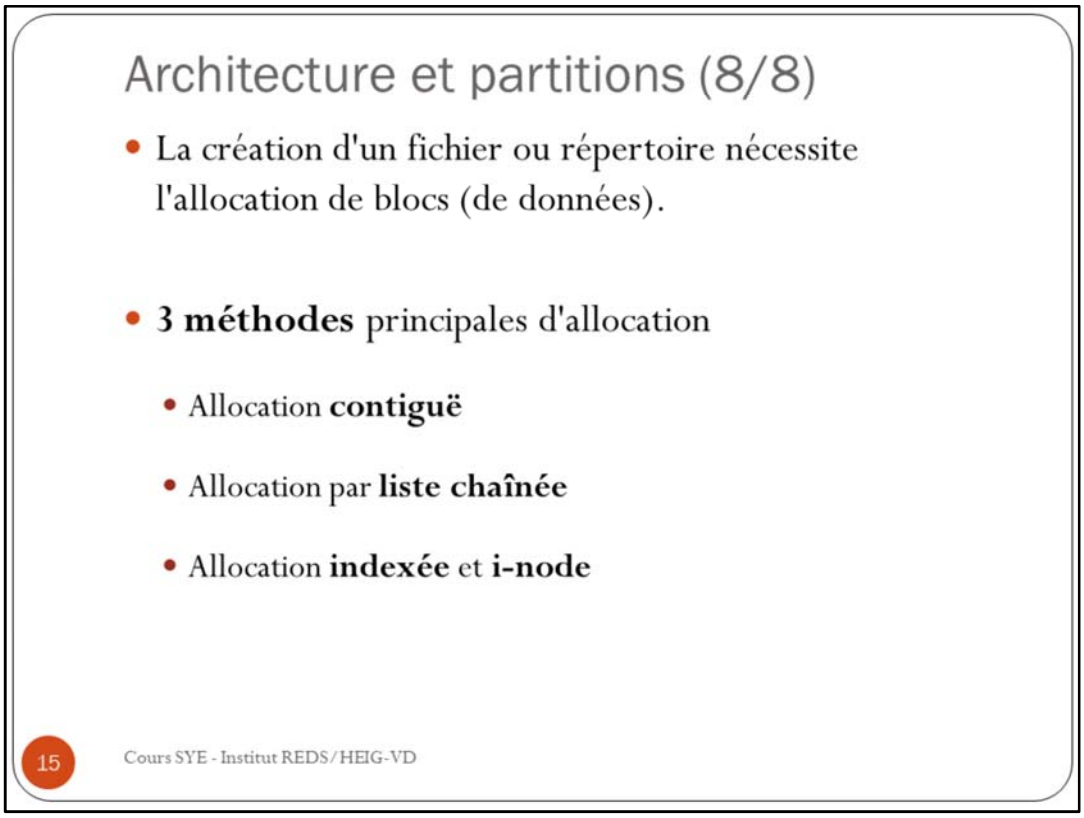
La figure ci-dessus met en évidence la décomposition d'une partition en blocs de taille normalement équivalente à la taille d'une page mémoire (soit 4 Ko).

Comme décrit précédemment, l'accès aux blocs des fichiers s'effectue à partir des entrées de répertoire et au travers des métadonnées.

Architecture et partitions (7/8)

- Soit un système de fichier utilisant des blocs de 4K. Le bloc contenant les métadonnées d'un fichier *myfile.txt* se trouve dans le **bloc no. 67** de la partition. Il n'y a qu'une seule partition sur le disque dur.

⇒Quelles sont les adresses (LBA) que le driver, utilisé par ce système de fichiers, devra envoyer au disque dur pour récupérer le bloc des métadonnées du fichier *myfile.txt*?



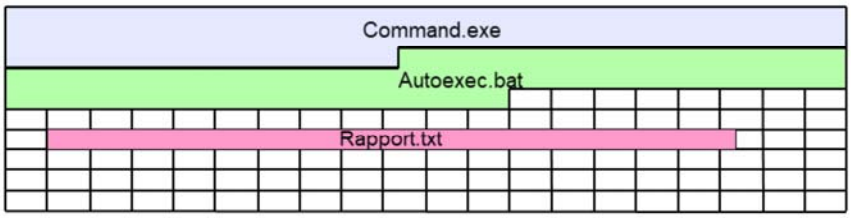
L'allocation des blocs (d'une partition) repose sur différentes méthodes bien connues, relativement complexes. Nous allons découvrir les trois méthodes de base sur lesquelles sont construites aujourd'hui la plupart des systèmes de fichiers.

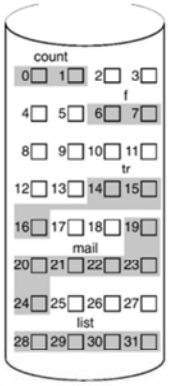
Les méthodes d'allocation des blocs auront un impact direct sur la performance durant les accès disque, sur la perte résiduelle inter-/intra-bloc et donc ainsi sur la fragmentation, sur les possibilités d'accès aux blocs (direct/séquentiel), sur les possibilités de récupérer une partition corrompue grâce aux techniques de journalisation par exemple, etc.

Allocation contiguë (1/2)

- **Allocation contiguë**
- Allocation de blocs contiguës dans la partition
- Métadonnées (référence aux blocs) très simples
- Accès aléatoire possible
- Problème de fragmentation externe
  - Blocs vides dispersés sur le disque
- **Les fichiers ne peuvent pas croître.**
- Exemple
  - ISO9660 (CD-ROM), UDF, etc.

Allocation contiguë (2/2)

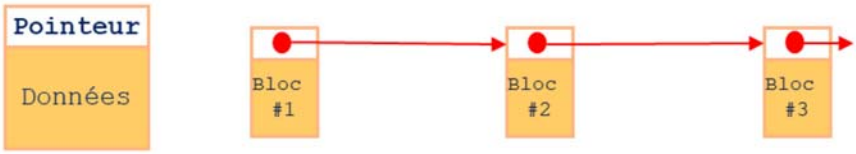




Fichier	Début	Longueur
Count	0	2
Tr	14	3
Mail	19	6
List	28	4
F	6	2

Allocation par listes chaînées (1/6)

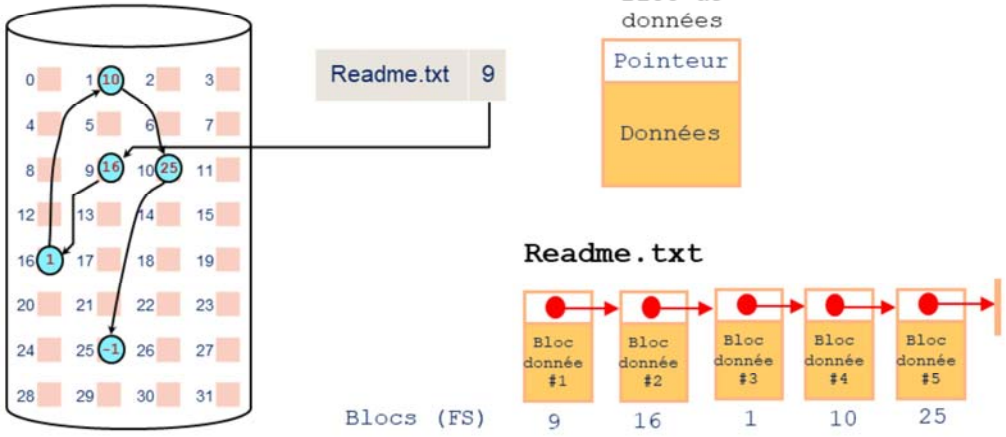
- Les blocs sont chaînés les uns avec les autres.
- Chaque bloc doit contenir un pointeur vers le bloc suivant.



- Un bloc contient des informations relatives aux *données utilisateur* **<u>et aussi</u>** des métadonnées de type *système (noyau)*.

Allocation par listes chaînées (2/6)

- Les métadonnées doivent contenir la référence vers le **premier** bloc du fichier.



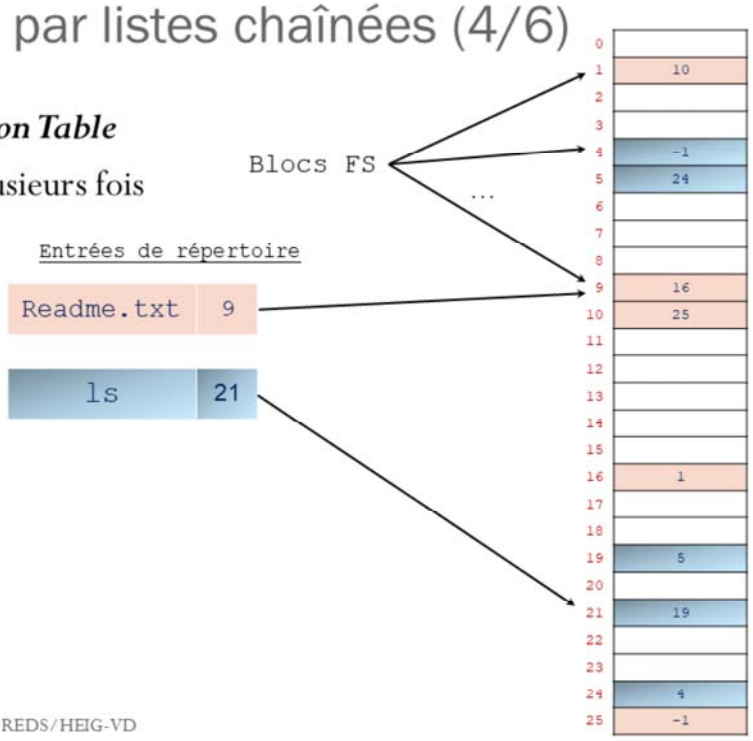
Allocation par listes chaînées (3/6)

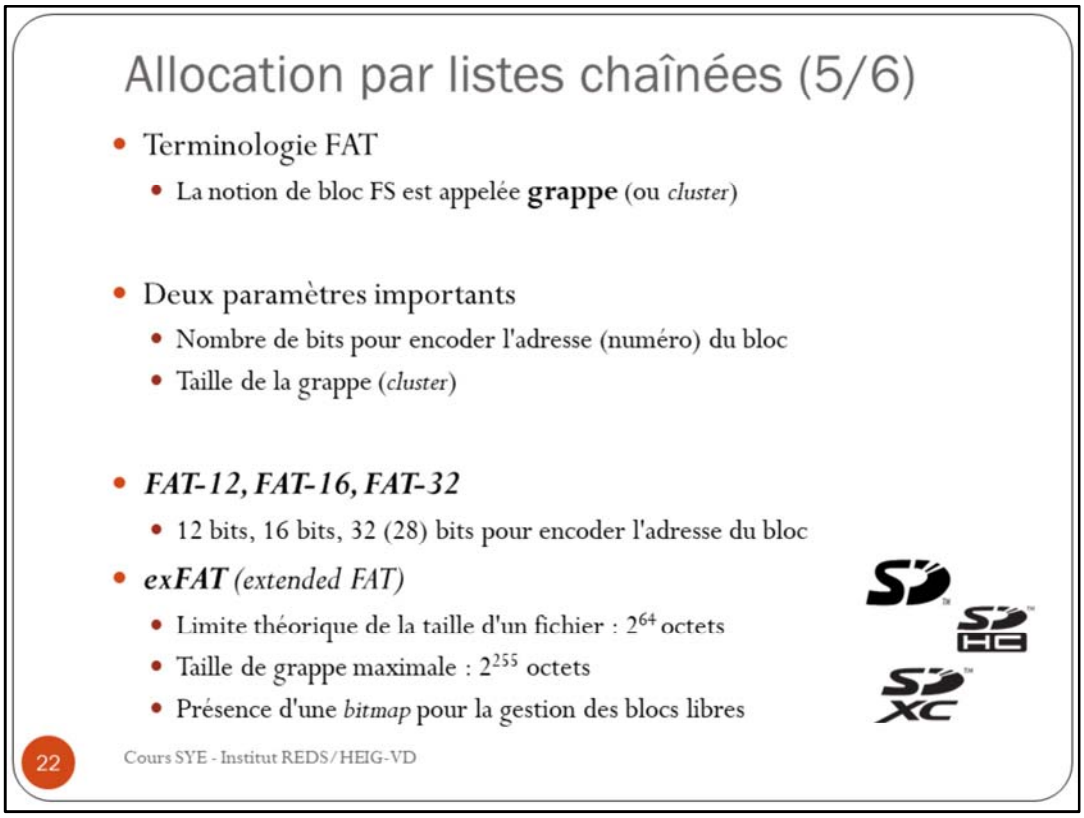
- Bonne gestion de l'espace disque
  - Limitation de la fragmentation externe
- Difficile de faire de l'accès direct
- Données critiques à l'intérieur d'un bloc de donnée
  - Un bloc corrompu conduit à la **rupture** de la chaîne.
- Amélioration significative en *sortant les pointeurs de la chaîne* des blocs de données
  - Solutions de type **FAT** largement répandue




Allocation par listes chaînées (4/6)

- ***File Allocation Table***
- Répliquée plusieurs fois





Dans les systèmes de fichiers de type FAT-*x*, la valeur *x* représente le nombre de bits prévu pour stocker l'adresse d'un bloc. Par ailleurs, la notion de bloc est équivalente à la notion de *grappe* (ou *cluster*) selon la terminologie FAT.

Ainsi, les limitations d'une partition FAT sont donc essentiellement liées aux nombres de bits prévus pour l'adresse d'une grappe et la taille de celle-ci.

Les tailles de grappe possibles dépendent du type de FAT et ont évoluées au fil des années. Typiquement, un système FAT-32 dispose de taille pouvant aller de 4 Ko à 64 Ko.

Le format *exFAT* est différent des formats FAT précédents; il offre la possibilité de gérer de très grands fichiers avec des tailles de grappes très élevés. La spécification de ce format n'est pas publiée. C'est le format utilisé sur de nombreux supports de stockage comme les cartes SD de type SDXC.

Allocation par listes chaînées (6/6)

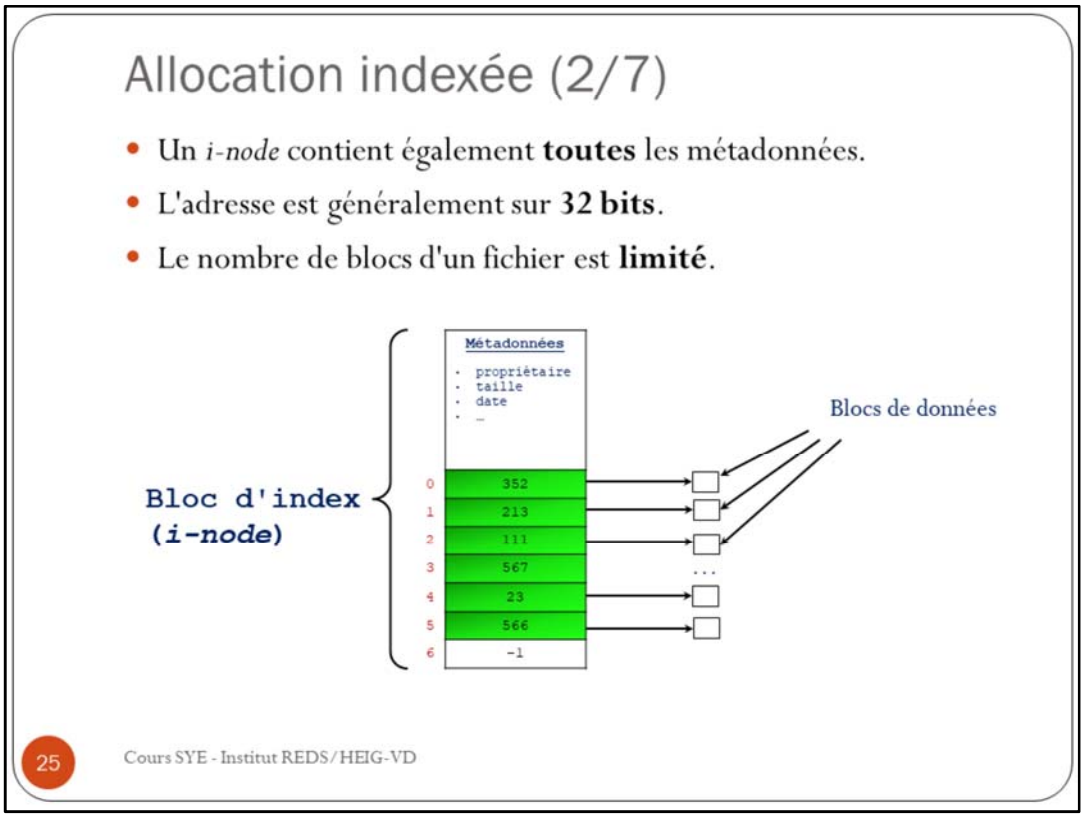
- Soit un disque de 20 Go et une taille de bloc de 4 Ko. Les numéros de bloc sont codés sur 32 bits.

⇨ Quelle sera la taille exacte (en octet) de la FAT ?

⇨ Jusqu'à quelle capacité pourrait-on reformater un disque avec ces caractéristiques ?

Allocation indexée (1/7)

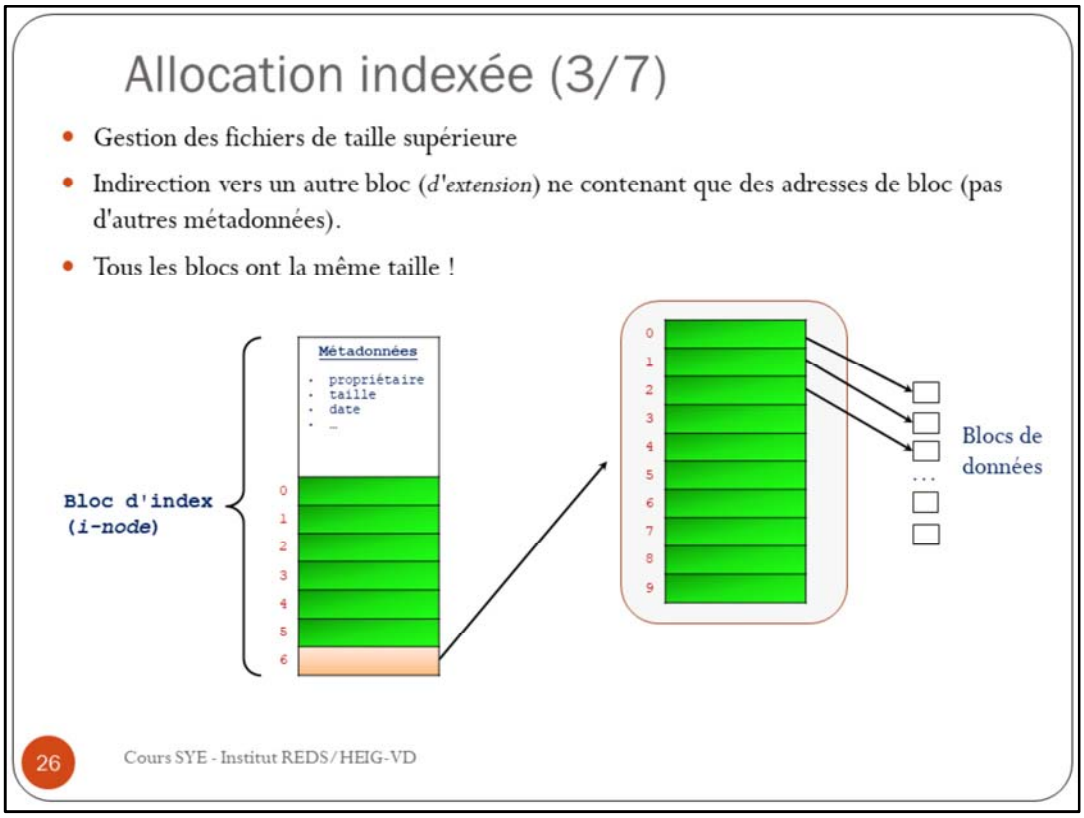
- Utilisation d'une table d'index
  - Chaque entrée de la table indique une adresse d'un bloc de donnée.
  - Les blocs sont ordrés selon les index de la table
  - Pas de chaînage
  - Limitation due à la taille de la table
- La table d'index est contenu dans un bloc.
  - Bloc d'index ou ***i-node***



L'i-node est généralement initialisé à la création du fichier. Cependant, afin de rendre le système plus performant, une collection d'*i-nodes* sont créés et pré-initialisés durant l'opération de formatage. Cela permet au système de fichier, connaissant l'emplacement de ces blocs d'index pré-initialisés, de disposer très rapidement d'un tel bloc et de pouvoir y remplir les métadonnées, ainsi que les références sur les blocs utilisateur.

S'il n'y a plus d'*i-node* disponible, le système de fichier passera par la gestion des blocs libres pour "demander" un bloc disponible, tout comme lors de l'allocation des blocs utilisateur.

Il est à noter que les métadonnées font partie intégrante de l'*i-node*. C'est dire que l'ensemble des adresses disponibles pour décrire le contenu du fichier est limité. Par ailleurs, si l'on considère un bloc de 4 Ko (ce qui est généralement le cas pour être compatible avec le système paginé) et des adresses de blocs stockées sur 32 bits, l'ensemble des adresses est relativement petit. C'est pourquoi, il est nécessaire d'avoir un mécanisme permettant l'utilisation de blocs d'index additionnels, appelés aussi ***blocs d'extension***.

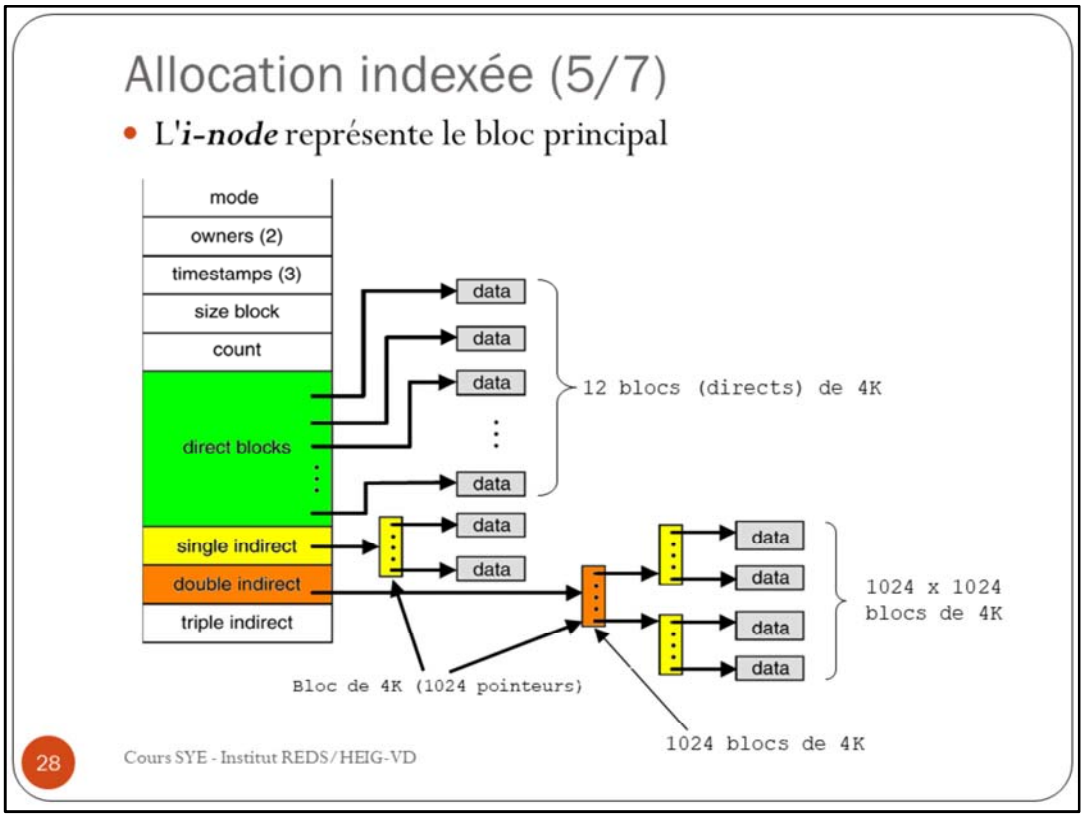


L'indirection de bloc permet d'introduire des adresses supplémentaires permettant ainsi de gérer des fichiers de plus grande taille. Chaque bloc d'extension ne peut contenir que des adresses vers des blocs de données.

La taille d'un fichier reste toutefois limitée par le fait que le nombre d'adresse reste limité par la taille du bloc (et la taille de l'adresse elle-même).

Allocation indexée (4/7)

- Trois niveaux pour les blocs d'extension
  - *Indirect simple*
  - *Indirect double*
  - *Indirect triple*
- Le nombre d'indirection est déterminé lors de du formatage de la partition.
  - Il n'est en principe pas possible de gérer une partition avec une configuration mixte.

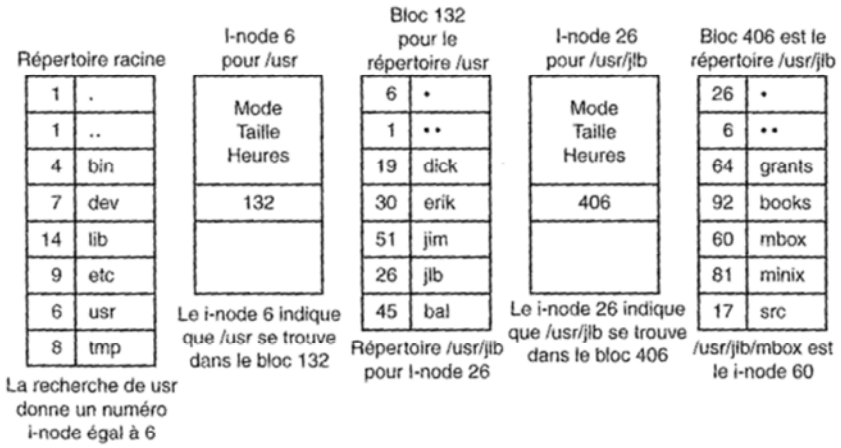


La figure ci-dessus montre l'organisation d'un bloc d'index (*i-node*) dans le cas ou le formatage a prévu un adressage pouvant aller jusqu'à trois indirections (*triple indirect*). La capacité total d'un fichier n'utilisant que des adresses directs est, dans ce cas, limité à 48 Ko (12 blocs de 4 Ko). L'utilisation d'une simple indirection permet de rajouter 1024 adresses supplémentaires (4096 / 4 octets, si l'adresse est codée sur 32 bits), soit une extension de 1024 * 4 Ko = 4096 Ko. La taille d'un tel fichier pourra donc atteindre au maximum 48 Ko + 4096 Ko = 4'144 Ko, soit env. 4 Mo. De même, le rajout d'une double indirection permet d'étendre le fichier à une taille maximale de 48 Ko + 4'096 Ko + (1024*1024) * 4 Ko ≅ 4 Go.

Le temps d'accès à un bloc dépendra donc fortement du nombre d'indirection. En revanche, cette décomposition permet de favoriser les fichiers de petite et moyenne taille, ce qui est le plus souvent utilisé. Plus rarement, les gros fichiers nécessiteront des indirections supplémentaires, mais dans ce cas, l'utilisation de mémoire tampon intermédiaire permettra aussi de réduire le temps d'accès.

Allocation indexée (6/7)

- Accès aux *i-nodes* depuis un répertoire



Allocation indexée (7/7)

- Soit un disque formaté avec une taille de bloc de 1 Ko, et un *i-node* contenant 984 octets de métadonnées. Les adresses de bloc sont sur 32 bits.

⇨ Quelle sera la taille maximale d'un fichier, lorsque l'*i-node* supporte seulement un adressage direct simple ?

⇨ Même question, lorsque l'*i-node* supporte un adressage *indirect simple* ?

⇨ Même question, lorsque l'*i-node* supporte un adressage *indirect double* ?

⇨ Même question, lorsque l'*i-node* supporte un adressage *indirect triple* ?

Références

- A. Silberschatz et al.:
  "Operating Systems Concepts", 6th edition, Wiley
- Andrew Tanenbaum,
  "Systèmes d'exploitation", 2ème édition