SYS2

Système d'exploitation

M.Bastreghi (mba)

Haute École Bruxelles Brabant — École Supérieure d'Informatique

Année académique 2020 / 2021

Système de Fichiers

système de fichiers = organisation d'une partition d'un disque

un disque

- un disque est une mémoire secondaire (mémoire de masse)
- il contient des **secteurs** (formatage de bas niveau)

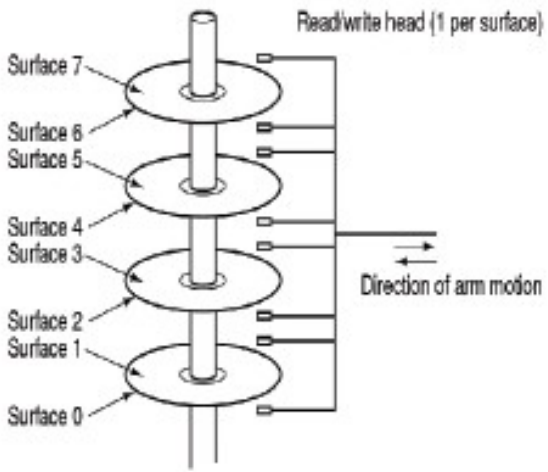
Localiser un byte sur un disque revient à spécifier le numéro de secteur dans lequel le byte se trouve et la position dans le secteur

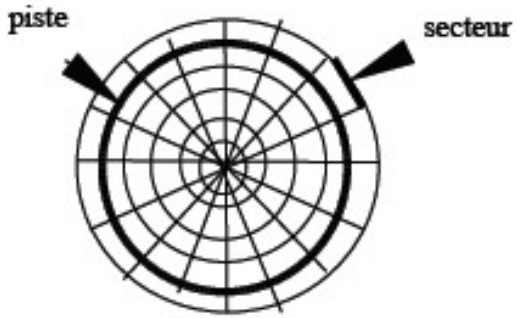
un secteur

- **unité physique** de stockage et d'échange
- un secteur = 512 (2048) bytes de données utiles plus quelques bytes d'information redondante permettant de valider son contenu.

un disque à plateaux

- un disque à plusieurs plateaux :





constitution d'un disque dur [TNB]

un disque à plateaux

- Une **piste** ou track est la circonférence parcourue par la **tête de lecture** pendant une rotation de disque, elle est identifiée par le couple **(Cylinder,Head)**.
- Un **cylindre** est l'ensemble de pistes parcourues sans déplacement de têtes (Heads).
- Sur une piste se trouvent plusieurs **secteurs** numérotés de 1 à 63.
- Un changement de Cylindre nécessite le changement de position des têtes de lecture (lent)

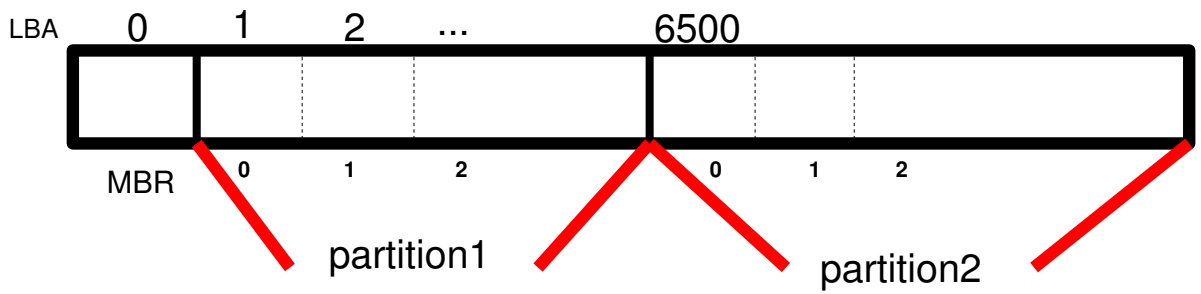
adressage CHS des secteurs

Cylinder - Head - Sector : (un numéro sur 3 bytes)
3 bytes : C (10 bits 0-1023), H (8 bits 0-255), S (6 bits 1-63)
Le MBR est le secteur (0-0-1)

L'adressage CHS n'est plus adapté aux tailles de disque actuelles : < (2 exp 24 secteurs)

adressage LBA des secteurs

Logical Block Address : un disque est une suite de secteurs numérotés depuis 0



Les secteurs du disque sont numérotés à partir de 0
Le Master Boot Record a comme LBA 0

partitionnement et formatage

Un disque est subdivisé en **partitions** (C :, D :, /dev/sda1, /dev/hda1, ...)
Il existe des commandes de partitionnement (**fdisk, parted,...)**

Chaque partition peut être organisée selon un format différent (système de fichiers) grâce à l'opération de **formatage** (**mkfs...**)

systèmes de fichiers

Une partition sera donc organisée en **système de fichiers** (**FAT, NTFS, EXT,...**)
Chaque partition a son propre système de fichiers

Chaque partition a son propre secteur 0

Une deuxième numérotation des secteurs au sein d'une partition coexiste avec la précédente et commence également à 0.

systèmes de fichiers

système de fichiers = organisation d'une partition qui permet de :

- stocker des grandes quantités d'informations de manière **permanente**
- **nommer** ces informations (fichiers)
- **partager** ces informations

- **retrouver (localiser)** les informations grâce à leur nom/chemin et non via des numéros de secteur. (c'est le rôle de l'appel système open)

systèmes de fichiers

la vue système d'un système de fichiers

1. vue **système** du service - **comment** faire cela ?

   (**mise en oeuvre** - code des Appels Système open, read, write, ...)

systèmes de fichiers - rôles

rôles d'un système de fichiers

- définir implantation des fichiers et répertoires
- **allouer de l'espace** aux fichiers/**répertoires** et permettre leur localisation
- gérer l'espace libre, les quotas,...
- permettre de définir des droits...
- ...

systèmes de fichiers - répertoires

- Un système de fichiers va nous permettre de stocker et retrouver les données de nos fichiers.
- Les systèmes de fichiers que nous utilisons le plus ont une structure hyérarchique arborescente dans laquelle les répertoires représentent des noeuds.
- Les répertoires ont eux-mêmes leurs données qui servent à localiser ou décrire les fichiers qu'ils contiennent

systèmes de fichiers - données et métadonnées

métadonnées = données qui décrivent des données.

Soit un fichier m'appartenant et contenant le texte "coucou"

- le texte coucou constitue **les données du fichier**
- MBA, la date de création la taille et les droits du fichier sont des **métadonnées du fichier**
- la taille des secteurs est une **métadonnées du système de fichiers**

systèmes de fichiers - données et métadonnées

métadonnées = données qui décrivent des données.

Soit un fichier m'appartenant et contenant le texte "coucou"

- le texte coucou constitue **les données du fichier**
- MBA, la date de création la taille et les droits du fichier sont des **métadonnées du fichier**
- la taille des secteurs est une **métadonnées du système de fichiers**

- les métadonnées du système de fichier permettent de s'y retrouver
- localiser la description(droits, taille,...) d'un fichier = localiser les métadonnées d'un fichier (répertoire, inode, ...)
- localiser un byte du fichier = localiser ses données

systèmes de fichiers - qualités

qualités d'un système de fichiers

- fiabilité - resistance aux pannes de courant ...
- performance - rapide
- portabilité (devices externes) - supporté par différents OS
- tailles maximum des partitions et fichiers - limitations
- ...

qualités d'un système de fichiers

- fiabilité - resistance aux pannes de courant ...
- performance - rapide
- portabilité (devices externes) - supporté par différents OS
- tailles maximum des partitions et fichiers - limitations
- ...

la portabilité d'un système de fichiers n'est pas une qualité intrinsèque du système de fichiers même (EXT - FAT)

vocabulaire

on distinguera :

- **données** des fichiers et répertoires
- **méta-données des fichiers et répertoires** (droits, taille, propriétaire, date de création,...)
- **méta-données du système de fichiers** (taille des secteurs, localisation de la racine, taille des blocs, blocs libres, blocs défectueux, fichiers remarquables, ...)

- ...

questions

Quelles commandes linux permettent de :

- voir les métadonnées d'un fichier sous linux ?
- modifier les métadonnées ?
- modifier les données d'un fichier en linux ?
- modifier les données d'un répertoire en linux ?

mise en oeuvre : allocation

allocation

comment sont alloués les secteurs aux fichiers/répertoires ?

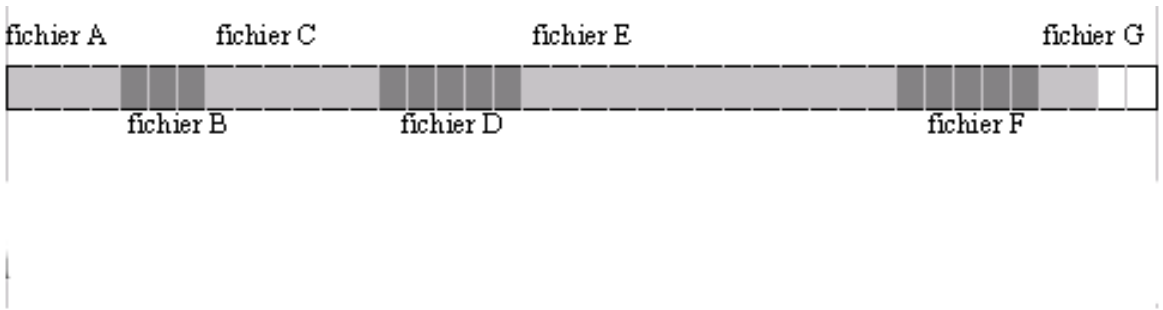
Deux approches :

- allocation contiguë
- allocation par blocs

allocation contiguë

Les bytes de données d'un fichier sont **consécutifs** sur la partition

Tout fichier démarre à une frontière de secteur et occupe les secteurs voisins
Sa lecture nécessite un seul positionnement initial de la tête de lecture

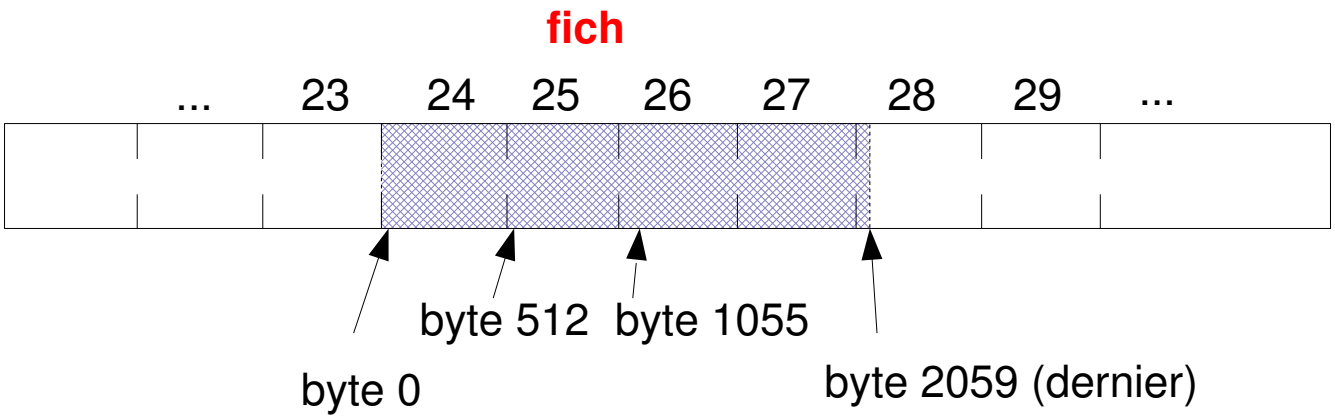


allocation contiguë - exemple localiser un byte

un fichier démarre au secteur 24 et contient 2060 bytes.

le byte n° 1055 du fichier se trouve :
dans le secteur : 26 = 24 + (1055 DIV 512)
dans le byte : 31 = (1055 MOD 512)

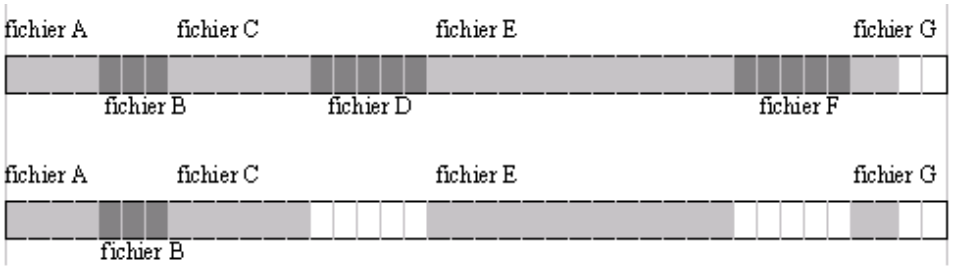
le byte n° 512 du fichier se trouve :
dans le secteur : 25 = 24 + (512 DIV 512)
dans le byte : (512 MOD 512) = 0



ajout - suppression

Et si on modifie ce système de fichiers ?

- supprimer D,F
- augmenter la taille de A ?
- créer un gros fichier ?



=> On va générer de la **Fragmentation Externe**

allocation contiguë - performance

- L'espace libre du disque s'est fragmenté
- il n'est plus possible de créer de nouveau fichiers sans devoir déplacer d'autres fichiers.
- les déplacements sont coûteux en écritures

fragmentation externe = fragmentation de l'espace libre

allocation contiguë - critique

(+)avantages (-)inconvénients

- (+) rapidité d'accès en **lecture**
- (-) **lenteur inadmissible** réécritures dues à la **fragmentation externe** suite à des modifications

-> convient aux systèmes de fichiers en lecture uniquement

Questions ?

allocation par blocs

Fichiers et partition découpés en blocs de **taille fixe**

- bloc = **nombre entier de secteurs (1, 2, 4, 8,. . ., 64, . . .)**
- bloc = **unité d'allocation**
- bloc = **unité d'accès** logique

allocation par blocs

- Un fichier occupe un **nombre entier de blocs**
- L'espace alloué aux fichiers est **non nécessairement contigu**
- => Des **métadonnées supplémentaires** sont nécessaires pour décrire le chaînage des blocs de chaque fichier/répertoire

allocation par blocs

- Un fichier occupe un **nombre entier de blocs**
- L'espace alloué aux fichiers est **non nécessairement contigu**
- => Des **métadonnées supplémentaires** sont nécessaires pour décrire le chaînage des blocs de chaque fichier/répertoire

dans le "jargon windows" un bloc est appelé **cluster**

le système continue à dialoguer avec le contrôleur de disque en termes de secteurs, mais les opérations d'accès et d'allocation se font par bloc (n secteurs à la fois).

allocation par blocs - numérotation des blocs

La découpe en blocs au sein de la partition :

- Les blocs commencent à la frontière d'un secteur
- Les blocs sont numérotés
- Le n° de bloc de la partition (P), permet de calculer le numéro de son premier secteur (ce n° est l'**adresse** du bloc P)
  - adresse de P = P*"nb Secteurs par bloc"

allocation par blocs - adresse d'un bloc

Les métadonnées du système de fichiers occupent également de l'espace disque souvent en dehors des blocs
De ce fait, le bloc 0 n'est généralement pas aligné avec le secteur 0.

Donc :

- adresse P = "adresse du bloc 0" + P x "nb secteurs par bloc"

FAT, EXT, NTFS ont cette particularité

allocation par blocs - adresse d'un bloc

Les métadonnées du système de fichiers occupent également de l'espace disque souvent en dehors des blocs
De ce fait, le bloc 0 n'est généralement pas aligné avec le secteur 0.

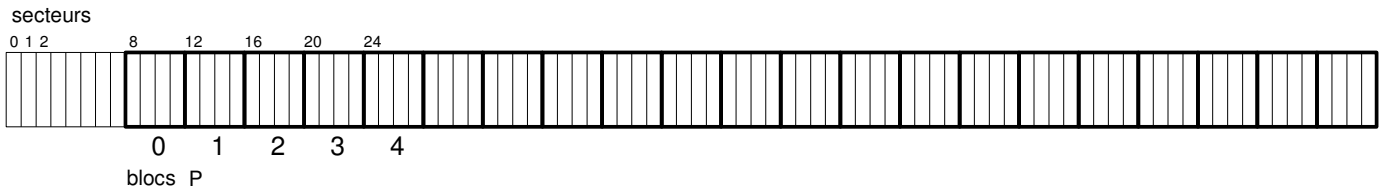
Donc :

- adresse P = "adresse du bloc 0" + P x "nb secteurs par bloc"

allocation par blocs - adresse d'un bloc

Soit une partition divisée en blocs : bloc = 4 secteurs ici.

Soit le bloc 0 aligné sur le secteur 8 de la partition



le bloc 0 commence au secteur : $8 + (0 \times 4) = 8$

le bloc 3 commence au secteur : $8 + (3 \times 4) = 20$

allocation par blocs - métadonnées du système de fichiers

métadonnées du système de fichiers en allocation par blocs

- **taille du secteur**
- **nombre de secteurs par bloc | taille du bloc**
- **début du bloc 0**

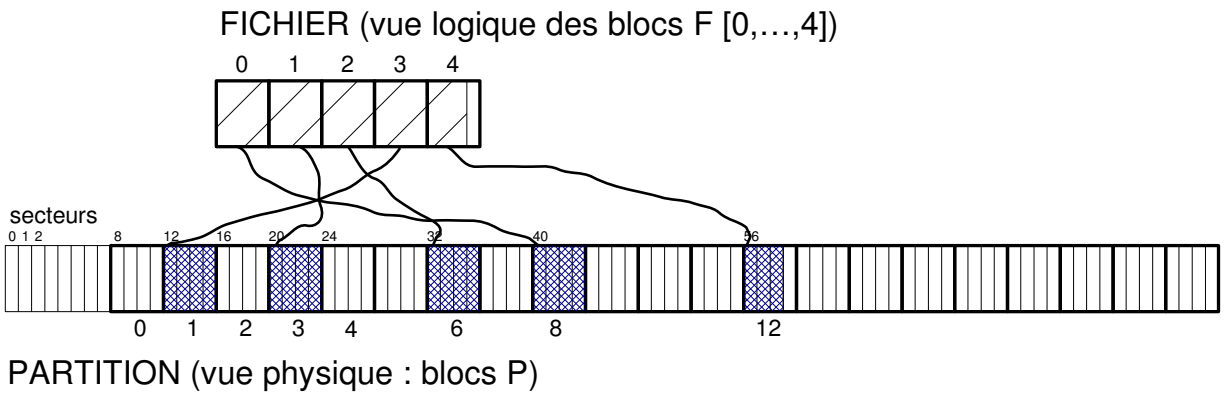
allocation par blocs numérotation F et P

Deux numérotations pour le même bloc (F, P) à ne pas confondre

- **F** = numéro d'ordre du bloc au sein du fichier (un fichier de 2000 bytes contient deux blocs (F=[0,1]) de 1024 bytes). Il s'agit d'une vue logique.
- **P** = numéro d'ordre du bloc au sein de la partition (vue physique)

allocation par blocs - exemple

- F=4 donne P=12 dans l'exemple



ajout - suppression, aisés

- ajout/suppression de données sont plus aisés car la taille des blocs est fixe et les fichiers non contigus
- les blocs des fichiers ont tendance à s'éparpiller (fragmentation des fichiers) - les déplacements de têtes deviennent pénalisants à la lecture/écriture.

allocation par blocs - fragmentation interne et des fichiers

- on voit apparaître une perte de place importante au sein du dernier bloc du fichier pas toujours rempli à 100%
- la **fragmentation interne** désigne l'espace perdu dans le dernier bloc des fichiers

- on observe une dispersion des blocs d'un fichier
- cette dispersion se nomme **fragmentation des fichiers**

- on voit apparaître une perte de place importante au sein du dernier bloc du fichier pas toujours rempli à 100%
- la **fragmentation interne** désigne l'espace perdu dans le dernier bloc des fichiers

- on observe une dispersion des blocs d'un fichier
- cette dispersion se nomme **fragmentation des fichiers**

- la fragmentation interne induit une perte d'espace disque, en effet un bloc est alloué à un fichier de manière exclusive
- la fragmentation des fichiers ralentit les accès disque à cause des déplacements de têtes fréquents qu'elle induit (cette particularité concerne essentiellement les disques mécaniques (à plateaux) dont les capacités (16TiB) dépassent encore aujourd'hui celles des disques de technologies SSD)

allocation par blocs - taille des blocs

fragmentation interne

le **dernier bloc** n'est pas toujours entièrement rempli

fragmentation interne = espace perdu à l'intérieur des derniers blocs des fichiers

choisir la taille à donner aux blocs : un choix stratégique

- grands blocs -> perte de place
- petits blocs -> perte de temps

allocation par blocs - taille des blocs

- taille d'1 bloc = **n** secteurs partition (n >= 1)
- **choix de n** fait au **formatage**

Quel est le bon choix pour n ?

Le bon compromis dépend de l'utilisation

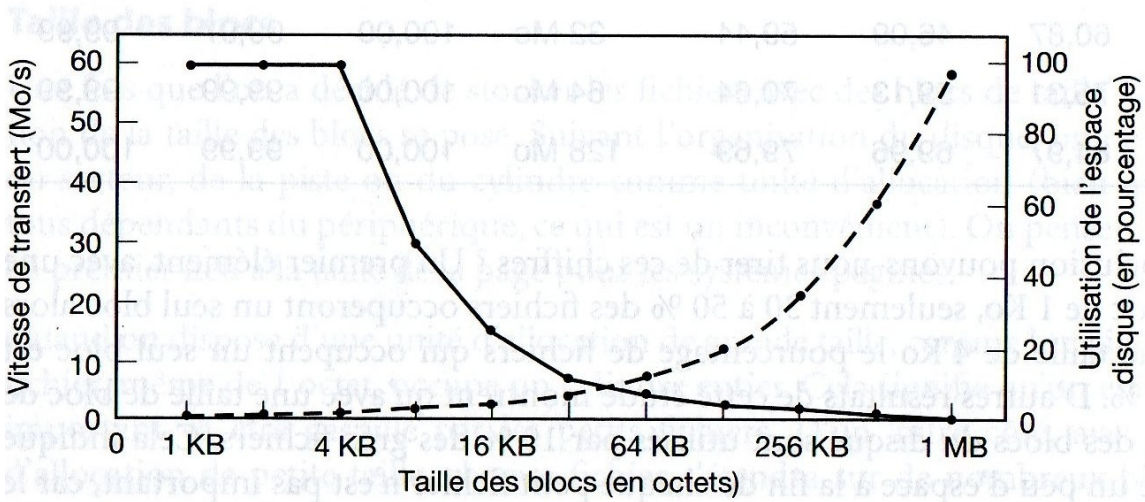
allocation par blocs - taille des blocs

«Choisir est un peu mourir ... »

auteur inconnu

allocation par blocs - taille des blocs

2 à 4 KiB = taille moyenne des fichiers sur un système Unix [TNB]



allocation par blocs - taille des blocs

avec des blocs de 8KiB 50% de l'espace disque est perdu !
avec des blocs de 64KiB 90% !

taille des blocs - gain de place ou performance ?

"avec des partitions > 1TiB il peut être préférable d'augmenter la taille des blocs à 64KiB et d'accepter un gaspillage de l'espace disque" [TNB]

dans ce cas, 1TiB -> +/-100 GiB utiles !

blocs - fragmentatioon des fichiers

fragmentation des fichiers

allocation par blocs -> **fichiers fragmentés** (blocs éparpillés)

-> beaucoup de déplacements des têtes -> LENT
-> solutions :

- **réorganisation systématique** (adopté par linux)
- outils de **défragmentation** fournis (xp, seven, vista)

allocation par blocs - critique

(+)avantages (-)inconvénients

- (+) souplesse nécessaire aux mises à jour de fichiers
- (-) sujet à la fragmentation interne et donc à une utilisation réduite de l'espace disque
- (-) sujet à la fragmentation des fichiers nuisant à la rapidité d'accès

Adapté à des systèmes de fichiers réinscriptibles, mais il faudra veiller à minimiser les problèmes survenus suite à ces nouvelles formes de fragmentation.

Questions ?

questions

- en allocation de l'espace par blocs, une plus grande taille de bloc augmente la fragmentation externe[V-F]
- quelle différence y a-t-il entre fragmentation *externe* - *interne* - *des fichiers* ?
- soit un fichier de 10360 bytes et une taille de bloc de 4KiB en allocation par blocs. Combien de blocs sont utilisés par ce fichier ? Quel pourcentage du dernier bloc est occupé ?

répertoires

répertoire : fichier particulier qui contient quelques métadonnées des fichiers

localiser un fichier - métadonnées du fichier

Un répertoire décrit d'autres fichiers.
Ses données sont les métadonnées des fichiers qu'il contient (FAT) ou permettent de retrouver ces métadonnées (EXT, NTFS)
Les métadonnées permettent notamment la **localisation** des fichiers

- ext - le **répertoire parent** contient les numéros d'inode des fichiers
- FAT - le **répertoire parent** contient le premier bloc des fichiers dans la partition
- NTFS - le **répertoire parent** contient le numéro d'entrée du fichier MFT décrivant chaque fichier du système

répertoires - métadonnées des fichiers

exemple de métadonnées d'un fichier :

- nom
- premier bloc (valeur de P pour F=0)
- longueur du fichier
- attributs divers
- n° de l'entrée de la MFT (NTFS)
- n° d'inode (EXT)
- ...

répertoires - métadonnées des fichiers

localiser un fichier ?

- localiser un fichier => lire le **répertoire parent**
- et pour lire le répertoire ? -> lire son parent !

Où cela s'arrête ?
Qui est le parent du répertoire racine ?

répertoires - commencer par la racine

La racine doit avoir un emplacement **fixe connu ou calculable**.
EXT, FAT, NTFS utilisent des **conventions**

- position **calculable** ou renseignée à un endroit calculable (FAT, NTFS)
- renseigné à une position **fixe** (EXT : inode 2)

répertoires - localiser un fichier

lire le fichier /home/mba/test
demande de **localiser, et lire 3 répertoires** :

- /
- / home
- / home / mba

Ce dernier donnera les informations qui permettent de localiser et finalement lire le fichier test

Questions ?

trouver la suite des blocs d'un fichier ?

Dans le cadre de l'allocation par blocs, **début et longueur** ne suffisent plus à localiser un byte N si celui-ci est plus loin que le premier bloc du fichier (F>0)
Comment établir la correspondance entre F et P dans ce cas ?

Des métadonnées supplémentaires doivent nous renseigner sur le chaînage des blocs au sein des fichiers.
Ces métadonnées sont typiquement des métadonnées des fichiers

Dans le cadre de l'allocation par blocs, **début et longueur** ne suffisent plus à localiser un byte N si celui-ci est plus loin que le premier bloc du fichier (F>0)
Comment établir la correspondance entre F et P dans ce cas ?

Des métadonnées supplémentaires doivent nous renseigner sur le chaînage des blocs au sein des fichiers.
Ces métadonnées sont typiquement des métadonnées des fichiers

pas toujours : en FAT le chaînage est une métadonnée du système de fichiers (la **table d'index**).

localiser - où est le byte N ?

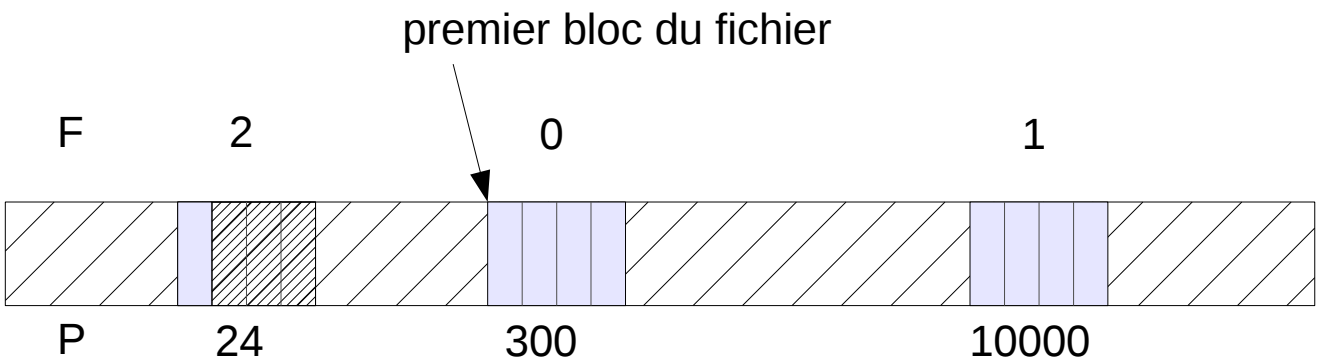
Tout fichier démarre à une frontière de bloc de la partition
Les blocs sont numérotés au sein du fichier depuis 0
Le byte N d'un fichier se trouve dans le bloc P correspondant à son bloc n° **F**
F = (N DIV TailleBloc)

Mais à quel bloc **P** de la partition correspond **F** ? Le système de fichiers doit fournir les informations qui permettent de le localiser

- F = n° du bloc dans le fichier
- P = n° du bloc de la partition

localiser - métadonnées ?

Comment établir la correspondance entre F (numéro d'ordre du bloc dans le fichier) et P (sa position dans la partition) ?



position du bloc suivant du fichier ?

localiser - métadonnées ?

Différentes approches permettant ed **localiser chaque bloc** du fichier :

- blocs chaînés
- un index par fichier (EXT, NTFS)
- **table d'index** globale pour les blocs (FAT)

Questions ?

Systèmes de Fichiers - appels système

Un Système d'exploitation mettra à disposition des applications les services pour utiliser les système de fichiers. un système Différentes approches permettant edlinux prévoit notamment :

- **open** localisation d'un fichier pour une session de lecture/écriture
- **close** cloture de la session de lecture/écriture

La localisation d'un fichier est une opération longue (plusieurs lectures de répertoires).
On travaille plutôt par sessions.
une seule localisation pour plusieurs lectures/écritures.

Systèmes de Fichiers - appels système

Quelques Appels Système pour les fichiers

- read - lire des bytes (localiser les bytes)
- write - écrire des bytes (localiser les bytes)
- lseek - modifier la position de lecture/écriture
- ...

appels Système - open / close

La localisation du fichier est mémorisée dans la TDFO par l'appel système **open**
TDFO = Table de Descripteurs de Fichiers Ouverts

Lire un fichier se fait généralement par plusieurs appels consécutifs à l'Appel Système **read**

La position courante (prochain byte à lire ou écrire) est mémorisée dans la même table et sera mise à jour par les appels système (read, write, lseek,...)

L'appel système **close**, libère l'entrée de la table

questions

- l'appel système open (/home/mba/test) lit trois fichiers [V-F]
- localiser le répertoire / demande de lire son parent [V-F]

remerciements

remerciements à P.Bettens et M.Codutti
pour la mise en page :-) Mba

Crédits

Ces slides sont le support pour la présentation orale de l'activité d'apprentissage **SYS2** à la HE2B-ÉSI

Crédits Crédits

La distribution opensuse
du système d'exploitation GNU Linux.
LaTeX/Beamer comme système d'édition.
GNU make, rubber, pdfnup, ... pour les petites tâches.

Images et icônes

deviantart, flickr, The Noun Project