

M1105 - Systèmes d'exploitation

Fichiers & Système de fichiers

`sebastien.kramm@univ-rouen.fr`

IUT de Rouen, dépt. Réseaux & Télécoms

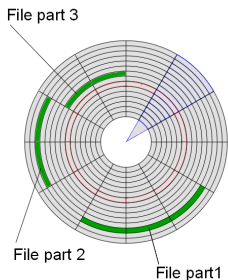
Version du 20 octobre 2016

Sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

Fichier ?

- Définition : séquence d'octets qui existe indépendamment d'un programme utilisateur.
- Stocké sur un support persistant (disque, CD, bande magnétique, ...) **ou** en mémoire vive.
- Identifié par un **nom**
- Le stockage n'est pas forcément séquentiel sur le support physique (fragmentation) :



Fichiers vu par l'OS

- L'OS fournit une interface (API) composée d'un ensemble d'appels systèmes pour gérer les fichiers :
fournit une **abstraction** permettant de manipuler le fichier quelque soit sa localisation physique ou sa fragmentation :
 - Ouvrir le fichier en lecture ou en écriture
 - Lire et/ou écrire
 - Ajouter du contenu à la fin du fichier
 - Se positionner à une certaine position dans le fichier
- "Répertoire" : type spécial de fichier "contenant" d'autres fichiers
Identifié comme tel par un flag dans l'organisation logique des fichiers.

Système de fichiers ?

Deux sens différents :

- 1 Structure logique d'un disque : comment les fichiers sont-ils stockés sur le disque ("format") ?
→ FAT32, NTFS, ext3, ext4, ISO9660 (CD et DVD), etc.
- 2 Collection de fichiers sur une partition d'un disque : organisation logique.

Name	Size	Type	Date Modified
bin	153 items	folder	ven. 29 avril
boot	14 items	folder	dim. 04 sept.
cdrom	0 items	Folder	mar. 10 mars
dev	218 items	folder	lun. 19 sept.
etc	303 items	folder	lun. 19 sept.
home	4 items	folder	ven. 11 déc.
lib	33 items	folder	dim. 04 sept.
lib64	1 item	Folder	dim. 04 sept.
lost+found	? items	Folder	mar. 10 mars
media	1 item	folder	sam. 17 sept.
mnt	3 items	folder	jeu. 02 juil. 2
opt	5 items	folder	jeu. 08 sept.
proc	311 items	Folder	lun. 19 sept.
root	? items	folder	mar. 22 mars
run	40 items	folder	lun. 19 sept.

Sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau**
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux



Système de fichiers : structure

Différents format ont été développés au cours du temps :

- Fonction du support physique :
DD \neq disque optique \neq bande magnétique
- On doit associer à chaque fichier des **méta données** : localisation physique, taille, propriétaire, permissions, date création, etc.
→ Implique une **diminution** de l'espace de stockage utile.
- L'organisation de ces métadonnées amène des **limitations** (longueur et encodage nom de fichier, taille max, etc.)
- L'organisation interne du disque (organisation en secteurs) : diminution espace utile
- Autres différences entre formats :
 - robustesse aux erreurs de la couche physique,
 - performance pour un type d'application donné,
 - risque de fragmentation excessive,
 - journalisation,
 - ...

Format de partition : questions à résoudre

- Sur disque, les fichiers occupent des blocs de taille fixe : comment fixer cette taille ?
- Comment gérer la succession de plusieurs blocs ?
- Optimisation pour les fichiers de grande taille ou de petite taille ?
- Pour les supports type HD
 - Comment trouver des blocs de disque nécessaires quand la taille du fichier augmente ?
 - Que fait-on des blocs libérés en cas de suppression d'un fichier ?
- Robustesse : quelle redondance prévoir ? (journalisation ?)
- Quelle finesse dans la gestion des droits d'accès ?

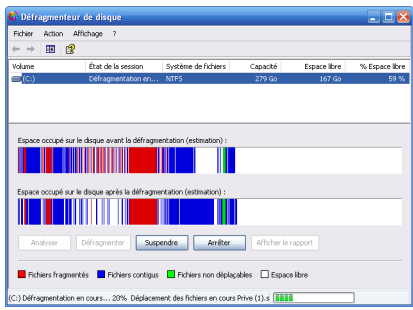
Systèmes de fichier journalisé

La journalisation est un concept destiné à apporter de la robustesse aux pannes

- Principe : tenue d'un journal (fichier spécial) référençant les opérations d'écriture sur le support **avant** que ce dernier ne soit réellement mis à jour
- Objectif : permet une reprise d'activité à la suite d'une coupure brutale (arrêt électrique).
- Inconvénient : diminue les performances
- Exemple de système de fichier journalisé : ext3, ext4 (Linux), NTFS (Windows), ...
- Exemple de système de fichier **non** journalisé : FAT16, FAT32 (Windows), ext2 (Linux)

Fragmentation des fichiers

- Certains systèmes de fichiers Windows sont plus sensibles au problème de la fragmentation, au fil du temps
- Cette fragmentation des fichiers **ralentit** le fonctionnement de la machine
⇒ nécessité de procéder régulièrement à une **défragmentation**.



- Les systèmes de fichiers Linux (ext3, ext4) y sont moins sensibles

Limitations

- Chaque système de fichier a des limitations
 - taille max du disque, du fichier, de la partition, ... ;
 - nombre de caractères en encodage du nom ;
 - nombre de fichiers max. par répertoire ;
 - gestion des permissions ;
 - datation ;
 - etc.
- Exemples :
 - FAT : noms de fichiers limités à 8 caractères ASCII + 3 pour l'extension, insensible à la casse.
Depuis Windows NT : VFAT a étendu la limitation à 255 car.
 - NTFS : génération automatique d'un nom "8.3", pour applications "*legacy*".

Importance ?

⇒ Lors de la copie d'une arborescence d'un support sur un autre !

Limitations

- Chaque système de fichier a des limitations
 - taille max du disque, du fichier, de la partition, ... ;
 - nombre de caractères en encodage du nom ;
 - nombre de fichiers max. par répertoire ;
 - gestion des permissions ;
 - datation ;
 - etc.
- Exemples :
 - FAT : noms de fichiers limités à 8 caractères ASCII + 3 pour l'extension, insensible à la casse.
Depuis Windows NT : VFAT a étendu la limitation à 255 car.
 - NTFS : génération automatique d'un nom "8.3", pour applications "*legacy*".

Importance ?

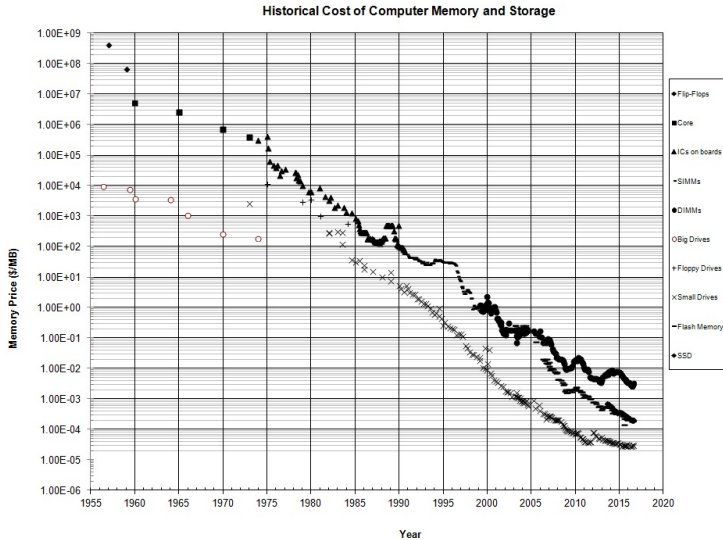
⇒ Lors de la copie d'une arborescence d'un support sur un autre !

Sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur**
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux



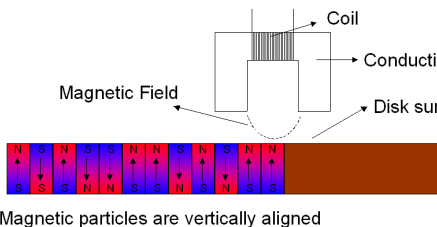
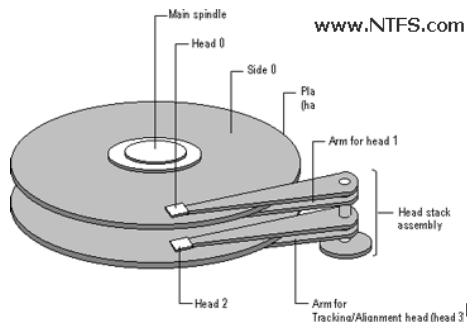
Evolution du coût du Mo (2016/08)



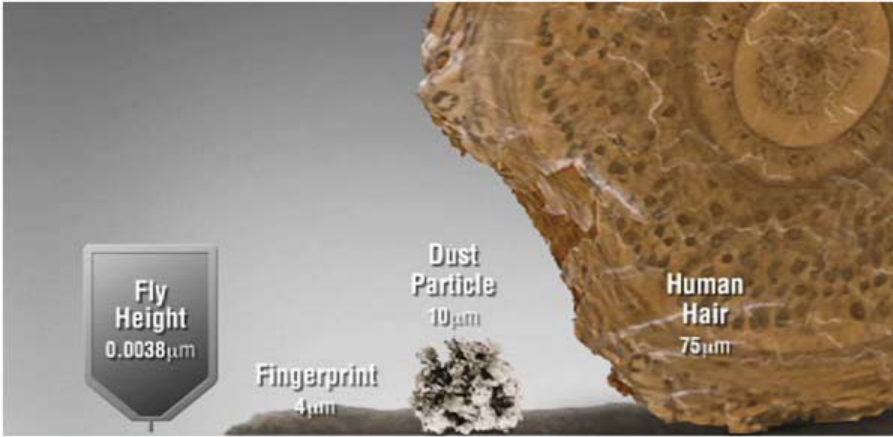
Source : John C. McCallum

Organisation physique d'un disque

- Principe général : plateaux tournant avec des bras de lecture, supportant des têtes magnétiques.



Dimensions



source : seagate.com

Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 **Structure d'un disque dur**
 - Présentation et historique
 - **Caractéristiques**
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

Spécifications

- Vitesse de rotation (constante) : de 5k à 10k tr/mn.
- Nbe de plateaux, nbe de têtes
- *Seek Delay* : latence liée au déplacement lateral de la tête
- *Rotation Latency* : latence liée à la rotation
- Taille des secteurs et des blocs

Exemple : Seagate Barracuda 3TB

Property	Measurement
Platters	3
Heads	6
Rotation Speed	7200 rpm
Average Seek Delay	8ms
Average Rotation Latency	4ms
Bytes/Sector	512
Sectors/Track	63

Les constructeurs donnent aussi le débit moyen crête et continu.

Rem : Disques optiques : vitesse linéaire constante → vitesse rotation variable

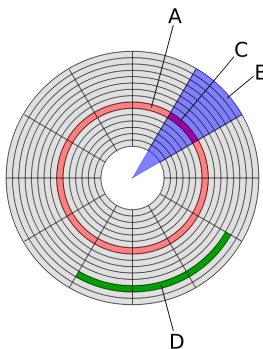
Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur**
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau**
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

Géométrie du disque

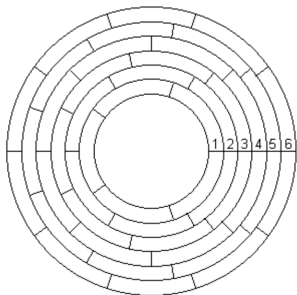
- Le disque est découpé en cylindres, *tracks* (pistes) et secteurs
- secteur : plus petite unité d'allocation
contient les données utiles, plus des métadonnées, en particulier des infos permettant la correction d'erreur (ECC : *Error Correction Code*)
La taille du secteur est **fixée** par le constructeur
- **block** (D) : unité d'allocation accessible par le système de fichier ("formatage" du disque)
peut-être de 1, 2, 4, ... secteurs

Historiquement : on accédait à un secteur via l'adressage "*Cylinder-Head-Sector*" (CHS)



En réalité

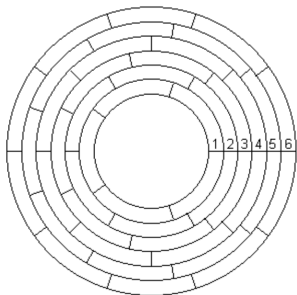
- Problème de l'approche précédente : les secteurs près du centre devront avoir une densité bien plus importante que près du bord
- Solution : les disques ont un nombre de secteurs par piste variable, fonction de la **zone**.



Chaque secteur est identifié par un **index** (adressage dit "LBA", *Logical Block Adress*), plutôt que par sa position : *Zone Bit Recording (ZBR)*.

En réalité

- Problème de l'approche précédente : les secteurs près du centre devront avoir une densité bien plus importante que près du bord
- Solution : les disques ont un nombre de secteurs par piste variable, fonction de la **zone**.



Chaque secteur est identifié par un **index** (adressage dit "LBA", *Logical Block Adress*), plutôt que par sa position : *Zone Bit Recording (ZBR)*.

Organisation d'un secteur



- Historiquement : 512 octets utiles + 65 octets
- Formattage bas niveau (marques de séparation et code de correction d'erreur)

2010 → : *Advanced Format* (format "4K")

- Depuis 2010, les constructeurs utilisent des secteurs de 4096 octets (4KiB)
- Motivation : augmentation de la taille moyenne des fichiers → perte de performance avec l'augmentation de la densité.

Comparison of 512- and 4096-byte sector formats^[7]

Description	512-byte sector	4096-byte sector
Gap, sync, address mark	15 bytes	15 bytes
User data	512 bytes	4096 bytes
Error-correcting code	50 bytes	100 bytes
Total	577 bytes	4211 bytes
Efficiency	88.7%	97.3%

source : Wikipedia

- Pas supporté par certains "vieux" OS (WinXP) → le DD fonctionne alors en mode "émulation secteur 512 B"

Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur**
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs**
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

Gestions de l'allocation de blocs pour des fichiers

Problématiques : soit un disque dur partiellement occupé.

- Je veux y stocker un nouveau fichier de grande taille. Où le placer ?
- Un des (gros) fichiers existant voit sa taille doubler. Comment stocker ce fichier ?

Lors de la conception d'un système de fichiers, plusieurs stratégies peuvent être envisagées :

- 1 Allocation contigue
- 2 Allocation chaînée
- 3 Allocation indexée

Gestions de l'allocation de blocs pour des fichiers

Problématiques : soit un disque dur partiellement occupé.

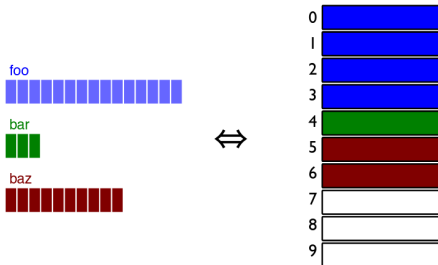
- Je veux y stocker un nouveau fichier de grande taille. Où le placer ?
- Un des (gros) fichiers existant voit sa taille doubler. Comment stocker ce fichier ?

Lors de la conception d'un système de fichiers, plusieurs stratégies peuvent être envisagées :

- 1 Allocation contigue
- 2 Allocation chaînée
- 3 Allocation indexée

1 - Allocation contigue

Chaque fichier est stocké sur des blocs contigus.

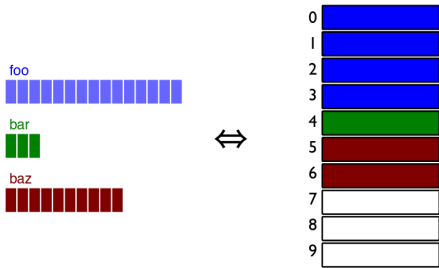


source : Rich Wolski

- Avantages : lecture des données rapide
- Inconvénients :
 - Nouveaux fichiers : il faut trouver un "trou" assez grand
 - Si la taille fichier augmente, il faut copier le fichier ailleurs.

1 - Allocation contigue

Chaque fichier est stocké sur des blocs contigus.

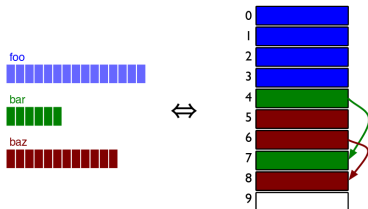


source : Rich Wolski

- **Avantage** : lecture des données rapide
- **Inconvénients** :
 - Nouveaux fichiers : il faut trouver un "trou" assez grand
 - Si la taille fichier augmente, il faut copier le fichier ailleurs.

2 - Allocation chaînée

- Les fichiers peuvent être découpés ("fragmentés"), on ajoute à la fin de chaque bloc un **lien** indiquant la localisation de la suite du fichier.
- Le répertoire contient pour un fichier la localisation du 1^{er} bloc.



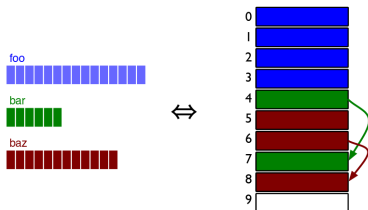
source : Rich Wolski

- Avantage : plus besoin de copie de fichier si plus de place
- Inconvénient : impossible d'accéder directement à un endroit quelconque du fichier : il faut partir du début

Implémentations : FAT (*File Allocation Table*)

2 - Allocation chaînée

- Les fichiers peuvent être découpés ("fragmentés"), on ajoute à la fin de chaque bloc un **lien** indiquant la localisation de la suite du fichier.
- Le répertoire contient pour un fichier la localisation du 1^{er} bloc.



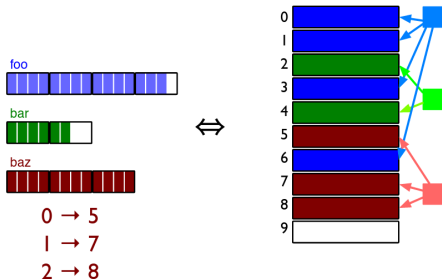
source : Rich Wolski

- **Avantage** : plus besoin de copie de fichier si plus de place
- **Inconvénient** : impossible d'accéder directement à un endroit quelconque du fichier : il faut partir du début

Implémentations : FAT (*File Allocation Table*)

3 - Allocation indexée

Une structure d'index mémorise pour chaque fichier la liste et l'ordre des blocs le composant.



source : Rich Wolski

- **Avantage** : fragmentation réduite et accès direct à un endroit quelconque du fichier.
- **Inconvénient** : réduction de l'espace utile du disque.

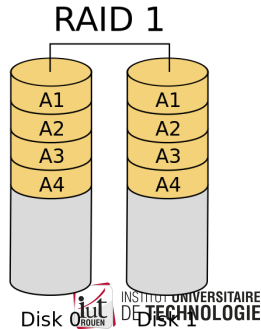
Implémentations : Linux ext2, ext3, ...

Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur**
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - **Gestion de la fiabilité**
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

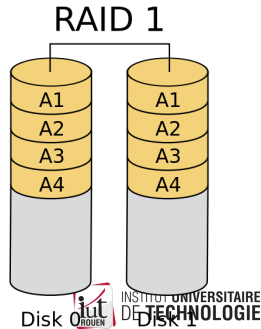
Fiabilité

- Un disque dur classique (mécanique) est susceptible de tomber à n'importe quel moment
→ en production, il faut des solutions !
- Solution 1 : sauvegardes (très) régulières
Question : quel intervalle ? une par jour ? par heure ? par minute ?
inconvenient : double perte de temps :
 - durée de la copie (même si copie **incrémentale**)
 - durée de la **restauration** des données en cas de panne
- Solution 2 : système de **redondance** des données : en cas de panne, aucune perte de données.
Remplacement du DD défaillant sans interruption de production.
 - L'OS ne voit que un seul disque logique.
 - La redondance est gérée au niveau de la baie de stockage.



Fiabilité

- Un disque dur classique (mécanique) est susceptible de tomber à n'importe quel moment
→ en production, il faut des solutions !
- Solution 1 : sauvegardes (très) régulières
Question : quel intervalle ? une par jour ? par heure ? par minute ?
inconvenient : double perte de temps :
 - durée de la copie (même si copie **incrémentale**)
 - durée de la **restauration** des données en cas de panne
- Solution 2 : système de **redondance** des données : en cas de panne, aucune perte de données.
Remplacement du DD défaillant sans interruption de production.
 - L'OS ne voit que un seul disque logique.
 - La redondance est gérée au niveau de la baie de stockage.



Sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 **Système de fichier : vue utilisateur**
 - Partionnement
 - Structure hierarchique
- 5 Fichiers sous Linux

Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur**
 - **Partitionnement**
 - Structure hierarchique
- 5 Fichiers sous Linux

Organisation en partitions

- Un disque est découpé en **partitions**, pouvant être formatées indépendamment.

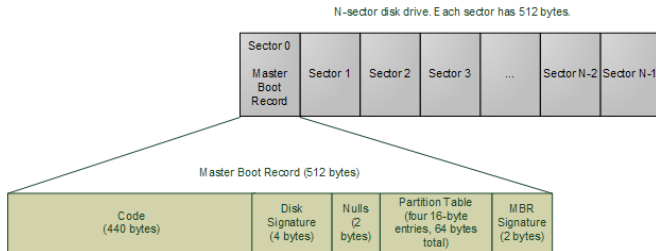
Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sda1	ext4	/		117.19 GiB	37.95 GiB	79.24 GiB	boot
/dev/sda2	ext3	/mnt/sda2	sda2	337.42 GiB	125.14 GiB	212.28 GiB	
/dev/sda3	linux-swap			11.15 GiB	0.00 B	11.15 GiB	

0 operations pending

- Chaque partition sera considérée comme un disque distinct et doit être **formatée**.
 - Linux : on lui associe un **point de montage**
 - Windows : on lui associe une lettre de lecteur logique (C :, D :, ...)

Partitions et MBR/GPT

- Les informations sur les partitions sont stockées dans une zone spéciale du disque (LBA 0) :
 - depuis 1980 : *Master Boot Record* (MBR) : 512 octets, contenant du code exécutable et la table des partitions ;
 - en voie de remplacement par *GUID Partition Table*¹ (GPT) : autorise des partitions de plus grande taille.



1. GUID : *Globally Unique identifier* : Identifiant unique

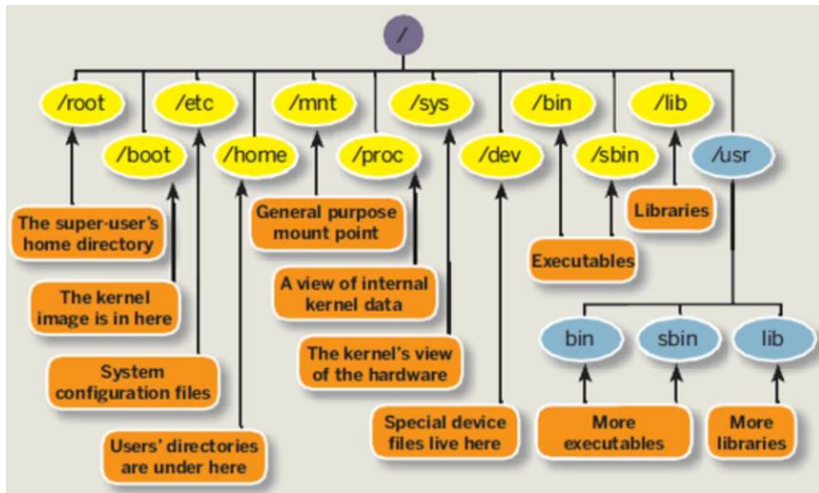
Sous-sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur**
 - Partitionnement
 - Structure hierarchique**
- 5 Fichiers sous Linux

Vu de l'utilisateur

- Système de fichier = structure hiérarchique codifiée, de type "arbre".
- Chaque interaction d'un utilisateur (via GUI ou CLI) se fait à **un** emplacement dans cet arbre (notion de "répertoire courant").
- Certains répertoires sont
 - réservés à l'OS ;
 - disponibles pour l'utilisateur.
- Chaque OS a son organisation propre :
 - Linux : le *Filesystem Hierarchy Standard* (FHS) normalise les noms des différents répertoires.
 - Windows : MS impose son organisation, mais laisse beaucoup de libertés. ("C:\Program Files", "C:\Program Files (x86)", ...)
- Attention : séparateur différent !
 - Linux & Mac : un/chemin
 - Windows : un\chemin

Linux : FHS



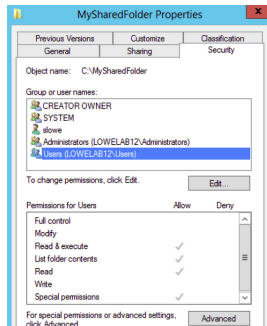
Permissions sur les fichiers

- Un système de fichiers peut mémoriser des autorisations sur les fichiers.
- Implique que les utilisateurs soient **authentifiés**.
- Les utilisateurs sont regroupés dans des groupes et les permissions peuvent être définies pour des groupes.

Historiquement :

- Windows et FAT32 : pas de possibilité de gestion des permissions
- Linux : implémentation d'un système simple hérité de UNIX :
 - permissions : lire/écrire/exécuter
 - utilisateurs : propriétaire, groupe et tout le monde

- Aujourd'hui : *Access Control List (ACL)*, activé en standard (Windows : NTFS, Linux : ext3/ext4).
- Système permettant une gestion plus fine des permissions (autoriser et interdire, autorisation par groupes, etc.)



Identification du type de fichier par l'OS

- Les fichiers ne sont pas tous du même type : programme exécutable, texte, son, vidéo, fichier de données binaire ou textuel, etc.
- La nature du fichier permet d'y associer des applications par défaut (GUI).
 - fichier texte → éditeur
 - fichier son ou vidéo → lecteur multimédia
 - document type bureautique → application bureautique (traitement de texte, tableur, ...)
 - fichier pdf → lecteur pdf
 - ...
- Windows : identification du type de fichier **uniquement** par l'extension → si on change l'extension, l'OS ne sait plus quoi faire !
- Linux : identification du type de fichier par l'extension **ET** par le contenu (premiers octets du fichier)

Sommaire

- 1 Introduction
- 2 Système de fichiers : vue bas niveau
- 3 Structure d'un disque dur
 - Présentation et historique
 - Caractéristiques
 - Organisation bas niveau
 - Technique d'allocation de blocs
 - Gestion de la fiabilité
- 4 Système de fichier : vue utilisateur
 - Partitionnement
 - Structure hierarchique
- 5 Fichiers sous Linux**



Fichier sous Linux/Unix

- La commande `ls -l` affiche toutes les informations sur les fichiers du dossier courant.

```
drwxr-xr-x  3 skramm  users          4096 avril  8 10:48 websites
-rw-r--r--  1 skramm  users          4542 mai    3 14:19 weka.log
```

Dans l'ordre :

- Codification des permissions
Le premier bit indique si c'est un dossier (d), un lien (l) ou un fichier régulier (-)
- Nombre de fichiers (pour les répertoires)
- Propriétaire & groupe
- Taille (pour les fichiers)
- Date de dernière modification & nom.

Droits des fichier sous Linux/Unix

- Les droits sont codifiés par 3 vecteurs de 3 bits
- 3 vecteurs pour
 - *User* (u) : propriétaire du fichier
 - *Group* (g) : le groupe auquel appartient le propriétaire
 - *Others* (o) : les autres, tout le monde
- 3 bits pour chaque droit :
 - r : droit en lecture (*Read*)
 - w : droit en écriture (*Write*)
 - x : droit en exécution (*eXecute*)²
- Ces droits peuvent être encodés avec une valeur exprimée en **octal** (3 symboles dans {0,...,7})

User			Group			Others		
r	w	x	r	w	x	r	w	x
1	1	1	1	0	1	0	0	1
7			5			1		

2. Pour les répertoires, donnera le droit de "traverser"

Edition des permissions

- La commande `chmod` permet de modifier les permissions
- Deux façons de l'utiliser :
 - ① en donnant la valeur en octal : impose les permissions de façon globale pour le fichier.
Exemple : `chmod 764 fichier` → fixe les droits à `rwrxw-r-`
 - ② en n'éditant qu'une partie des permissions en spécifiant de façon symbolique quelle permission et à qui on les donne avec la syntaxe
`chmod [u g o a] [+ - =] [r w x] fichier`
 - Exemple 1 :
`chmod u+x fichier` → "ajouter le droit d'exécution au propriétaire du fichier"
 - Exemple 2 : on peut regrouper les catégories :
`chmod ug+x fichier` → "ajouter le droit d'exécution au propriétaire et au groupe"
- Les deux peuvent être équivalents. Exemple :
`chmod 764 fichier` ↔ `chmod u=rwx,g=rw,o=r fichier`

Fichiers et *i-nodes*

- Les partitions Linux contiennent une table d'index de taille prédéfinie, qui contient les **i-nodes** :
- **i-node** : structure de données contenant
 - l'emplacement du fichier sur le disque
 - les métadonnées du fichiers (droits d'accès, timestamps, taille fichier, ...)
 - *User* (UID) et *Group* (GID) d'appartenance
 - Un compteur de liens
- Le **nom** du fichier est stocké dans le répertoire
- En pratique, un fichier est une **association** nom ↔ *i-node*

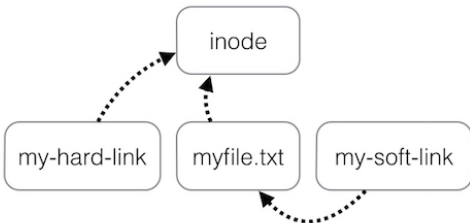
Conséquence de cette approche :

- un fichier peut avoir plusieurs noms (*hardlinks*) : noms différents associés avec le même *i-node*.
- on peut définir des liens symboliques sur un fichier (*softlinks*).

```
sk@PORT-LITIS-SKRAMM:~$ ls -l /bin
total 9924
-rwxr-xr-x 1 root root 1021112 oct. 7 2014 bash
-rwxr-xr-x 1 root root 31152 oct. 21 2013 bunzip2
-rwxr-xr-x 1 root root 1918032 nov. 14 2013 busybox
-rwxr-xr-x 1 root root 31152 oct. 21 2013 bzip2
lrwxrwxrwx 1 root root 6 mars 10 2015 bzip2 -> bzdifff
-rwxr-xr-x 1 root root 2140 oct. 21 2013 bzdifff
lrwxrwxrwx 1 root root 6 mars 10 2015 bzip2 -> bzip2
-rwxr-xr-x 1 root root 4877 oct. 21 2013 bzip2
lrwxrwxrwx 1 root root 6 mars 10 2015 bzip2 -> bzip2
-rwxr-xr-x 1 root root 3642 oct. 21 2013 bzip2
-rwxr-xr-x 1 root root 31152 oct. 21 2013 bzip2
```

Liens sur des fichiers

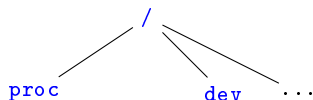
- Conséquence de cette approche : on peut définir des **liens** sur un fichiers
- deux types de liens :
 - lien symbolique (*soft link*) : pointe sur le nom du fichier
 - lien physique (*hard link*) : pointe sur le inode



Systèmes de fichiers virtuels

Dans le FHS, certains répertoires de la racine sont **virtuels**

- Ne correspondent à aucun fichier réel (taille : 0 sur le disque)
- Sert d'**interface** pour accéder aux données du noyau.



- `/dev` : contient les pilotes de périphériques. Permet d'envoyer et recevoir via le concept de fichier. Deux types : "octet" (*Character*) et "bloc"
- `/proc` : Information sur les processus et le noyau