

Fichiers & Système de fichiers

M1105 - Systèmes d'exploitation

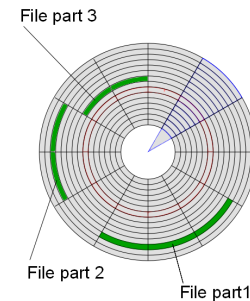
Sebastien.Kramm@univ-rouen.fr

IUT de Rouen, dépt. Réseaux & Télécoms

Version du 19 novembre 2017

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 Modifi
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	7 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	7 items	Folder	mar. 22 mars
run	40 items	Folder	lun. 19 sept.

Système de fichiers : structure

Différents formats 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 ?

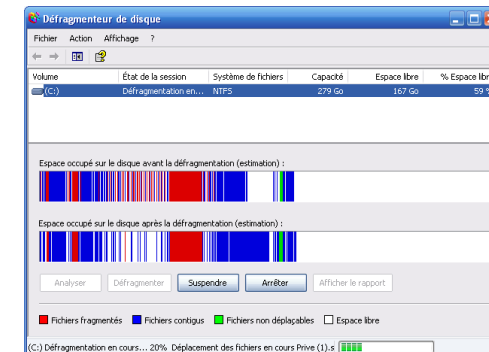
Systèmes de fichier journalisé

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 : (déplacements multiples de la tête pour la lecture d'un fichier).
⇒ nécessité de procéder régulièrement à une **défragmentation**.



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

Limitations

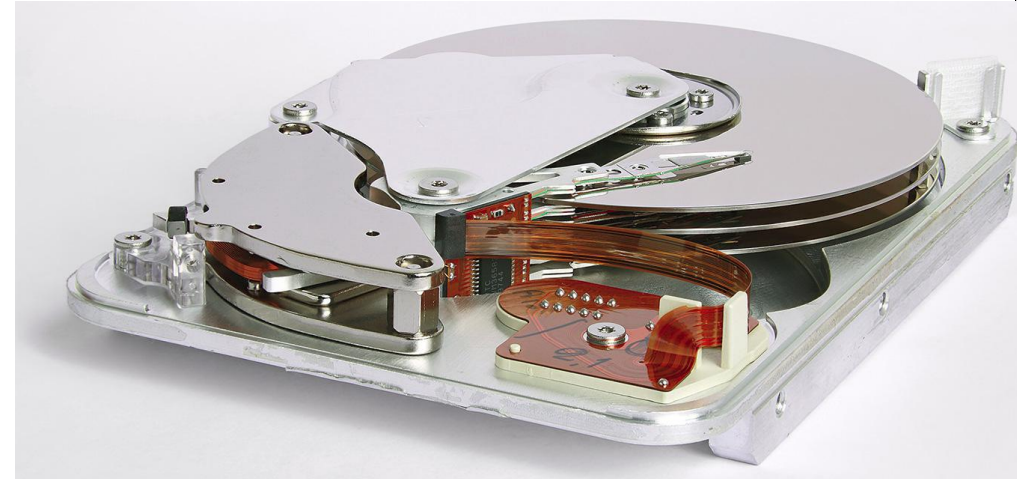
- ▶ Chaque système de fichier a des limitations
 - ▶ taille max du disque, du fichier, de la partition, ...;
 - ▶ nombre de caractères en encodage du nom;
 - ▶ nombre de fichiers max. par répertoire;
 - ▶ gestion des permissions;
 - ▶ datation;
 - ▶ etc.
- ▶ Exemples :
 - ▶ FAT : noms de fichiers limités 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 !

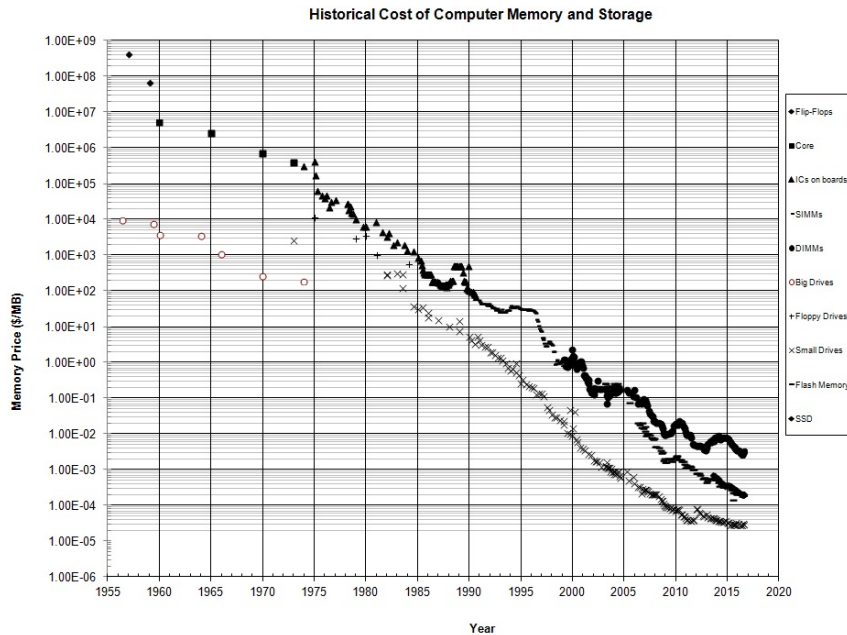
HD aujourd'hui

Dimensions courantes : 2,5 " ou 3,5 "



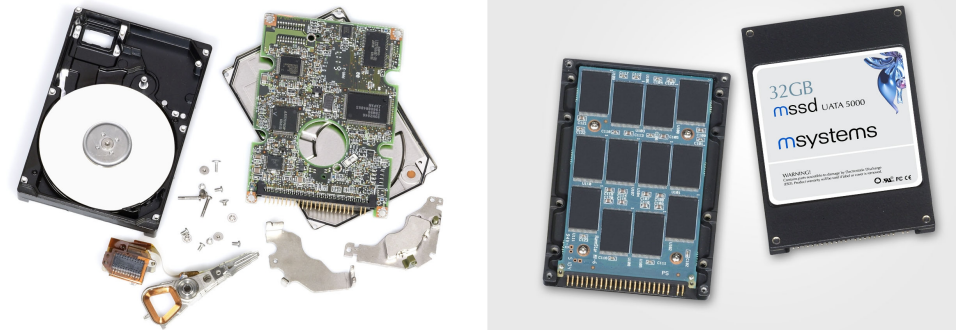
(rappel : 1 pouce = 25,4 mm)

Evolution du coût du Mo (2016/08)



Source : John C. McCallum

HD vs. SSD

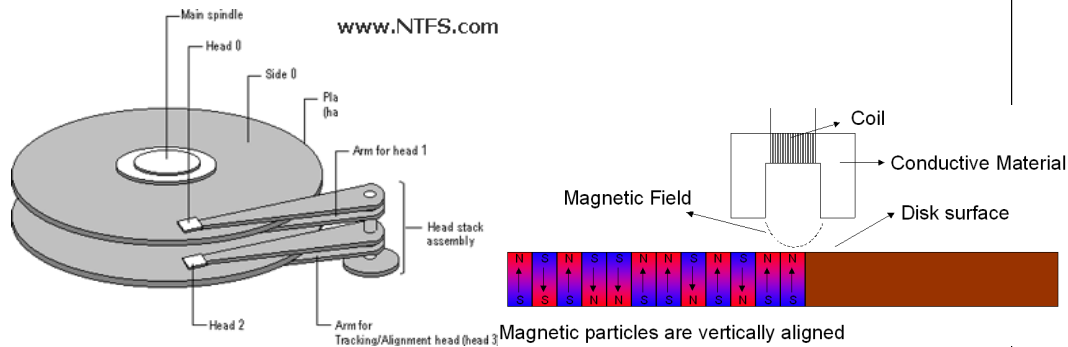


SSD (Solid State Drive) : disque sans mécanique ni mouvements, composé de mémoire type "Flash".

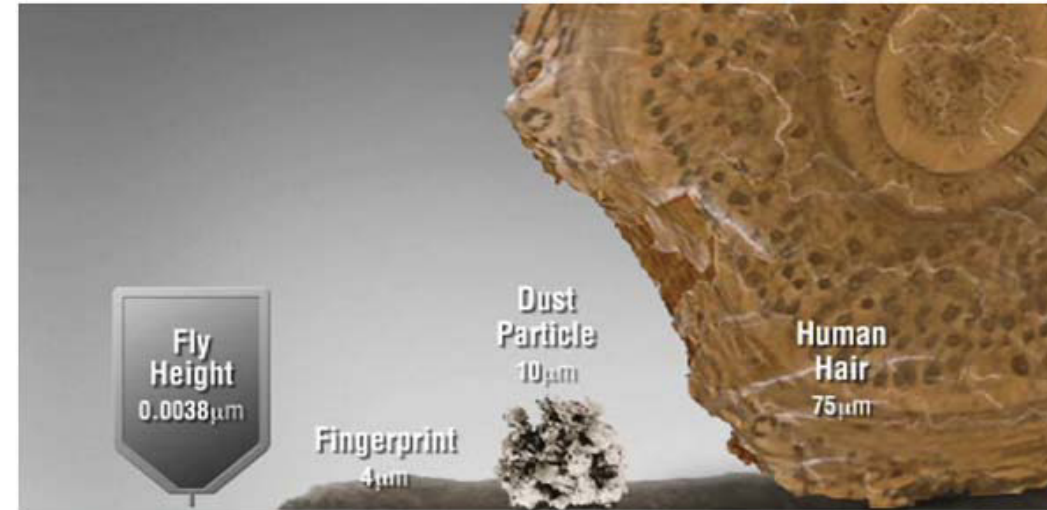
Capacités disponibles (2017-Q4) : 32 GB → 2 TB

Organisation physique d'un disque

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



Dimensions



source : seagate.com

Spécifications

Exemple : Seagate Barracuda 3TB

- 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

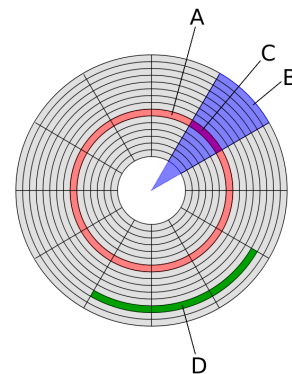
Property	Measurement
Platters	3
Heads	6
Rotation Speed	7200 rpm
Average Seek Delay	8ms
Average Rotation Latency	4ms
Bytes/Sector	512
Sectors/Track	63

Les constructeurs donnent aussi le débit moyen crête et continu.

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

Géométrie du disque

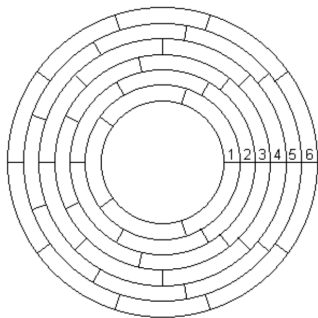
- Le disque est découpé en cylindres, *tracks* (pistes) et secteurs
- secteur : plus petite unité d'allocation contient les données utiles, plus des métadonnées, en particulier des infos permettant la correction d'erreur (ECC : *Error Correction Code*)
La taille du secteur est **fixée** par le constructeur
- **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)

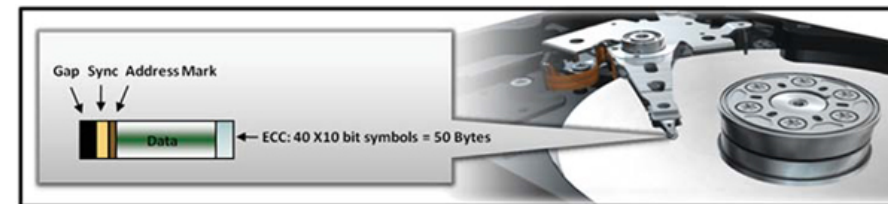
En réalité

- ▶ 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 Address*), 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"

Gestions de l'allocation de blocs pour des fichiers

Problématiques : soit un disque dur partiellement occupé.

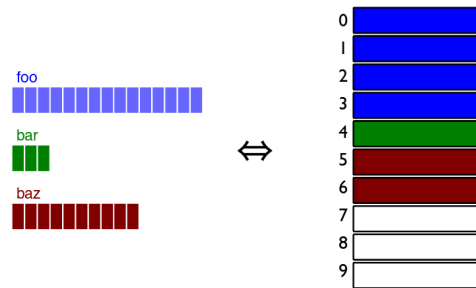
- ▶ Je veux y stocker un nouveau fichier de grande taille. Où le placer ?
- ▶ Un des (gros) fichiers existant voit sa taille doubler. Comment stocker ce fichier ?

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.

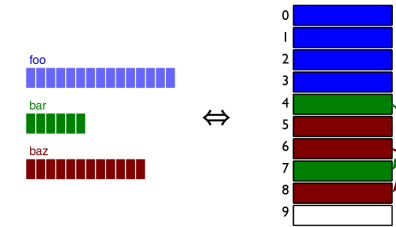


source : Rich Wolski

- ▶ 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.



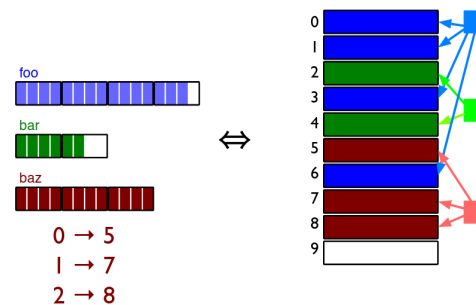
source : Rich Wolski

- ▶ 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*)

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, ext4, ...

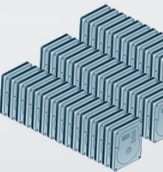
Fiabilité

- ▶ Un disque dur classique (mécanique) est susceptible de tomber à n'importe quel moment



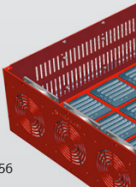
BACKBLAZE STO
MAJOR COMPONENTS LIST

45 HARD DRIVES - \$5400



CUSTOM BUILT CASE - \$758

- 6 FANS
- 1 BOOT DRIVE
- 9 MULTIPLIER BACKPLANES



Solutions ?

- ▶ Solution 1 : sauvegardes (très) régulières

Question : quel intervalle ? une par jour ? par heure ? par minute ?

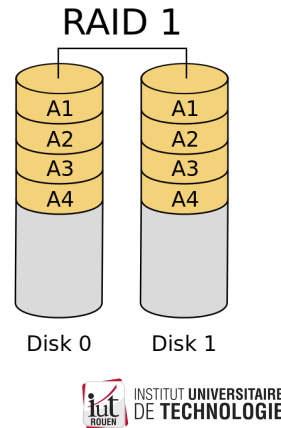
inconvenient : double perte de temps :

- ▶ 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.



Techniques RAID : *Redundant Array of Independent Disks*

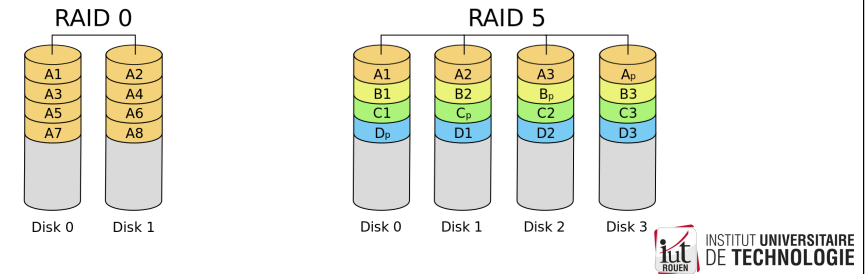
- ▶ Idée : utiliser plusieurs disques gérés comme un disque unique

- ▶ Objectifs : augmentation

- ▶ de la performance, et/ou
- ▶ de la fiabilité, et/ou
- ▶ de la capacité, et/ou
- ▶ de la tolérance aux pannes.

- ▶ Niveaux de RAID standard :

- ▶ RAID-0 : augmentation de performance, en utilisant n disques en parallèles. Aucune sécurité.
- ▶ RAID-1 : augmentation de fiabilité : réplication totale des données sur les disques
- ▶ RAID-5 : technique plus évoluée : augmentation de fiabilité avec optimisation de l'espace, mais minimum 3 disques.



Données SMART

- ▶ *Self-Monitoring, Analysis and Reporting Technology* : données fournies par les disques.
- ▶ Des dizaines d'attributs mesurables, ainsi qu'un seuil indicatif : en dessous (ou au dessus, selon l'attribut), la probabilité de panne **augmente**.
- ▶ Les contrôleurs de disques proposent aussi des tests permettant d'évaluer son état.

Device: /dev/sda Model: ST500LM000-1EJ162

Identity Attributes Capabilities Error Log Self-test Logs Perform Tests

SMART Attributes Data Structure revision number: 10

ID	Name	Failed	Norm-ed value	Worst	Threshold	Raw value	Type	Update
1	Raw Read Error Rate	never	113	99	6	56523016	pre-failure	contir
3	Spin-Up Time	never	98	98	85	0	pre-failure	contir
4	Start / Stop Count	never	99	20	1651		old age	contir
5	Reallocated Sector Count	never	100	100	10	0	pre-failure	contir
7	Seek Error Rate	never	73	60	30	23012513	pre-failure	contir
9	Power-On Time	never	96	96	0	3847	old age	contir
10	Spin-Up Retry Count	never	100	97	0		pre-failure	contir
12	Power Cycle Count	never	99	99	20	1577	old age	contir
184	End to End Error	never	100	100	99	0	old age	contir
187	Reported Uncorrectable	never	100	100	0	0	old age	contir
188	Command Timeout	never	100	99	0	4	old age	contir
189	High Fly Writes	never	100	100	0	0	old age	contir
190	Airflow Temperature	never	62	48	45	38 (Min/Max 34/38)	old age	contir
191	G-Sense Error Rate	never	100	100	0	0	old age	contir
192	Head Retract Cycle Count	never	100	100	0	1	old age	contir

INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

Données SMART

- ▶ Attention : les données SMART sont indicatives :

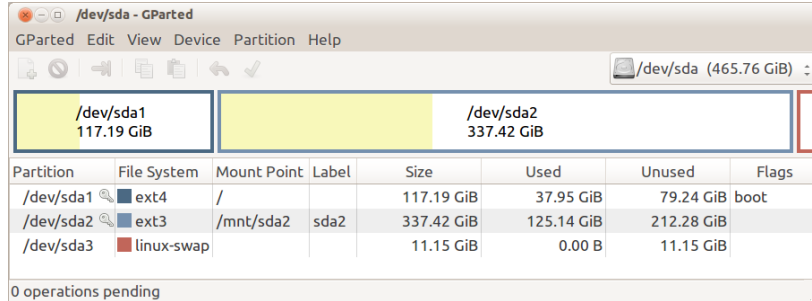
Etude (2016/10) : 23% de disques tombent en panne sans erreur préalable !

A lire :

- ▶ <http://www.hardware.fr/news/14806/pannes-disques-durs-donnees-smart.html>
- ▶ <http://www.tomshardware.fr/articles/fiabilite-disque-dur-ssd,2-762.html>

Organisation en partitions

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



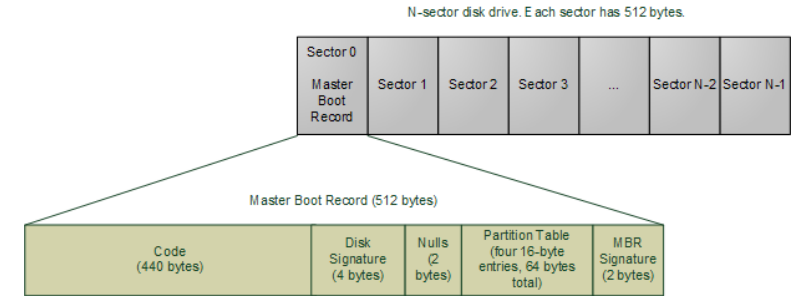
Partition	File System	Mount Point	Label	Size	Used	Unused	Flags
/dev/sda1	ext4	/		117.19 GiB	37.95 GiB	79.24 GiB	boot
/dev/sda2	ext3	/mnt/sda2	sda2	337.42 GiB	125.14 GiB	212.28 GiB	
/dev/sda3	linux-swap			11.15 GiB	0.00 B	11.15 GiB	

- ▶ Chaque partition sera considérée par le système de fichier 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 : **MBR** (*Master Boot Record*) : 512 octets, contenant du code exécutable et la table des partitions ;
 - ▶ Remplacé aujourd'hui par **GPT** (*GUID Partition Table*¹) : autorise des partitions de plus grande taille.

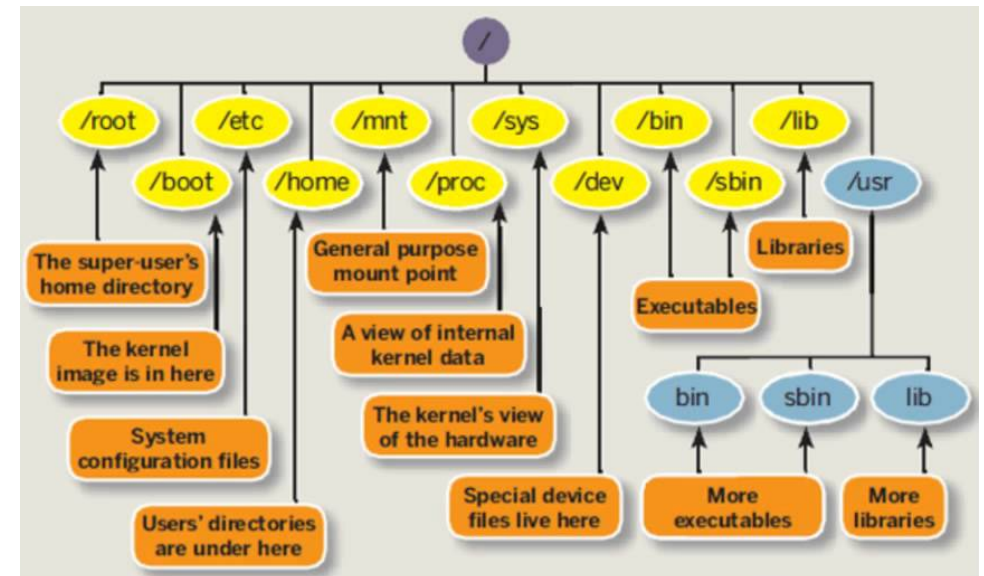


1. GUID : *Globally Unique identifier* : Identifiant unique

Vu de l'utilisateur

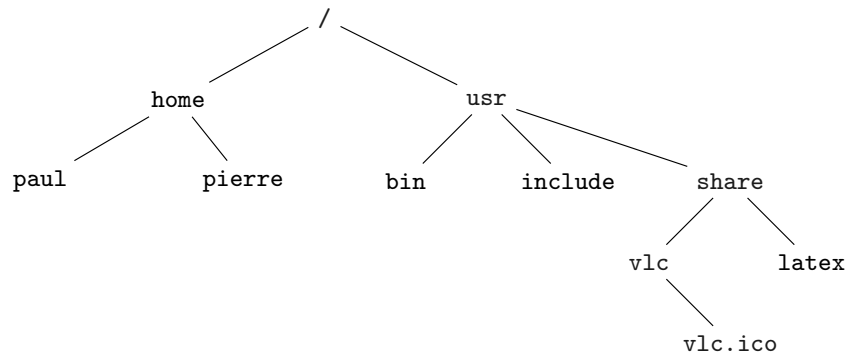
- ▶ Système de fichier = structure hiérarchique codifiée, de type "arbre".
- ▶ Chaque interaction d'un utilisateur (via GUI ou CLI) se fait à **un** emplacement dans cet arbre (notion de "répertoire courant").
- ▶ Certains répertoires sont
 - ▶ réservés à l'OS ;
 - ▶ disponibles pour l'utilisateur.
- ▶ 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



Chemin absolu

- ▶ Chaque fichier ou répertoire est désigné de façon unique par son **chemin absolu** qui part de la racine.

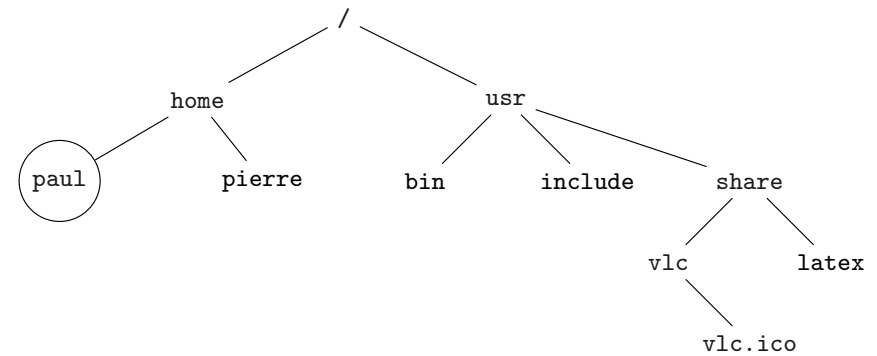


⇒ Le fichier `vlc.ico` est accessible via le chemin `/usr/share/vlc/vlc.ico`

- ▶ Windows : le chemin absolu **doit** inclure une lettre d'unité :
`c:\Program Files\Microsoft\bidule\truc\bin\machin.exe`

Chemin relatif

- ▶ Depuis un dossier courant quelconque, on peut **remonter** dans la hiérarchie avec ..



⇒ Depuis le dossier `paul`, on accède au fichier `vlc.ico` avec le chemin `../usr/share/vlc/vlc.ico`

- ▶ Windows : pas de chemins relatifs entre unité disques...

Visualisation : attention !

- ▶ Certains OS/explorateurs de fichier **travestissent** la réalité : certains dossiers sont montrés avec un nom différent :

- ▶ Exemple :

- ▶ Nom affiché / Nom réel



- ▶ la seule réalité fiable :



- ▶ on peut aussi "cacher" des fichiers
 - ▶ Linux : nom commençant par un "."
 - ▶ Windows : attribut "hidden" (métadonnées)

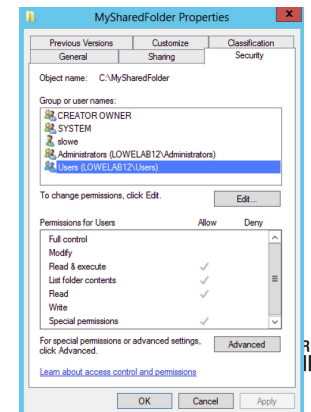
Permissions sur les fichiers

- ▶ 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)

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 :

1. Codification des permissions
Le premier bit indique si c'est un dossier (d), un lien (l) ou un fichier régulier (-)
2. Nombre de fichiers (pour les répertoires)
3. Propriétaire & groupe
4. Taille (pour les fichiers)
5. 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. forme octale (binaire) : fixe les permissions de façon globale pour le fichier.
Exemple : `chmod 764 fichier` → fixe les droits à `rwrxw-r--`
 2. forme symbolique : édite qu'une partie des permissions en spécifiant 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*

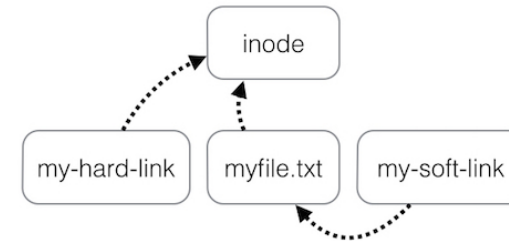
Conséquence de cette approche :

- ▶ 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*).

```
REPORT-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 1910032 nov. 14 2013 busybox
-rwxr-xr-x 1 root root  31152 oct. 21 2013 bzipcat
lrwxrwxrwx 1 root root    0 mars 10 2015 bzcomp -> bzdiff
-rwxr-xr-x 1 root root  2140 oct. 21 2013 bzdiff
lrwxrwxrwx 1 root root    0 mars 10 2015 bzgrep -> bzgrep
-rwxr-xr-x 1 root root  4877 oct. 21 2013 bzip2
lrwxrwxrwx 1 root root    0 mars 10 2015 bzgrep -> bzgrep
-rwxr-xr-x 1 root root  3642 oct. 21 2013 bzgrep
-rwxr-xr-x 1 root root  31152 oct. 21 2013 bzlib2
```

Liens sur des fichiers

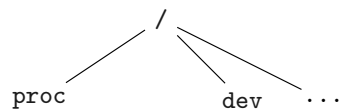
- ▶ Conséquence de cette approche : on peut définir des **liens** sur un fichiers
- ▶ deux types de liens :
 - ▶ lien symbolique (*soft link*) : pointe sur le nom du fichier
 - ▶ lien physique (*hard link*) : pointe sur le inode



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