Systèmes d'exploitation - Gestion des fichiers

Pierre Gançarski

DUT Informatique - S31

ATTENTION
Ces transparents ne sont qu'un guide du cours : de nombreuses explications et illustrations manquent.

De nombreuses précisions seront données au tableau et à l'oral pendant le cours.

Université de Strasbourg

IUT Robert Schuman
UNIVERSITÉ DE STRASBOURG

Plan

Objectifs d'un système de fichiers

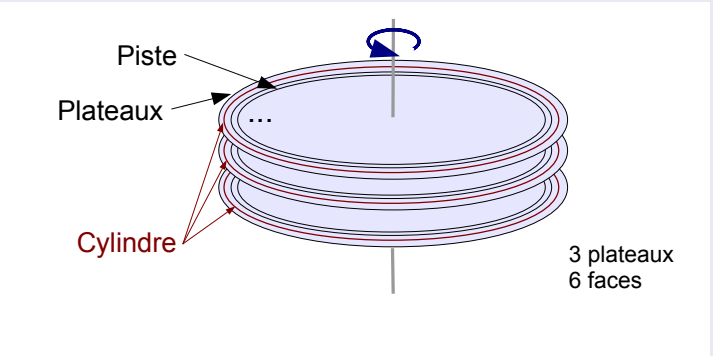
Stockage et persistance

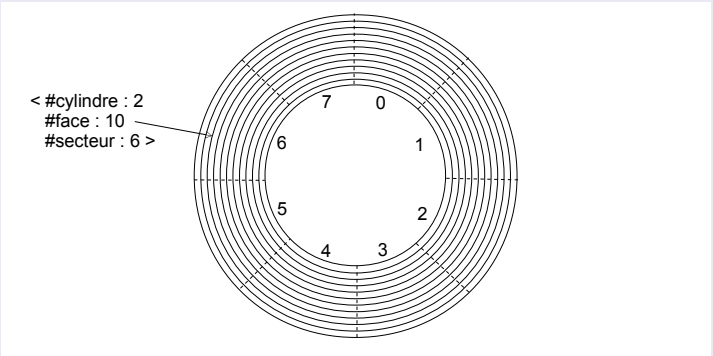
- Stockage offline (à long terme) d'une grande quantité d'information.
- Les informations doivent être persistantes.
- Plusieurs processus doivent pouvoir accéder simultanément à une même information.

Disque

Rappel

- Un disque dur est organisé en pistes formant des cylindres.
- Les pistes sont découpées en secteurs (généralement 512 octets)





- Peuvent être aussi vus comme une suite ordonnée d'adresses :

$$0 \leftrightarrows <0,0,0> ; 1 \leftrightarrows <0,0,1> \ldots 86 \leftrightarrows <3,4,6>$$

- Disque SSD : a conservé la même forme d'adressage.

Blocs/Clusters

- Bloc (Unix) Cluster (NTFS) : regroupement de secteurs contiguës
  - Réduit la taille des adresses
  - Peut amener une fragmentation interne

Fichiers

Tout est fichier

- Fichier = information sur le disque composée d'une suite d'octets.
- Un fichier possède au moins deux noms :
  - Nom symbolique : identifiant utilisé par les processus et les utilisateurs pour accéder au fichier : toto.c ; fi.dat, ...
  - Nom système (ou interne) : donne les informations permettant de connaître les blocs affectés au fichier
    - UNIX : **inode**
    - WINDOWS - NTFS : **descripteur de fichier**
- La gestion des noms symboliques et internes et leur interrelation dépend du système de gestion de fichiers (File system)

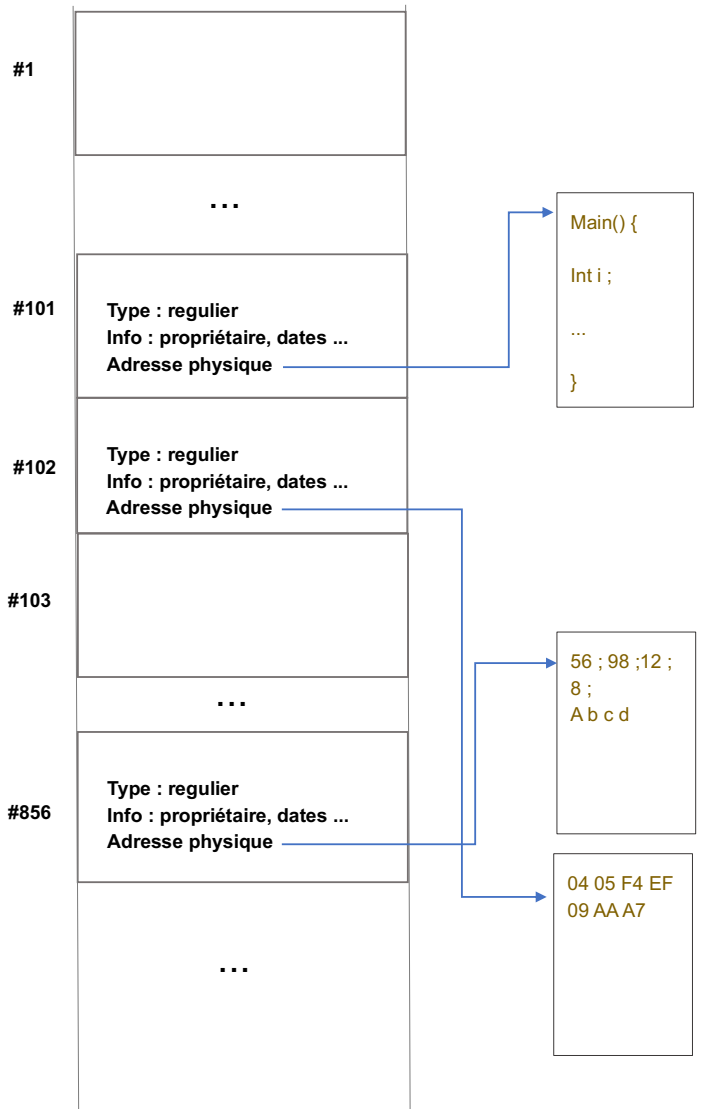
SGF : gestion des noms internes

Table des inodes/descripteurs

Chaque fichier est décrit par :

- des informations générales : propriétaire, dates ...
- un type : régulier, répertoire, spécial ...
- une "adresse" permet de connaître les blocs affectés

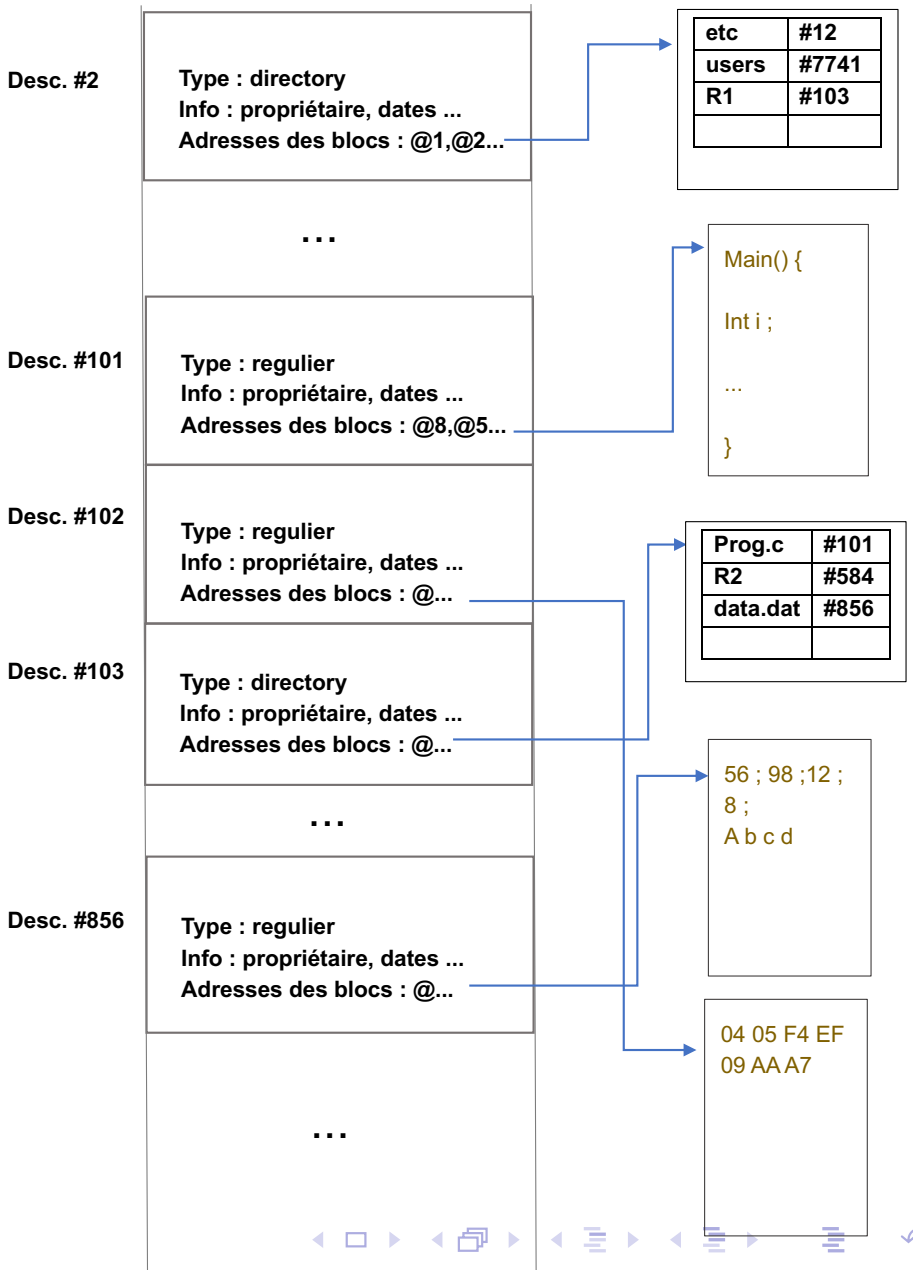
→ le format du descripteur et surtout de l'adresse physique dépend du type de système de gestion (sera vu plus tard)



Fichiers et répertoires

Tout est fichier

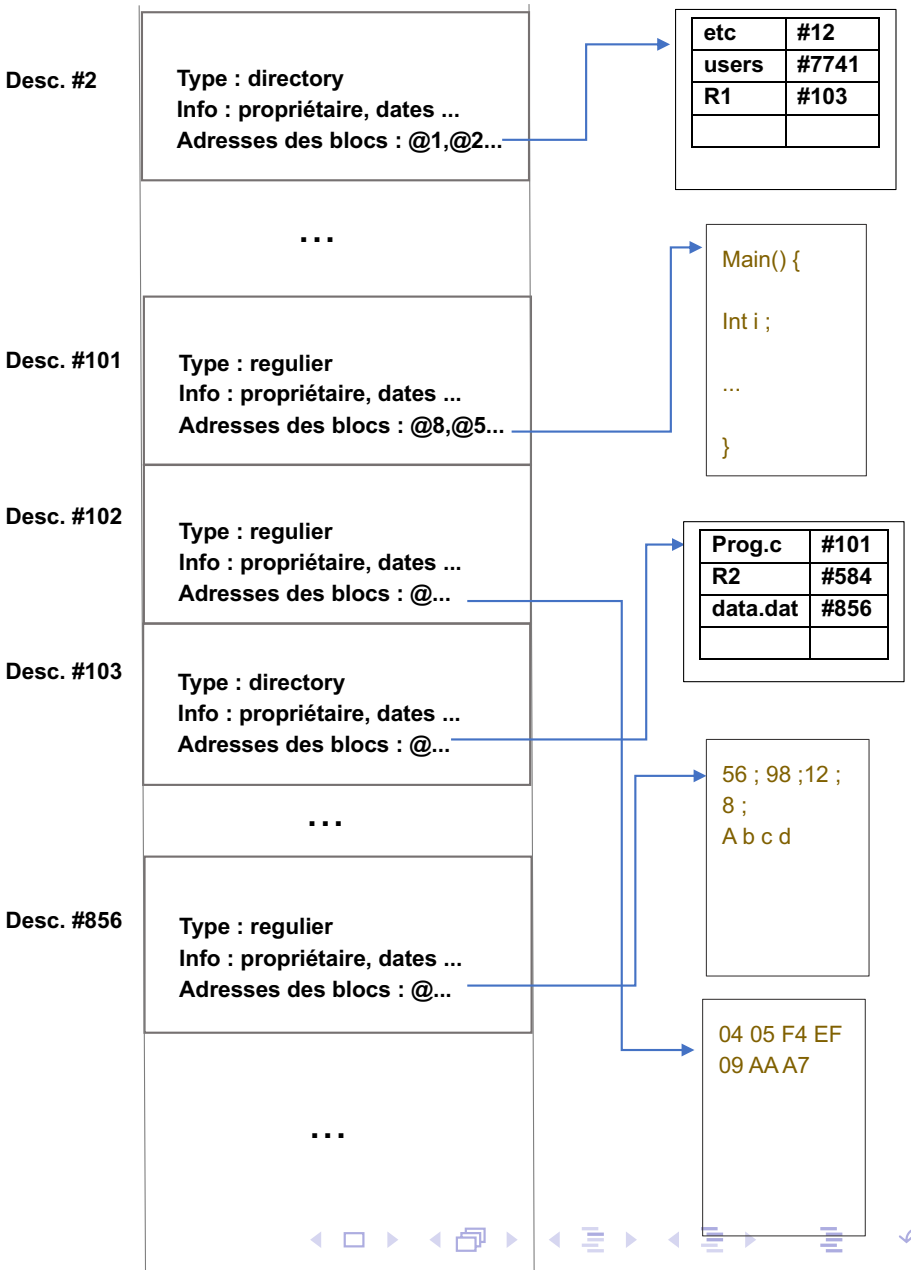
- Fichier = information sur le disque composée d'une suite d'octets.
- Répertoire = fichier contenant une liste d'associations entre des noms symboliques de fichiers et leurs noms système
- Tous les systèmes modernes reposent sur une structure de répertoires hiérarchiques



Fichiers et répertoires

Exemple

Quelles sont les différentes étapes pour réaliser la commande
`cat /R1/toto.c`
sachant que la racine est donnée par le descripteur D = 2

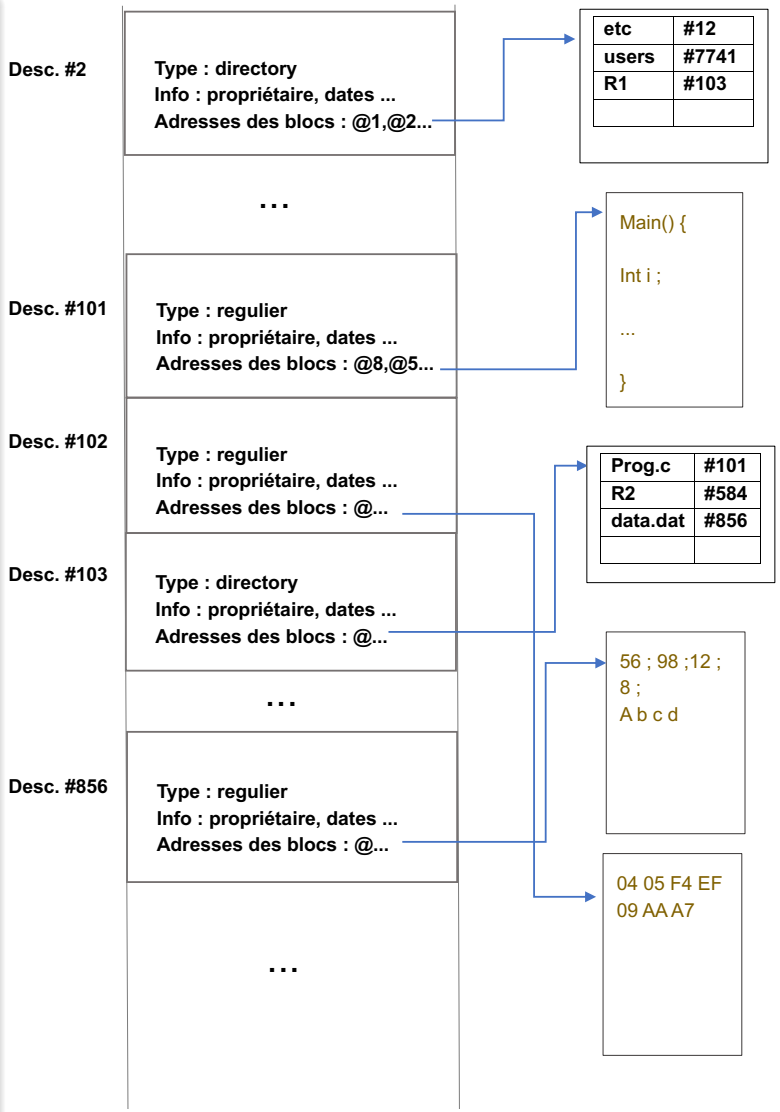


Fichiers et répertoires

Exercice

1. D = 2, F = "R1"
2. Le système lit le descriptif D
3. Il note le type T
4. Il charge le fichier correspondant
5. Si T == "Directory", D contient une table de correspondances
   - Il balaie cette table à la recherche de F
   - S'il ne le trouve pas : FIN
     Sinon D = numero de descripteur associé à F
   - Il lit la suite de la commande et recommence en 2 (→ ici D = 103 et F = "toto.c")

   Sinon il effectue le cat sur le fichier chargé.



Plan

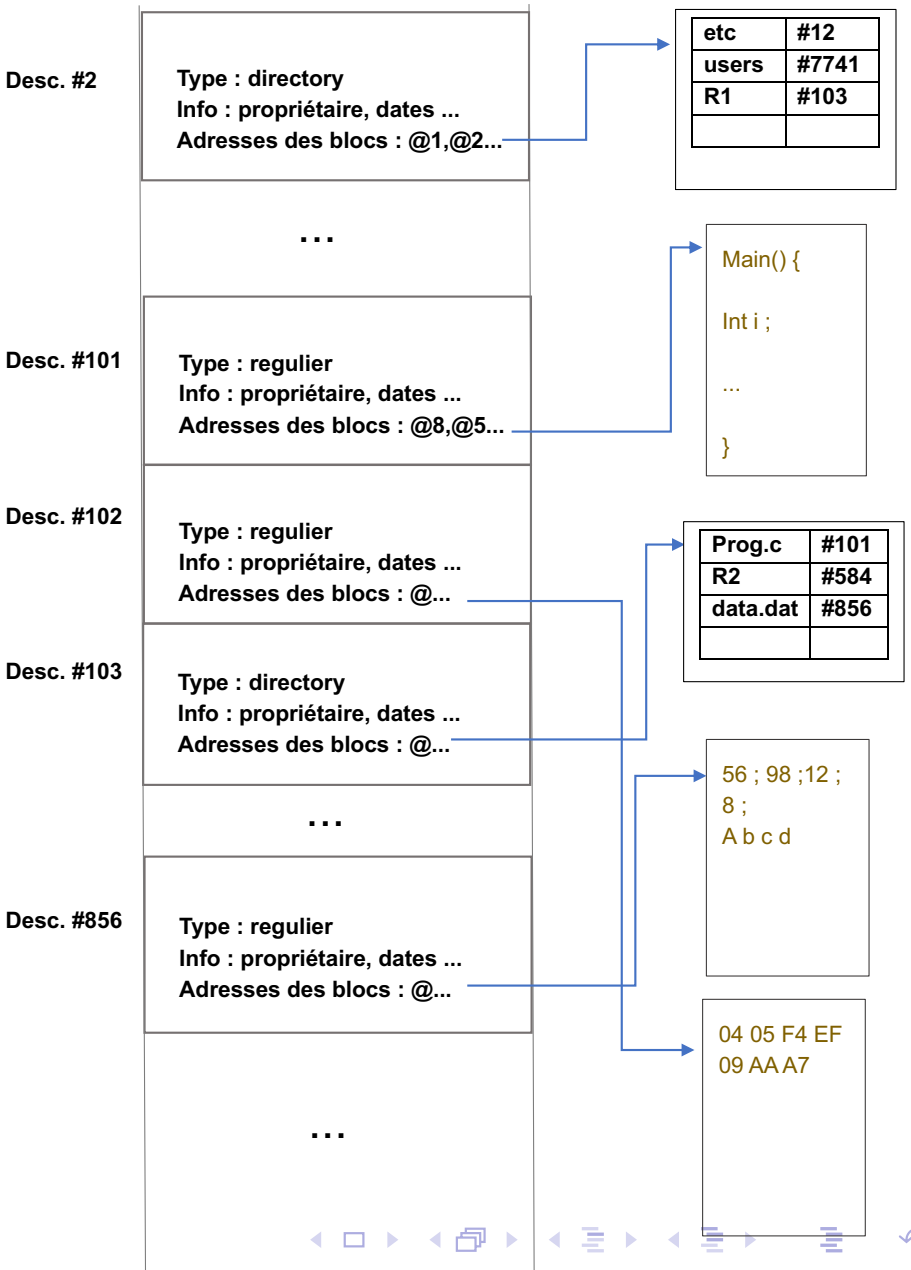
Fichiers et répertoires

Table des inodes/descripteurs

Chaque fichier est décrit par :

- des informations générales : propriétaire, dates ...
- un type : régulier, répertoire, spécial ...
- une adresse (information physique) permet de connaître les blocs affectés

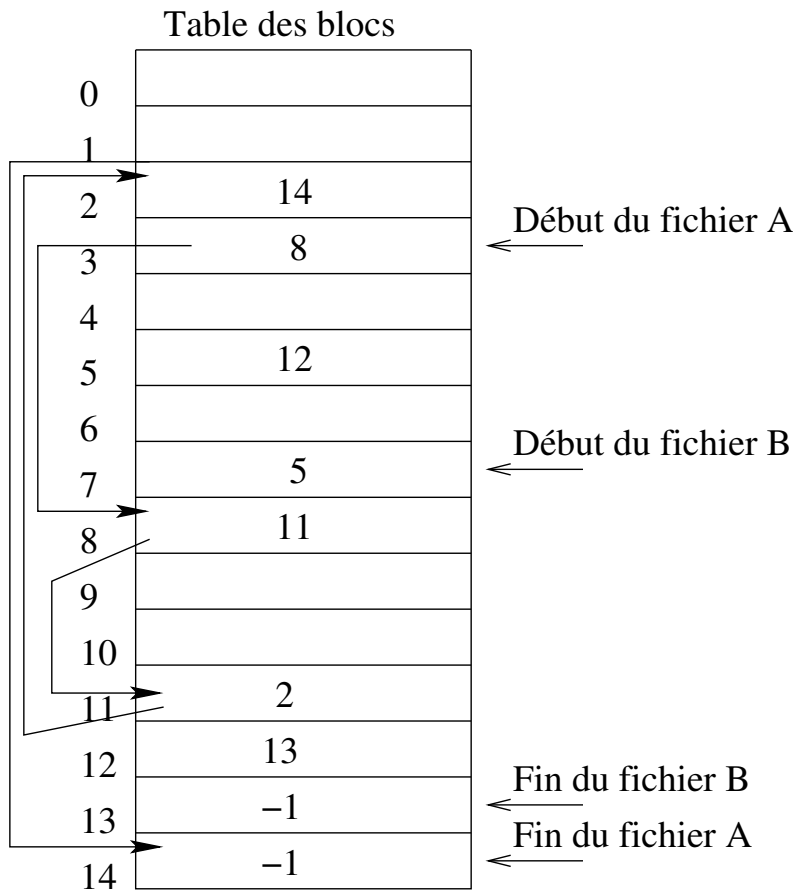
→ Comment est codée cette information physique ?



Systèmes de gestion de fichiers : MS-DOS - Windows

FAT 32 (File Allocation Table)

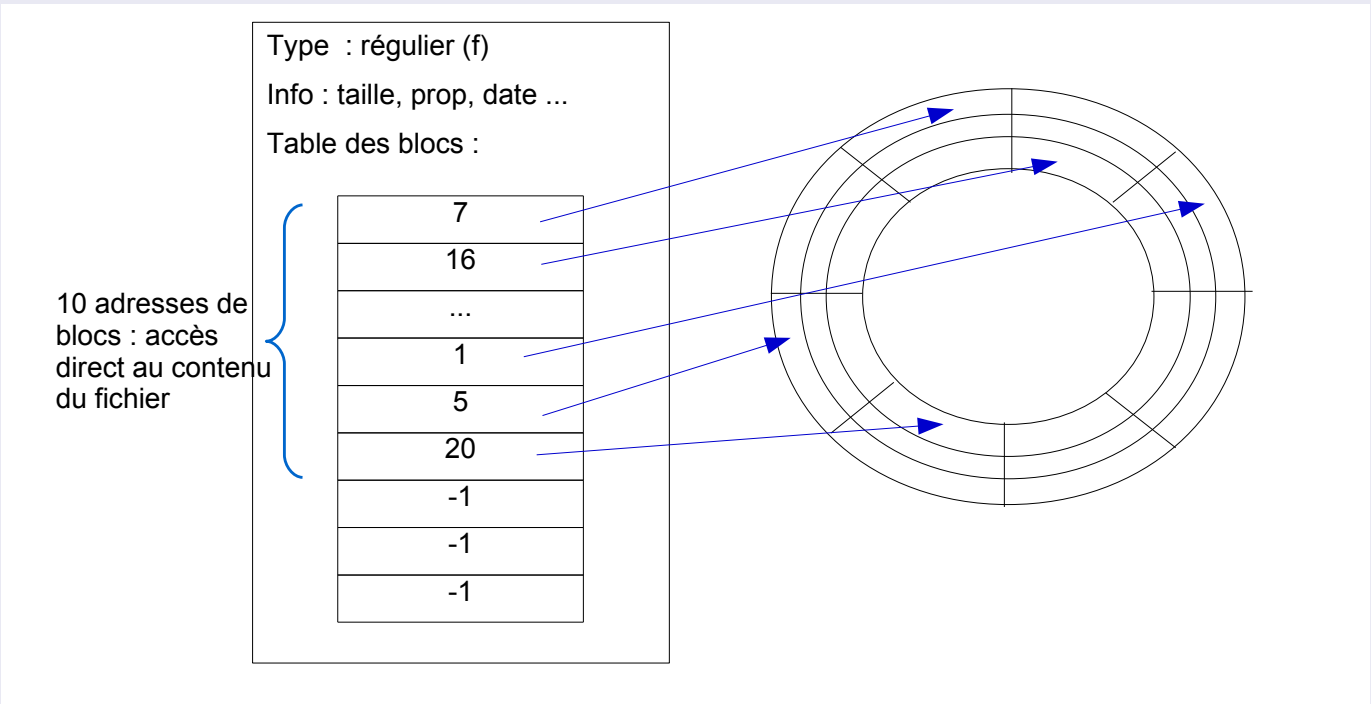
- L'adresse physique est l'indice du premier bloc dans une table des blocs
  (Fichiers A : 3 – B : 7) ,
- Avantage : La recherche des blocs peut s'effectuer en mémoire principale : rapide, pas d'accès disque.
- Inconvénient : la taille de la table = nombre de blocs sur le disque.



Systèmes de gestion de fichiers : UNIX

Unix

- L'adresse physique est donnée via une **inode** qui contient une table donnant la liste des blocs associés au fichier.

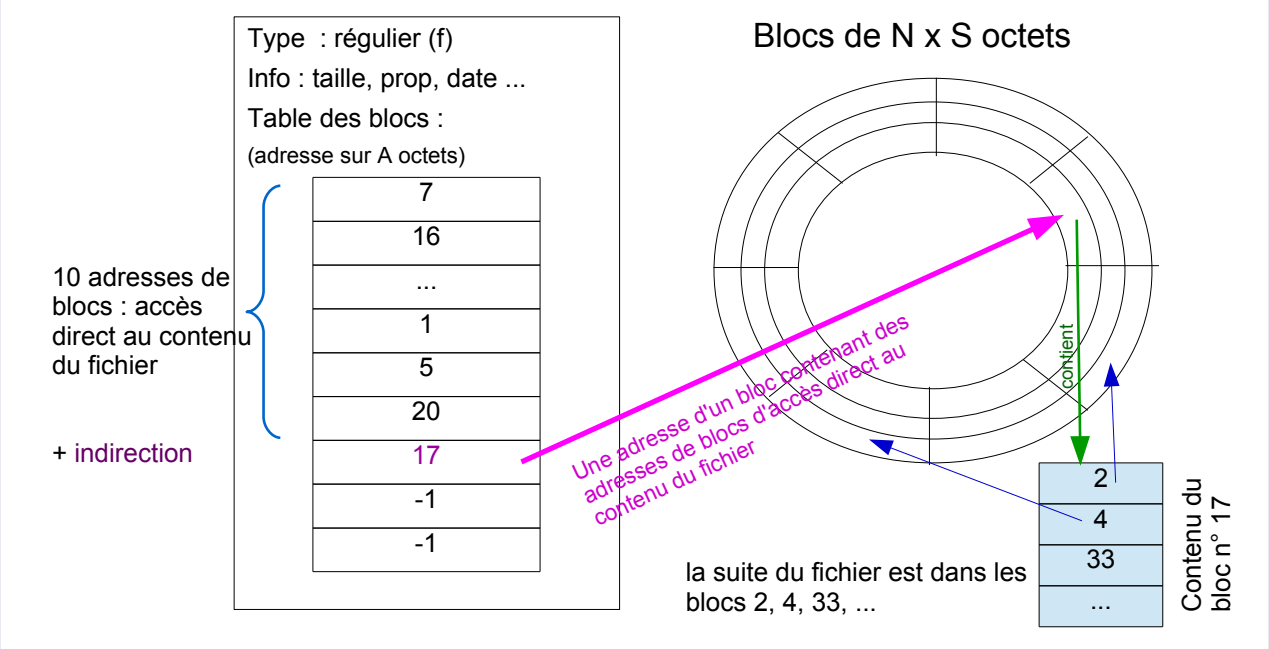


- Avantage : l'ensemble des blocs est connu en consultant l'inode. Faible encombrement en mémoire.

Politique d'allocation des blocs sur disque - type UNIX

Inodes et gros fichiers

- Pour pouvoir gérer des gros fichiers, les inodes intègrent des indirections simples, doubles (voire triples)



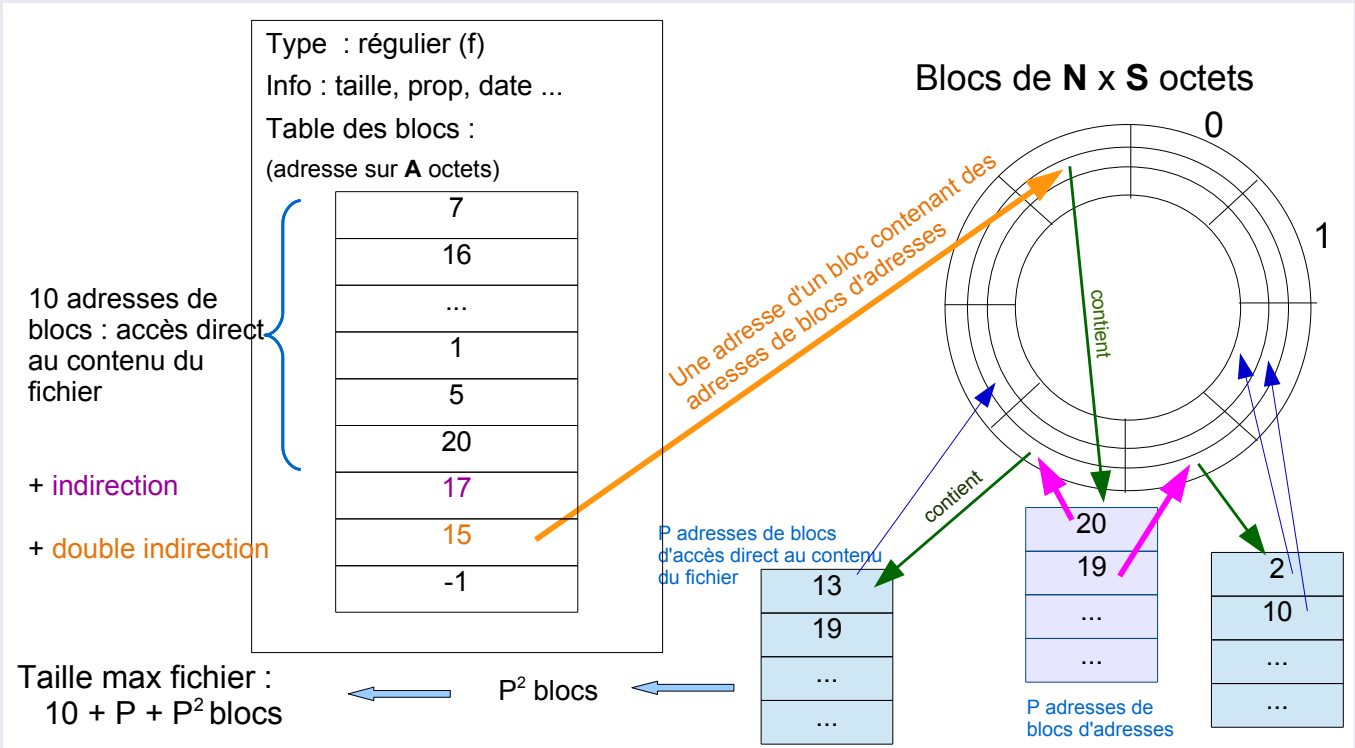
S = 512 octets, N = 8 secteurs par bloc, adresse A sur 4 octets
→ Un bloc peut contenir P = 1024 adresses
→ Un inode peut contenir 10 + P adresses = 1 034

⇒ Taille max d'un fichier = 1034 blocs = 1034 × 4Ko/bloc = 4 136Ko

Politique d'allocation des blocs sur disque - type UNIX

Inodes et gros fichiers

- Pour pouvoir gérer des gros fichiers, les inodes intègrent des indirections simples, doubles (voire triples)

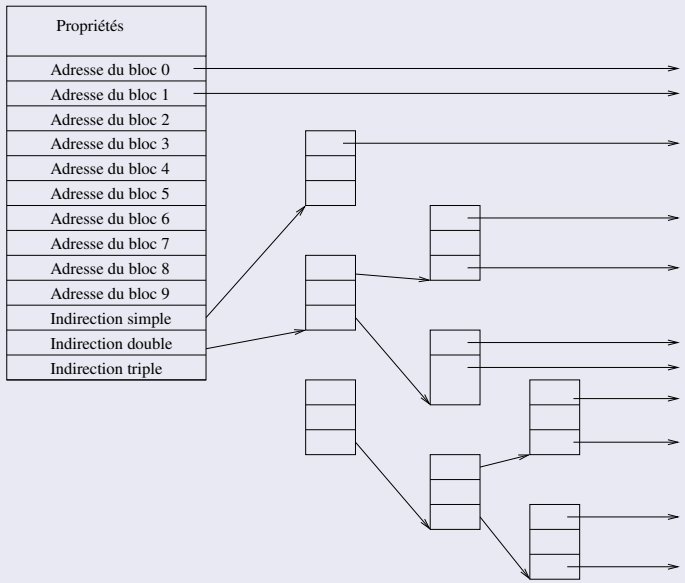


Taille max = $10 + 1024 + 1024^2$ blocs = 1 049 610 $\times$ 4 Ko/bloc = 4 198 440 Ko $\approx$ 4,1 Go

Politique d'allocation des blocs sur disque - type UNIX

Inodes et gros fichiers

- Pour pouvoir gérer des gros fichiers, les inodes intègrent des indirections simples, doubles (voire triples)



- Taille max = $10 + P + P^2 + P^3$ blocs = 1 074 791 434 adresses $\times$ 4Ko/bloc = 4 299 165 736 Ko $\approx$ 4 To

  Utilisation de 2 indirections double : Taille = $10 + P + 2P^2$ = 2 098 186 $\times$ 4Ko/bloc = 8 392 744 Ko $\approx$ 8,2 Go

Politique d'allocation des blocs sur disque - type UNIX

Inodes et gros fichiers

- Place utilisée sur le disque pour stocker les blocs d'indirection d'une inode d'un fichier de taille maximale :
  - Indirection simple : 1 bloc
  - Indirection double : P blocs
  - Indirection triple : P^2 blocs

  soit $1 + (1+P) + (1+P^2) = 1\,049\,603$ blocs $= 4\,198\,412$ Ko ≈ 4 Go
- Taille d'une adresse : avec 32 bits, la taille maximale d'un disque est de 2^{32} *blocs* de 4 Ko $= 2^{32} \times 2^2 = 2^4 \times 2^{30} = 16$ Go

Homework's

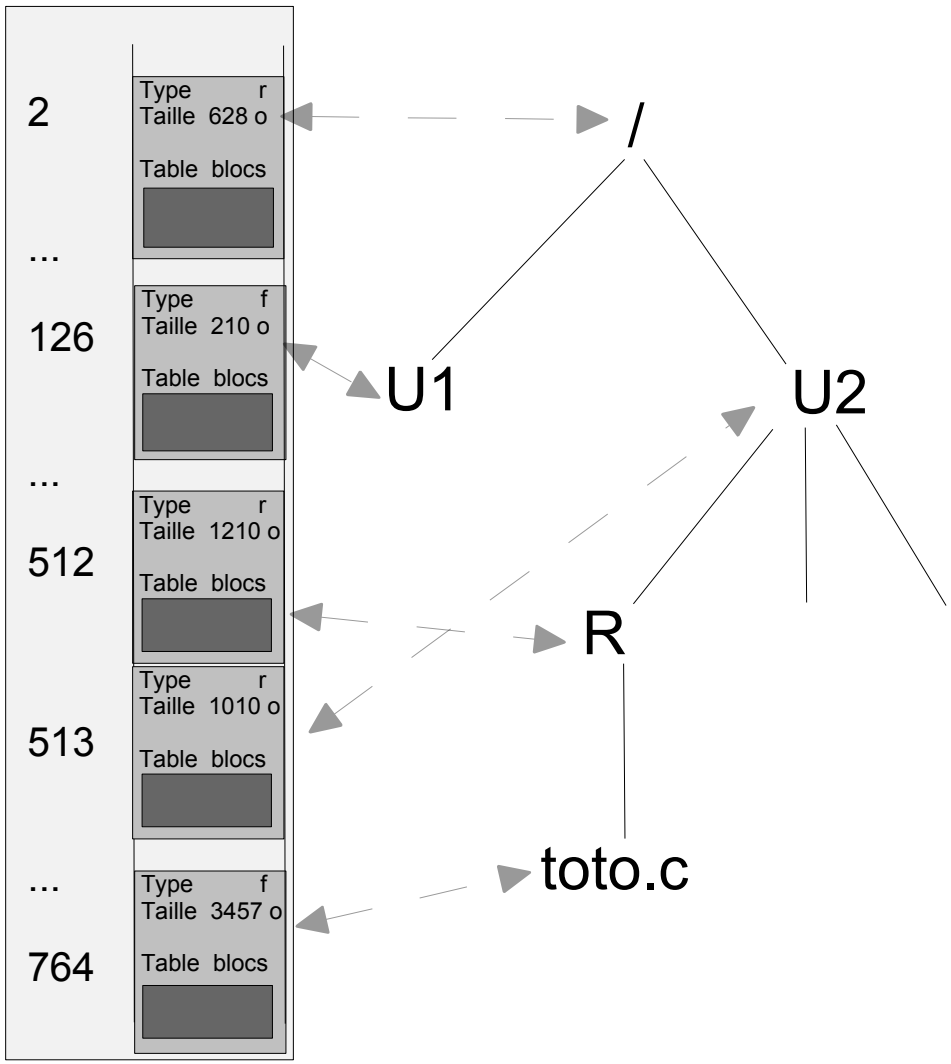
Refaire le calcul de la taille maximum d'un fichier avec un taille de block de 4Ko et des adresses sur 64 bits. Idem mais avec des blocks de 8Ko. Commentez.

Réponses : 513Mo - 8To

Gestion des fichiers/répertoires - type UNIX

Table des inodes

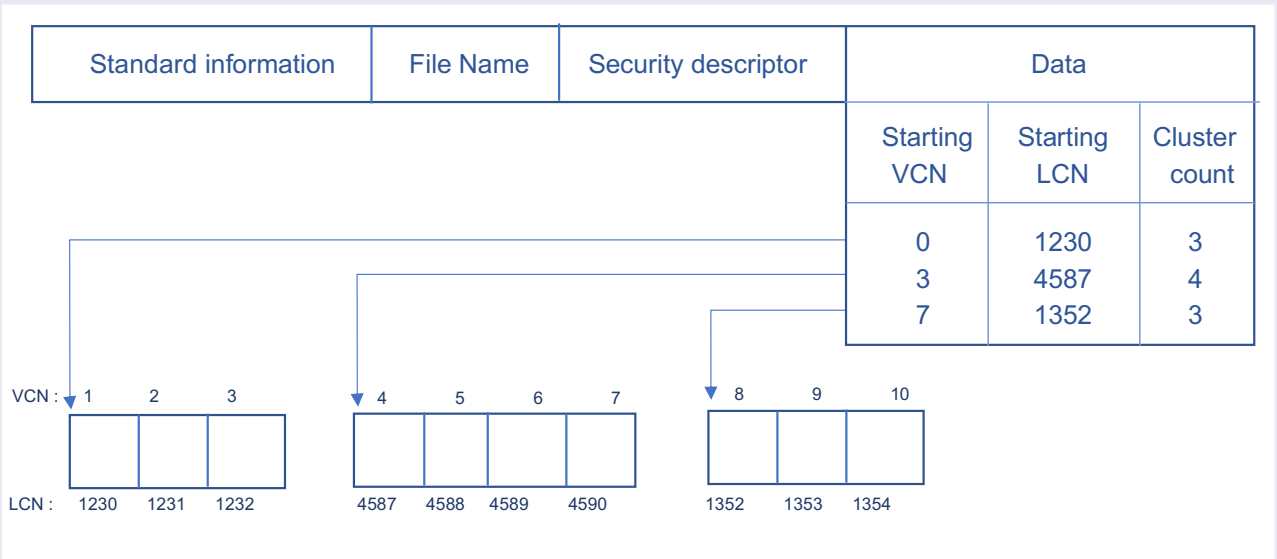
- Contient toutes les informations nécessaires pour gérer les fichiers et répertoire...
- La racine / a toujours l'inode 2



Systèmes de gestion de fichiers : NTFS

Descripteur de fichier

- La taille de ces descripteurs est fixée à la création du volume et est comprise entre 1 Ko et 4 Ko
- Contient une champ $Data qui donne une liste de zones continues .

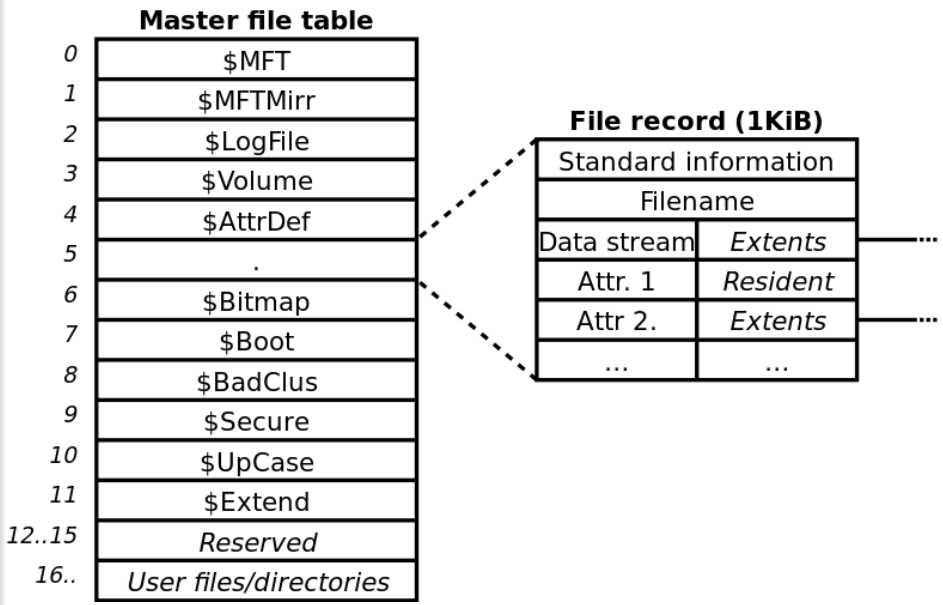


- Si le fichier est petit, il est mis directement dans $Data
- S'il est trop gros : une nouvelle entrée dans la table est affectée (indirection)

Systèmes de gestion de fichiers : NTFS

Table des descripteurs

- Les 16 premiers descripteurs sont utilisés pour décrire le volume lui-même
- Le descripteur #5 correspond à la racine
- les fichiers commence au descrpteurs #16
- La MFT ne peut que grandir (jusqu'à une limite fixée)



UNIX ≠ NTFS

- UNIX : Les inodes sont en partie "réparties" sur le disque
- NTFS : Tout est dans la MFT

Plan

Organisation d'un système de fichiers UNIX

Partition

- Une partition est une partie de disque comportant un système de fichiers ($\approx$ disque virtuel)

Bloc de boot	Superbloc	Gestion des blocs libres	I–nodes	Répertoire racine	Répertoires et fichiers

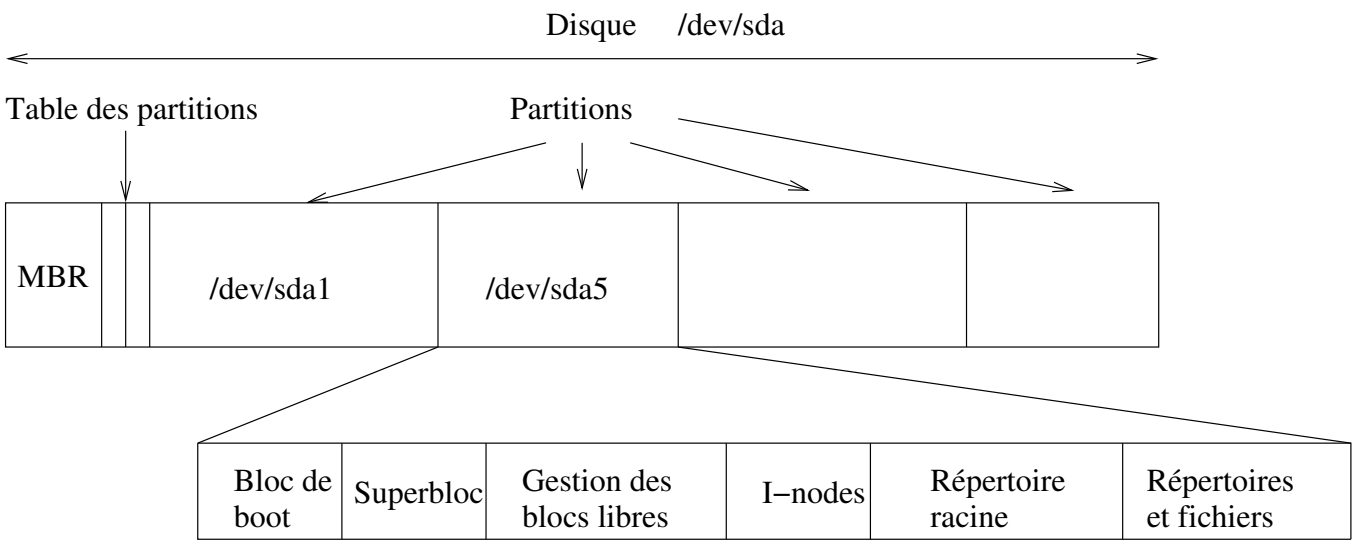
Structure d'une partition UNIX

- Premier bloc (boot block) permet le chargement du système d'exploitation éventuellement présent sur la partition.
- Chaque partition a sa propre table des inodes
- Superbloc : paramètres relatifs au système de fichiers.

Gestion des partitions UNIX

Structure type d'un disque

- MBR : Master Boot Record.
  - Contient le programme de démarrage du système d'exploitation.
  - La fin du MBR contient la table des partitions : informations sur les adresses de début et de fin des partitions du disque.
  - Le MBR détermine la partition active, et y lit le premier bloc (boot block) permettant le chargement du SE présent sur la partition.
- Superbloc : paramètres relatifs au système de fichiers.



Organisation d'un système de fichiers

Montage d'une partition

- Pour être accessible par un système UNIX, une partition doit être montée, c'est-à-dire rattachée à l'arborescence des fichiers
- Commande : `mount -t type device dir`
- Exemple : mount -t ext4 /dev/sda1 /
  - Associe la racine du système à l'inode 2 de la partition /dev/sda1
  - `ls -ia /` donne :
    2 . 523 etc 621 user
    2 .. 365 R

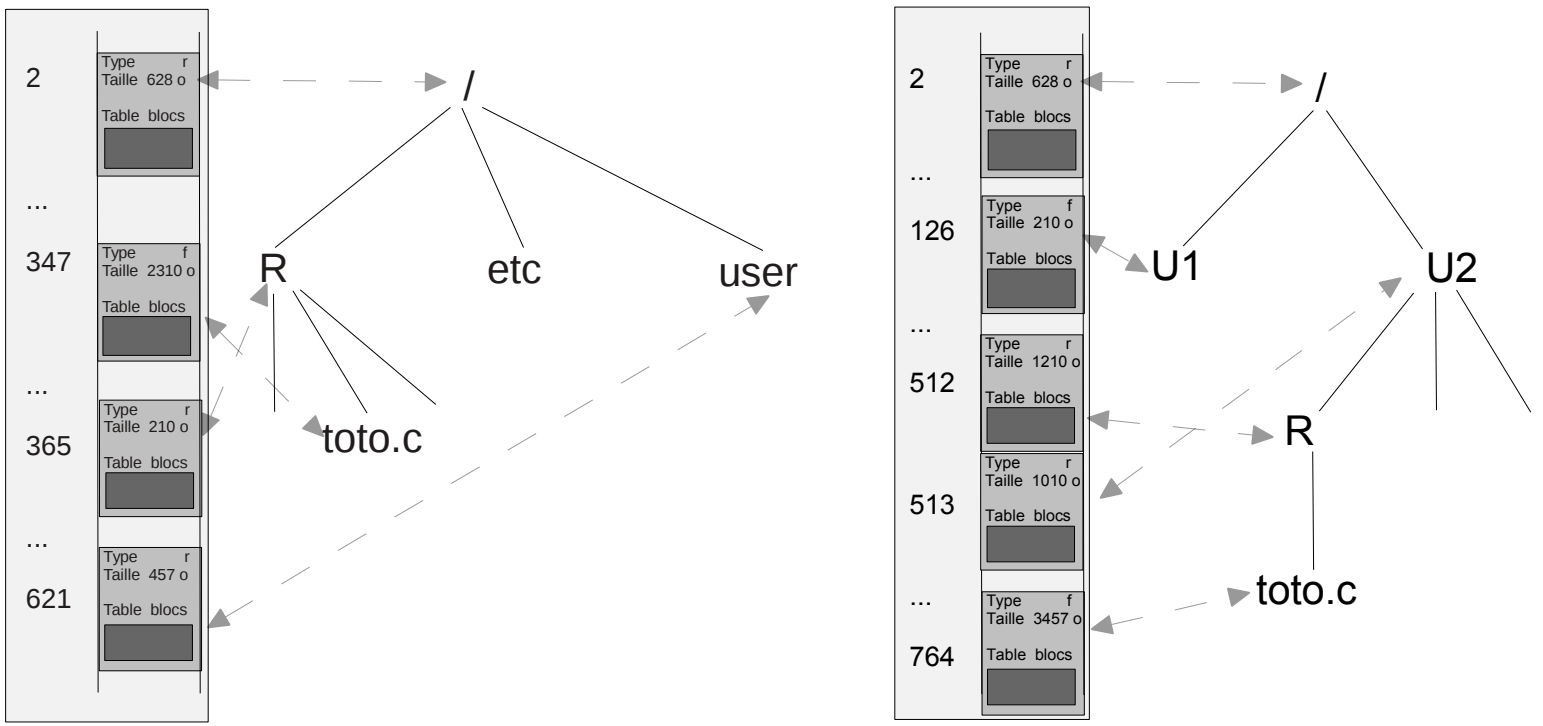
Montage de plusieurs partitions

- Unix permet de monter plusieurs partitions à condition qu'elles forment une arborescence

Organisation d'un système de fichiers

Montage de plusieurs partitions

- Exemple : soient les deux partitions `/dev/sda1` et `/dev/sda5`

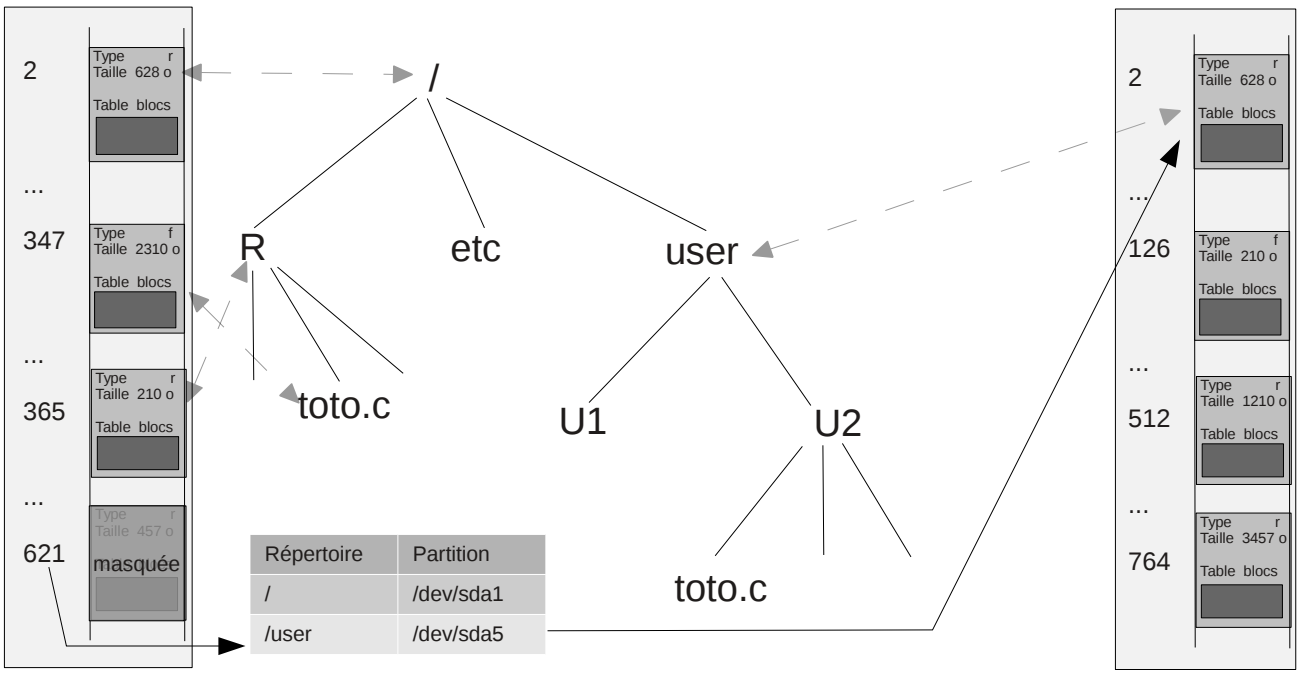


Organisation d'un système de fichiers

Montage de plusieurs partitions

- `mount -t ext4 /dev/sda1 /`
  `mount -t ext4 /dev/sda5 /user`

⇒ Masque l'inode 621 de `/dev/sda1` par l'inode 2 de `/dev/sda5`

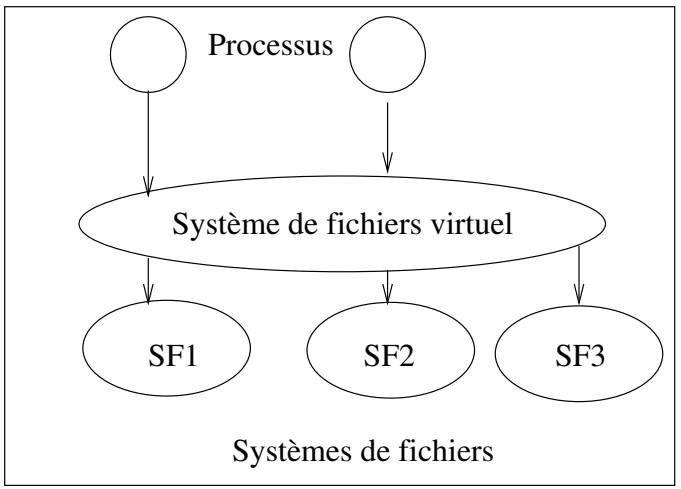


- `ls -ia /` donne : 2 . 523 etc **2 user**
  2 .. 365 R

Montage de plusieurs partitions

Systèmes de fichiers virtuels

- Permet de gérer de manière transparente plusieurs systèmes de fichiers différents.
- Un interface **VFS** (Virtual File System) fait le lien entre la couche haute et la couche basse : les systèmes de fichiers concrets.
- Principe utilisé dans les systèmes UNIX
- Permet aussi de monter des disques distants : NFS



Entrées/Sorties

Liens avec les périphériques

- Le répertoire `/dev` est utilisé pour associer des périphériques à des fichiers "virtuel" (`/dev/tty`, `/dev/media/usb1`, ...)
- Chacun de ces fichiers se réfère à un périphérique, qui peut -ou pas- exister.
- Les applications utilisateur peuvent utiliser ces fichiers pour interagir avec le périphérique.
- Par exemple, le serveur graphique X va "écouter" `/dev/input/mice` associé à la souris et faire bouger le pointeur à l'écran.
- Initialement, `/dev` contenait tous les fichiers correspondant à tous les périphériques possibles et imaginables que l'on pouvait trouver dans une configuration matérielle.

Plan

Cohérence d'un système de fichiers

Gestions des blocs

- Utilitaires permettant de vérifier la cohérence du système de fichiers : UNIX `fsck`, Windows `scandisk`.
- Construction de deux tables :
  - une table mémorisant le nombre de fois qu'un bloc est référencé dans un fichier
  - une table mémorisant le nombre de fois qu'un bloc figure dans la table des blocs libres

Gestions des blocs : état cohérent

Blocs utilisés :	1	1	0	1	1	0	0	1	1	0
Blocs libres :	0	0	1	0	0	1	1	0	0	1

Cohérence d'un système de fichiers

Gestions des blocs : Etats incohérents

Blocs utilisés :	1	1	**0**	1	1	0	0	1	1	0
Blocs libres :	0	0	**0**	0	0	1	**2**	0	0	1

- Bloc libre non référencé, ou duppliqué
- Solution : reconstruire la liste des blocs libres.

Blocs utilisés :	1	1	0	**2**	1	0	0	1	1	0
Blocs libres :	0	0	1	0	0	1	1	0	0	1

- Etat incohérent : bloc utilisé dupliqué (le plus grave)
- Solution : faire une copie du bloc et l'associer à l'un des deux fichiers l'utilisant.

Cohérence d'un système de fichiers

Cohérence répertoires - fichiers

- Dans chaque inode est indiqué le nombre N_i de répertoires auxquels le fichier/répertoire appartient
  Attention : un fichier/répertoire peut apparaître dans plusieurs répertoires grâce aux liens matériels (hard link)
- Test de cohérence par parcours des répertoires :
  $\rightsquigarrow$ vérifier que le nombre N_r de répertoires dans lesquels le fichier/répertoire apparaît est cohérent avec l'information N_i dans l'inode du fichier/répertoire
  - Si $N_r = N_i$: Ok
  - Si $N_r \neq N_i$: mettre N_i à jour ($N_i = N_r$)
  - Cas particulier où $N_r = 0 \rightarrow$ le fichier n'apparait dans aucun répertoire :
    - Si $N_i = 0$: le fichier est supprimé
    - Si $N_i > 0$: le fichier est mis dans le répertoire `lost+found`

Mémoire cache

La mémoire cache

- Lecture d'un mot en mémoire principale : au plus 10 ns
- Lecture d'un mot sur disque : quelques millisecondes pour la recherche de la piste et le positionnement sur le bon secteur.
- Si un seul mot est nécessaire, la lecture sur disque est un million de fois plus lente.
- Possibilité d'optimiser les temps d'accès.
- Technique courante : utilisation d'une mémoire cache. Ensemble de blocs conservés en mémoire pour améliorer les performances.

Systèmes journalisés et distants

Systèmes de fichiers journalisés

- Objectif : robustesse aux pannes, assurer la cohérence du système de fichiers en toute circonstance.
- Principe : conserver dans un journal la trace des actions effectuées par le système avant qu'elles soient effectuées.
- En cas de plantage, le système peut retrouver la trace des actions à effectuer et terminer le travail.
- Exemples de tels systèmes de fichiers : NTFS, ext3, ext4