Programmation système

M. Davy MOUSSAVOU- Consultant
Linux, Certifié Linux International

Système de fichier

Objectifs du cours

Après cette session les apprenants:

- Connaissent l'organisation d'un système de fichier;

- Connaissent l'organisation

Agenda

- Séquence 1: Organisation des disques
- Séquence 2: Organisation d'un système de fichier
- Séquence 3: Appels systèmes de gestion de système de fichier

Séquence 1 : Organisation des disques

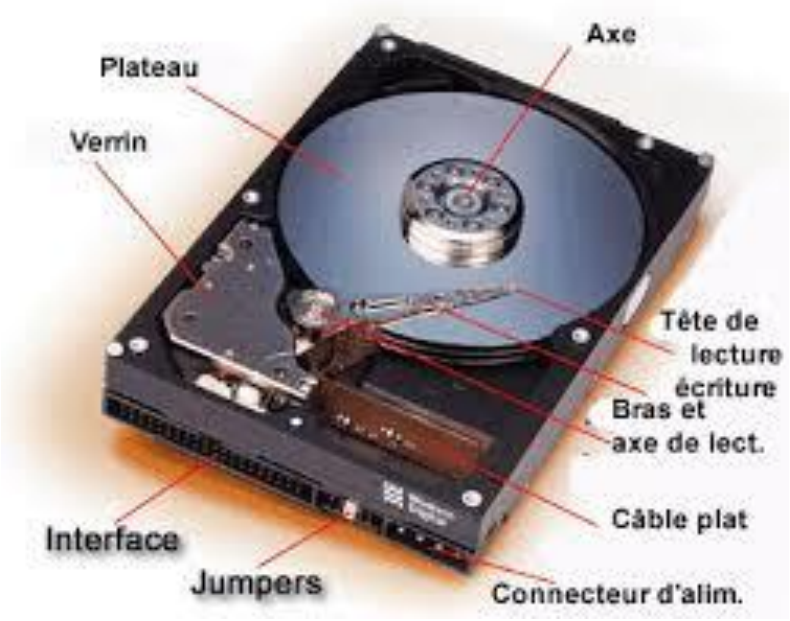
Séquence 1 : Organisation des disques

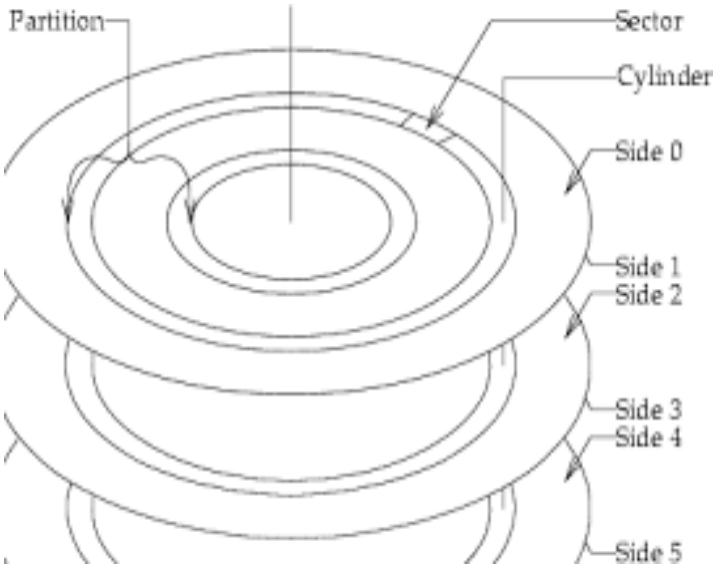
❑ Objectifs

- Comprendre la structure interne d'un disque dur;
- Organisation des partitions;

Séquence 1 : Organisation des disques

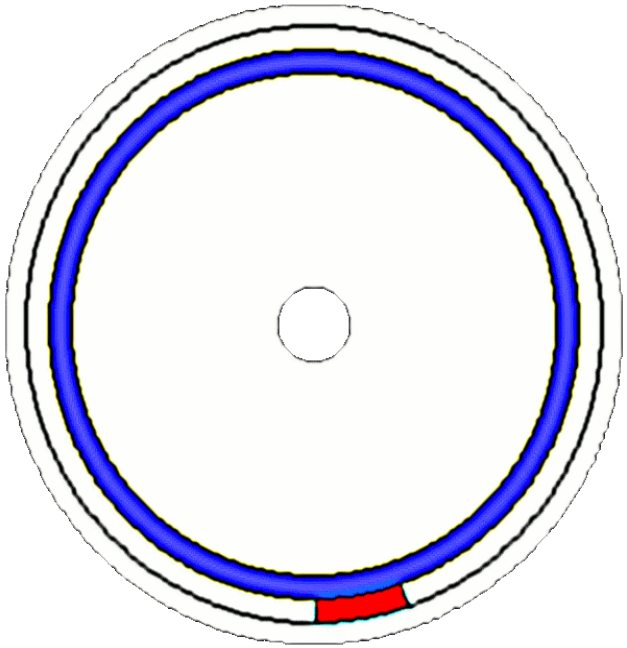
Structure d'un disque dur

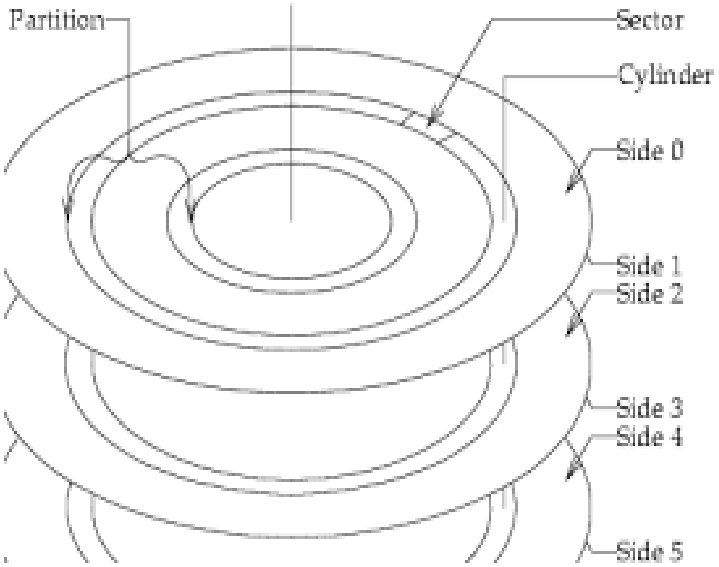




Séquence 1 : Organisation des disques

Structure d'un disque dur

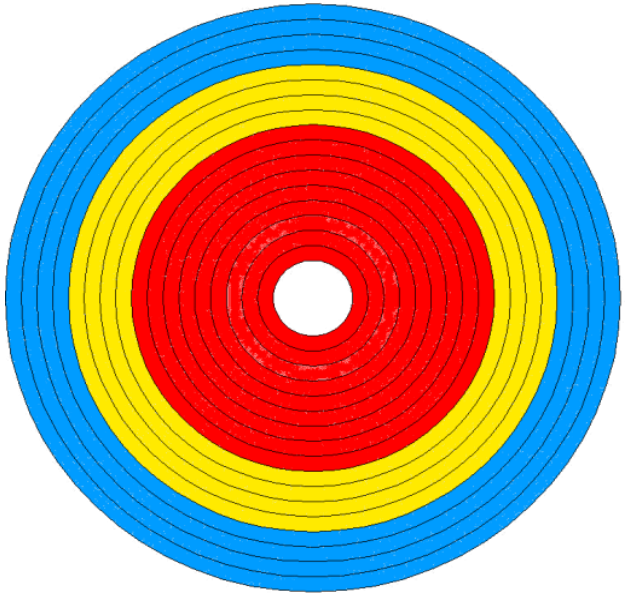


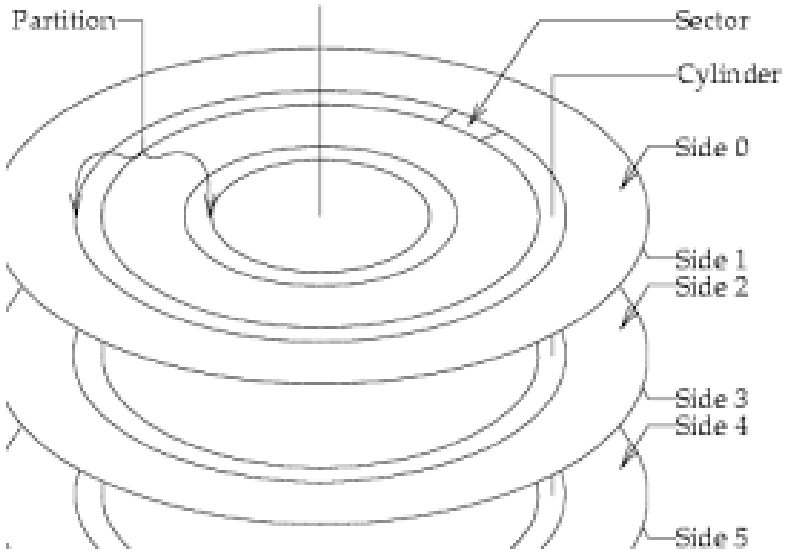


Un système de fichier?

Séquence 1 : Organisation des disques

Organisation des partitions





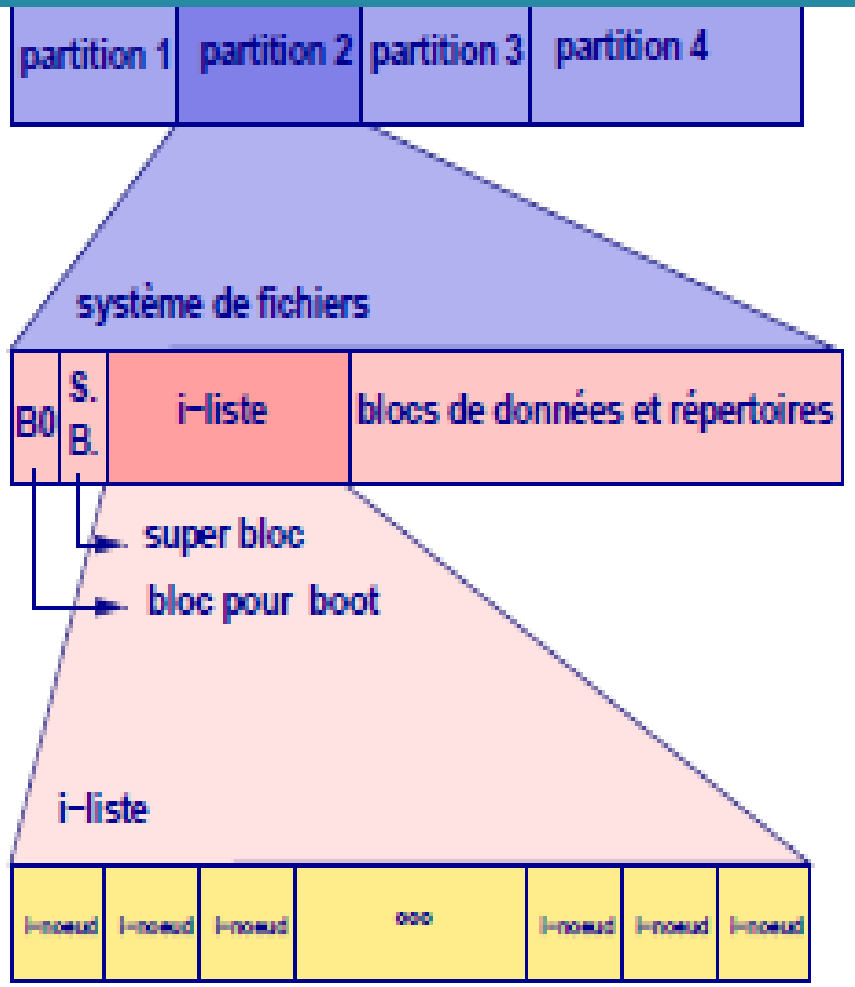
- Que signifie partitionner ?
- Que signifie formater ?

Séquence 1 : Organisation des disques

Organisation des partitions

- Un disque dur est composé de partitions, chacune considérée comme un disque d'un point de vue logique.

- Les partitions disposent d'un ensemble de secteurs particuliers, notamment:
  - Le super bloc
  - Le bloc d'amorçage



Séquence 2: Organisation du système de fichier

Séquence 2 : Organisation d'un système de fichier

❑ objectifs

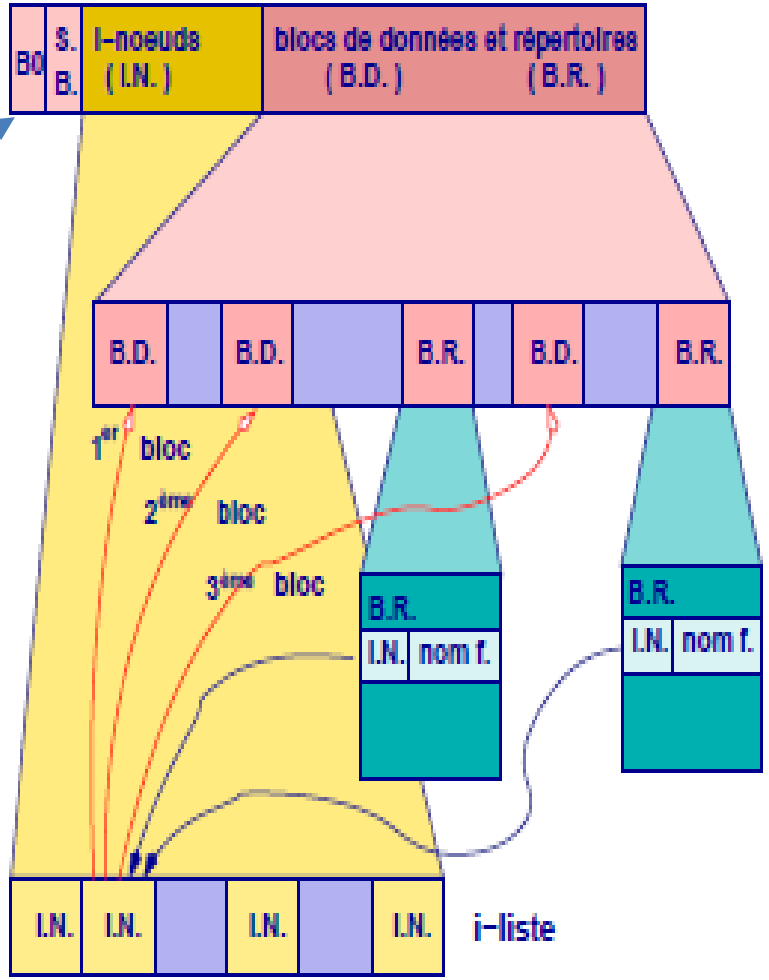
- ➢ Comprendre la structure d'un système de fichier
- ➢ Identifier le type de système de fichier

Séquence 2 : Organisation d'un système de fichier

Bloc d'amorçage

Bloc d'amorçage

C'est le premier bloc d'un système de fichier.
Il peut contenir un programme en charge d'initialiser le système. Ce programme se nomme chargeur de démarrage(grub ou lilo).
Ce bloc est généralement situé sur le premier système de fichiers.
Si plusieurs partitions existent, leur bloc d'amorçage est vide

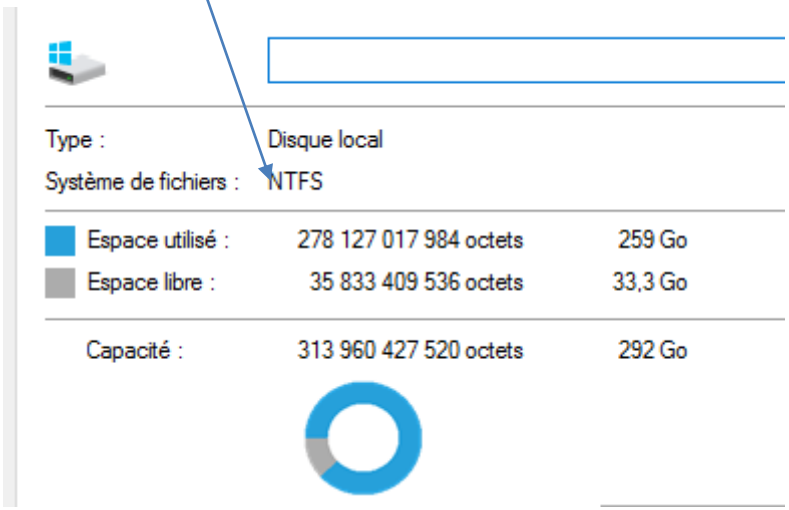


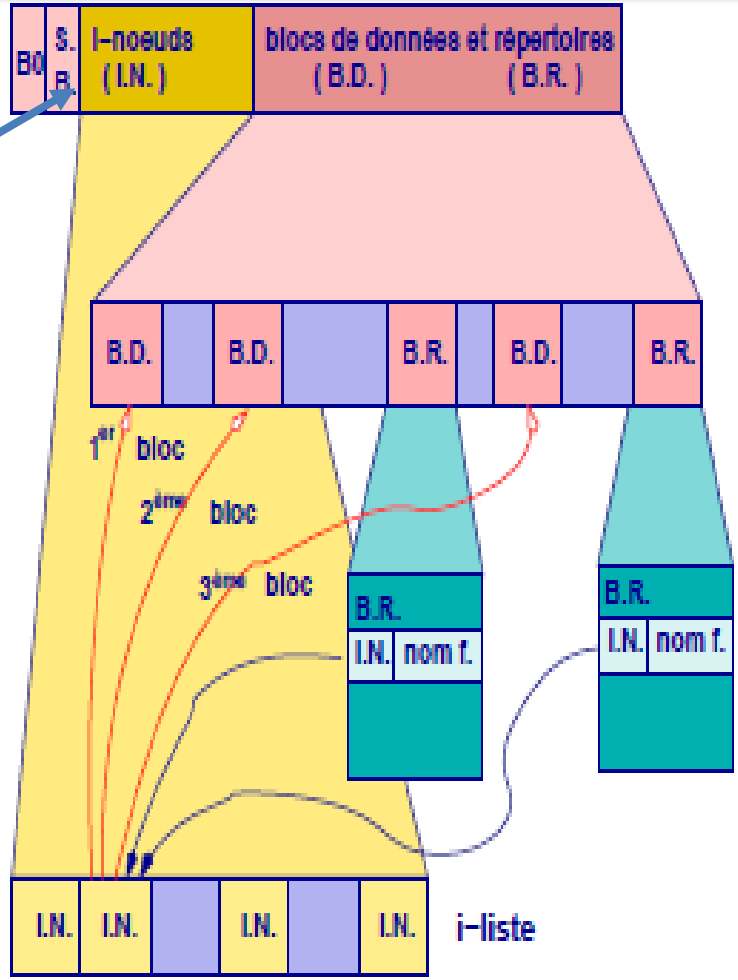
Séquence 2 : Organisation d'un système de fichier

Super bloc

son type;

Super bloc

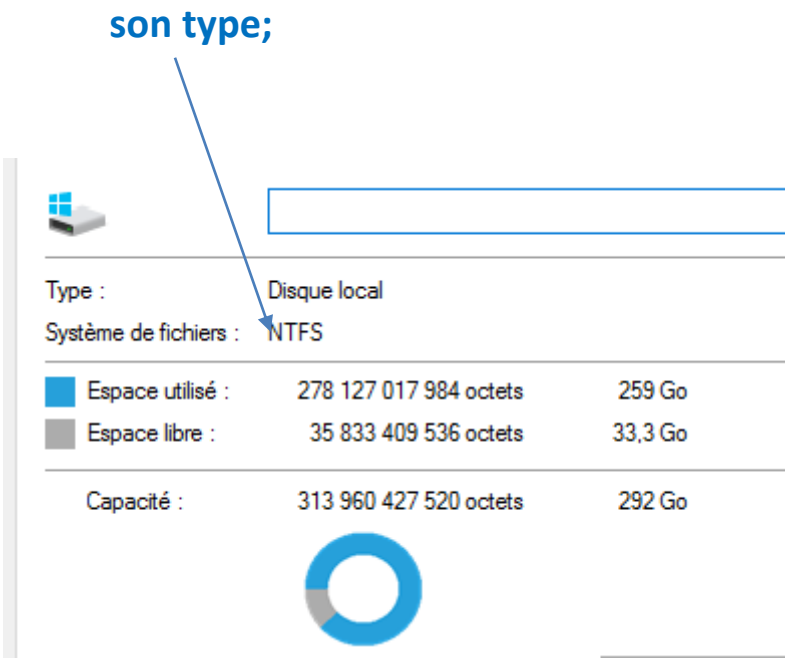




Séquence 2 : Organisation d'un système de fichier

Super bloc

Super bloc



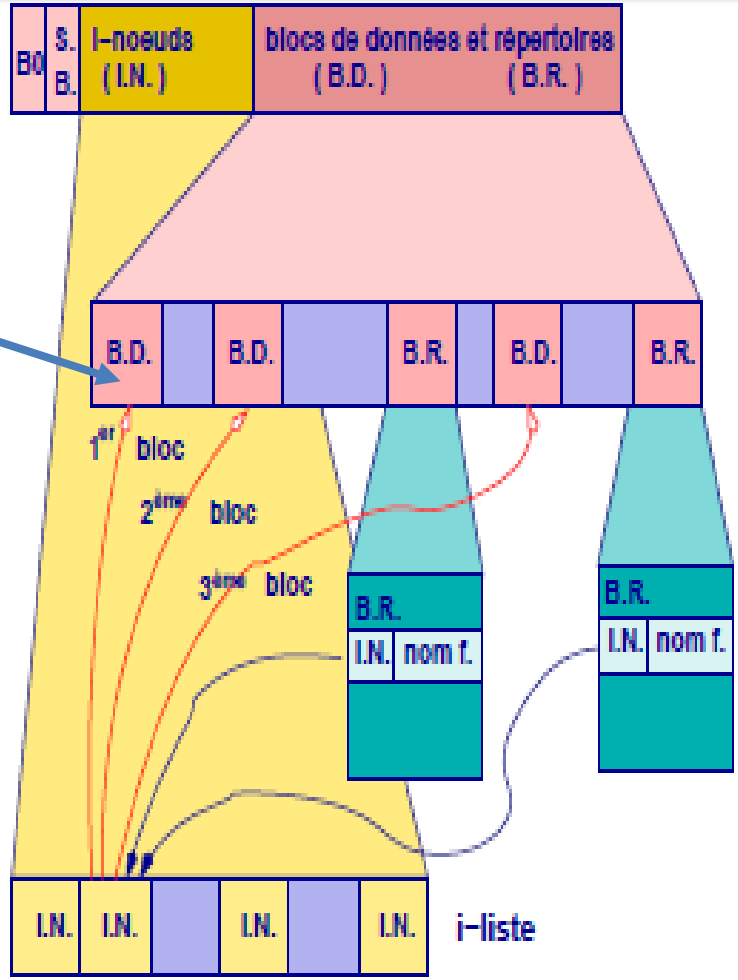
Il contient toute les information concernant le système de fichier, notamment :

- **son type;**
- **sa taille en blocs, et inodes**
- **Nombre de bloc libres**
- **Nombre de blocs réservés aux inodes**
- **Taille d'un bloc de données**
- **Heure de la dernière modification effectuée**
- **Heure de la dernière vérification du système de fichiers**

Séquence 2 : Organisation d'un système de fichier

Les blocs de données et de répertoires

Les blocs

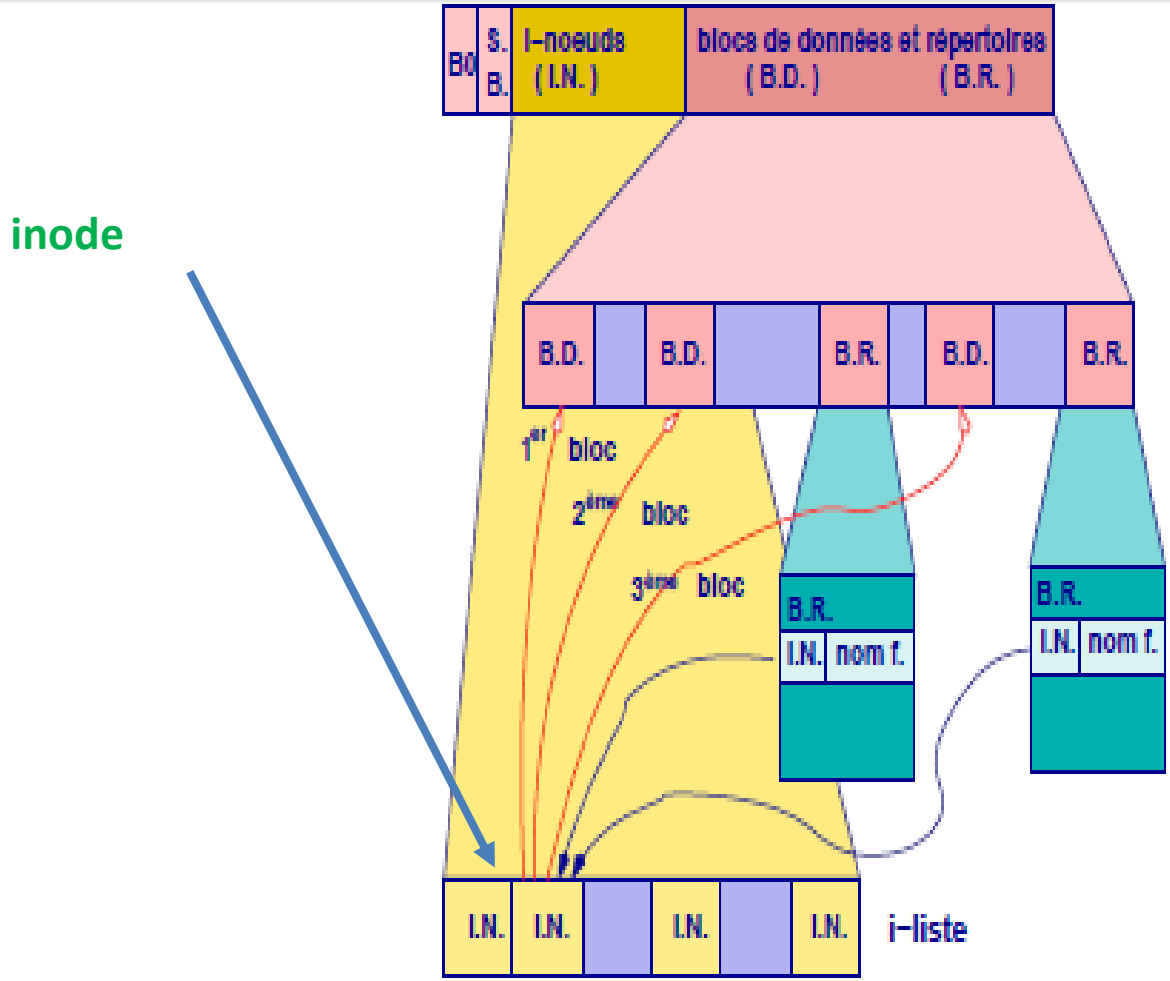


Un bloc sur un disque a un taille prédéfinie. Il se définit généralement au moment du formatage du système de fichier. Ainsi on distingue deux types de blocs:

- Bloc de données(BD);
- Bloc de répertoires(BR).

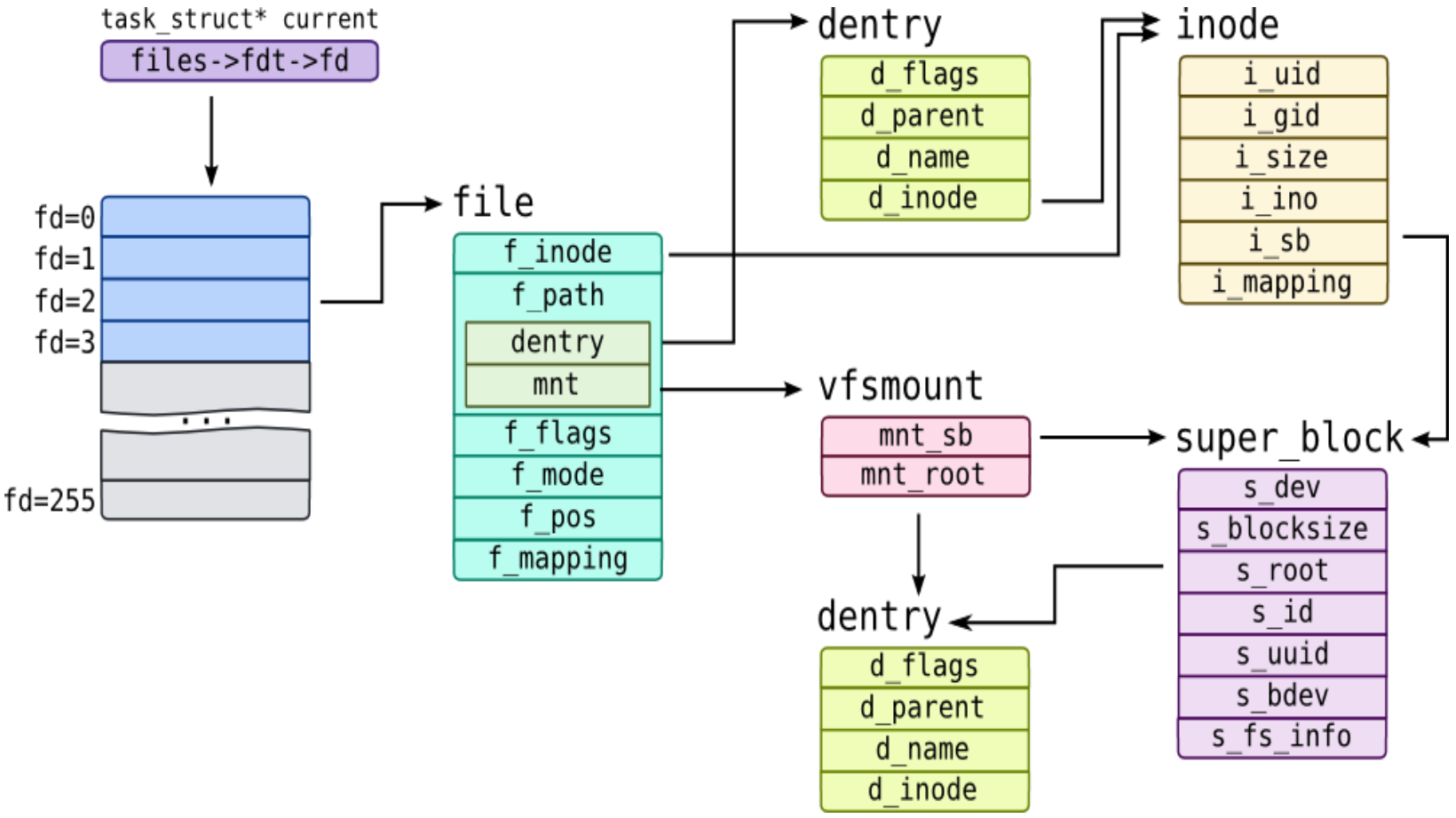
Séquence 2 : Organisation d'un système de fichier

La table d'inode



Séquence 2 : Organisation d'un système de fichier

La table d'inode



Séquence 2 : Organisation d'un système de fichier

La table d'inode

Les informations contenues dans la table d'inode:

- Type du fichier et les droits d'accès des différents utilisateurs;
- L'indentification du propriétaire du fichier;
- L'indentification du groupe propriétaire du fichier;
- La taille du fichier;
- Le nombre de liens physiques sur le fichiers;
- La date de la dernière modification(écriture) du fichier;
- La date de dernière consultation(lecture) du fichier;
- La date de la modification d'attributs
- L'adresse blocs utilisés sur le disque pour ce fichier
- L'identification de la ressource associé(pour les fichier spéciaux)

Séquence 2 : Organisation d'un système de fichier

Type de système de fichier

Nom	description
ffs	Le file system local de NetBSD (équivalent de ufs sur alpha ou hfs sur apple).
iso9660	Le système de fichiers des CDROMs à la norme ISO 9660.
Ext2,ext3,ext4	Le système de fichiers historique de linux (maintenant ext3 ou reiserfs)
Fat	Le système de fichiers du bon vieux DOS dans les variantes 12, 16 ou 32 bits
mfs	Système de fichiers optimisé pour adresser la mémoire RAM.
nfs	Sait accéder à des données sur le réseau par le protocole nfs.
kernfs	Pseudo file system qui permet de visualiser des paramètres kernel.
procfs	Pseudo file system représentant l'image mémoire des processus.

Séquence 3: Appels système de gestion de système de fichier

Séquence 3 : Appels systèmes de gestion de fichier

❑ objectifs

- ➢ Afficher les informations sur la table d'inode
- ➢ Effectuer les opérations de base sur les fichiers et répertoires

Séquence 3 : Appels systèmes de gestion de fichier

Information sur un fichier

❑ stat(2)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

Chemin du fichier

```
struct stat {
    dev_t     st_dev;     /* Périphérique                */
    ino_t     st_ino;     /* Numéro i-nœud               */
    mode_t    st_mode;    /* Protection                  */
    nlink_t   st_nlink;   /* Nb liens matériels          */
    uid_t     st_uid;     /* UID propriétaire            */
    gid_t     st_gid;     /* GID propriétaire            */
    dev_t     st_rdev;    /* Type périphérique           */
    off_t     st_size;    /* Taille totale en octets     */
    blksize_t st_blksize; /* Taille de bloc pour E/S     */
    blkcnt_t  st_blocks;  /* Nombre de blocs de 512B alloués */
    time_t    st_atime;   /* Heure dernier accès         */
    time_t    st_mtime;   /* Heure dernière modification */
    time_t    st_ctime;   /* Heure dernier changement état */
};
```

Exemple 1:

```
int main(int argc, char *argv[])
{
struct s;
stat(argv[1], &s);
return 0;
}
```

```
$cc my_stat–o my_stat
$./my_stat fic1
```

Séquence 2 : Appels systèmes de gestion de fichier

❑ stat(2)

```
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>

int stat(const char *path, struct stat *buf);
int fstat(int fd, struct stat *buf);
int lstat(const char *path, struct stat *buf);
```

Exemple :

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<unistd.h>

int main(int argc, char *argv[])
{
struct stat s;

//appel de la fonction

stat(argv[1], &s);

//affichage

prinft(''inode:%d\n'', s.st_ino);
return 0;
}
$cc my_stat–o my_stat
$./my_stat fic1
```

Séquence 3 : Opérations d'entrée sortie sur les fichiers

Ouverture d'un répertoire

- **opendir(3)**

```
SYNOPSIS
       #include <sys/types.h>
       #include <dirent.h>

       DIR *opendir(const char *nom);
       DIR *fdopendir(int fd);
```

Exemple :

```
#include<sys/types.h>
#include<dirent.h>

int main(int argc, char *argv[])
{
DIR *rep;

//appel de la fonction

rep=opendir(argv[1]);

return 0;
}
```

Séquence 3 : Opérations d'entrée sortie sur les fichiers

Lecture du contenu d'un répertoire

- **readdir(3)**

```
SYNOPSIS
       #include <dirent.h>

       struct dirent *readdir(DIR *dirp);
```

Exemple :

```
#include<sys/types.h>
#include<dirent.h>

int main(int argc, char *argv[])
{
DIR *rep;

struct dirent *contenu;

//appel de la fonction open

rep=opendir(argv[1]);

//appel de la fonction readdir

contenu=readdir(rep);

return 0;
}
```

Atelier