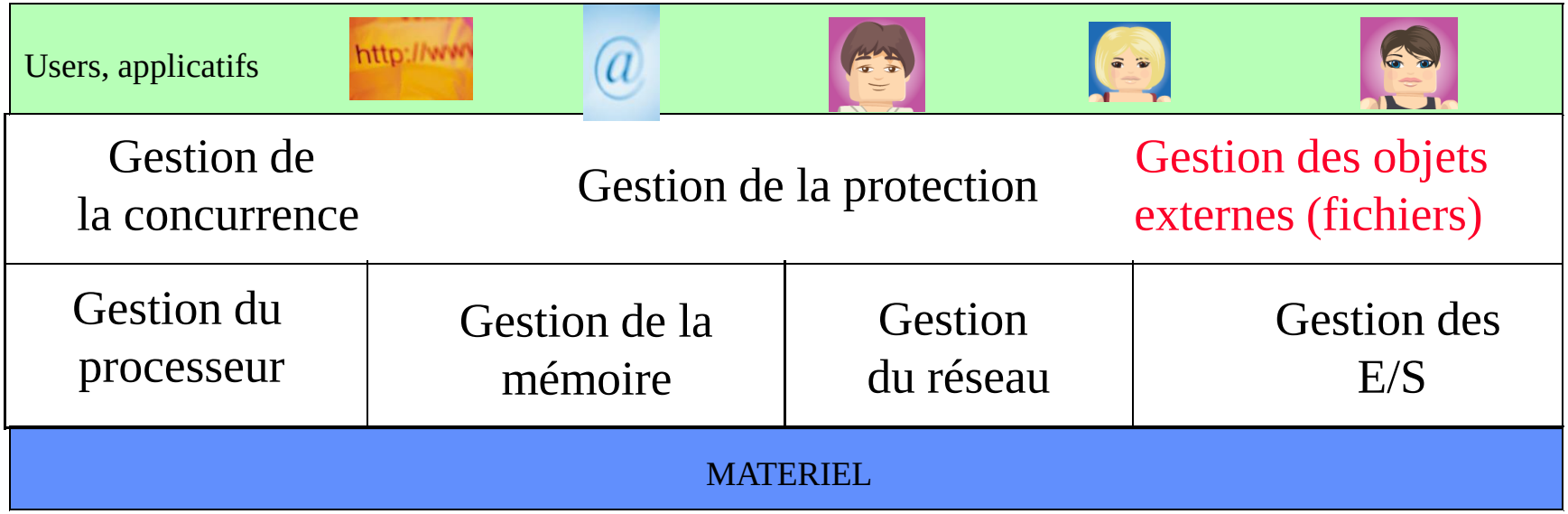
Système de gestion de fichiers

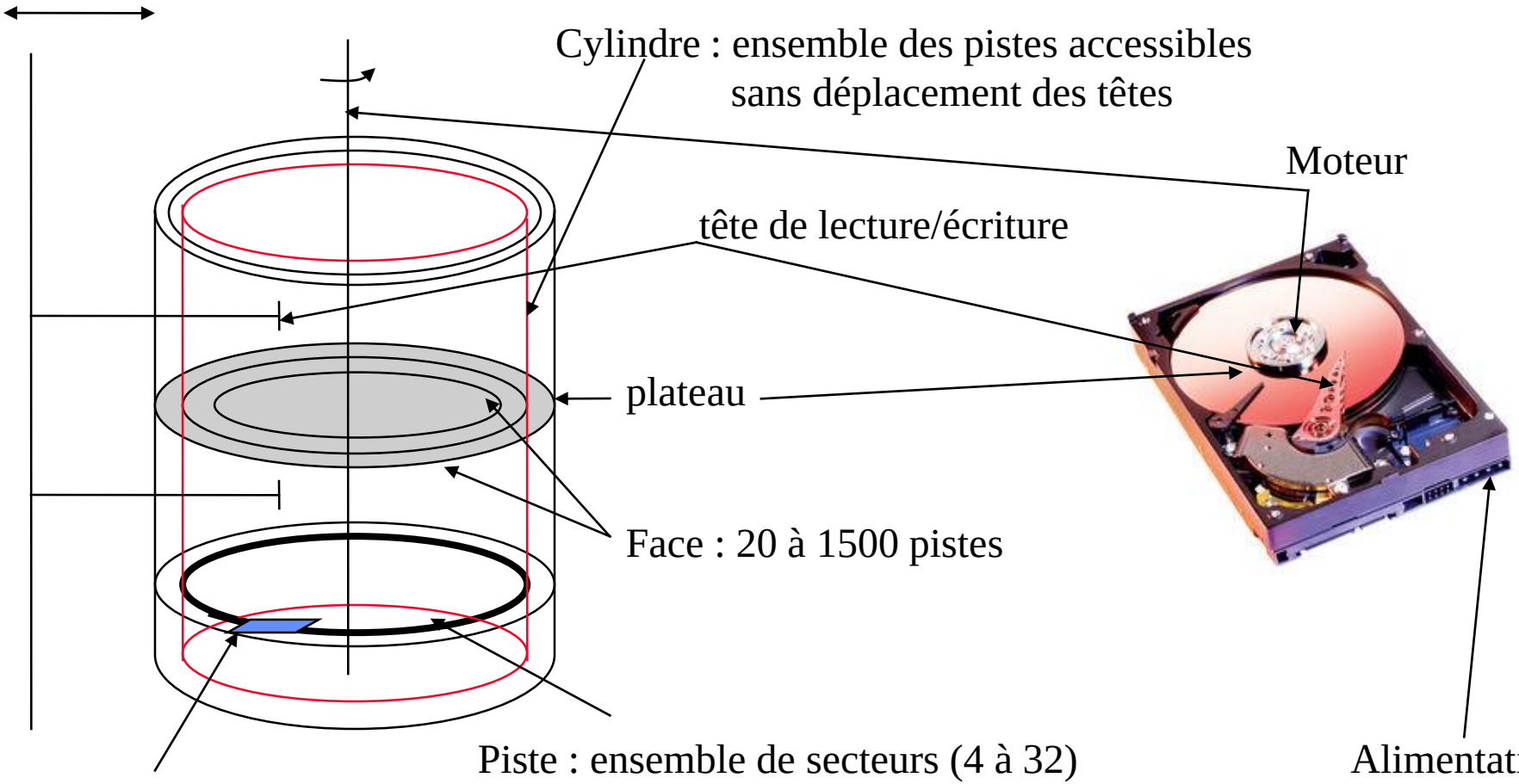


Gestion des objets externes : la mémoire centrale est une mémoire volatile :

- il faut stocker les données devant être conservées au delà de l'arrêt de la machine sur un support de masse permanent
  - l'unité de conservation sur le support de masse est le *fichier.*
  - *Le système d'exploitation gère les fichiers via le Système de gestion de fichiers (SGF)*
  - *Deux vues : une vue logique (utilisateur) ; une vue physique (système)*

Structure du disque dur

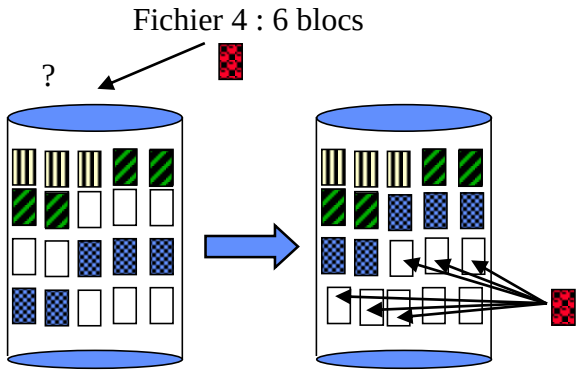
↳ Adresse d'un secteur : n°face, n°cylindre,n°secteur



Secteur : plus petite unité d'information accessible
32 à 4096 octets
512

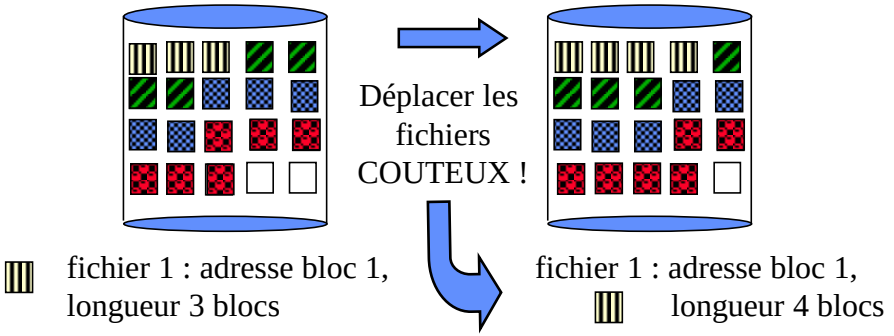
Allocation contiguë

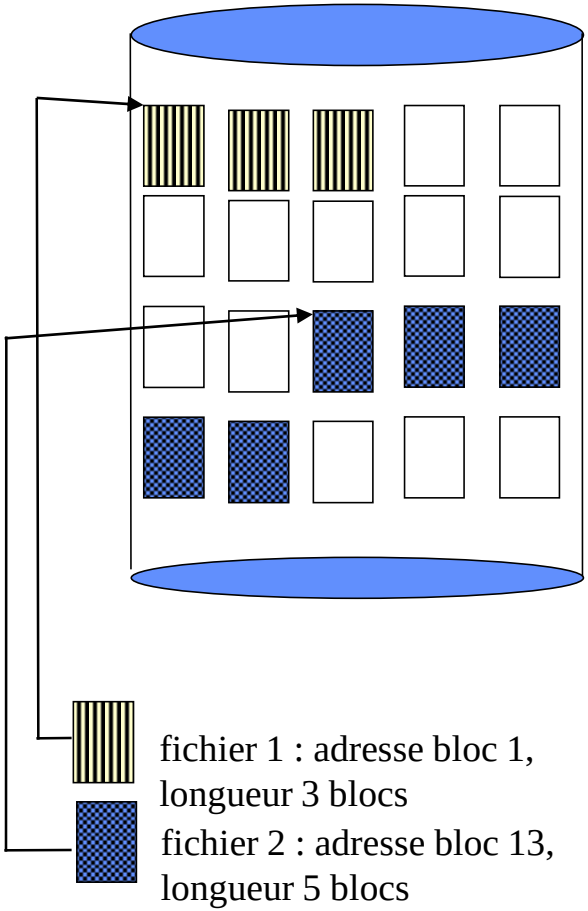
- Un fichier occupe un ensemble de blocs contigus sur le disque
- Allocation : trouver un trou suffisant (First Fit, Best Fit)
- Difficultés
  - création d'un nouveau fichier



Fragmentation , Compactage

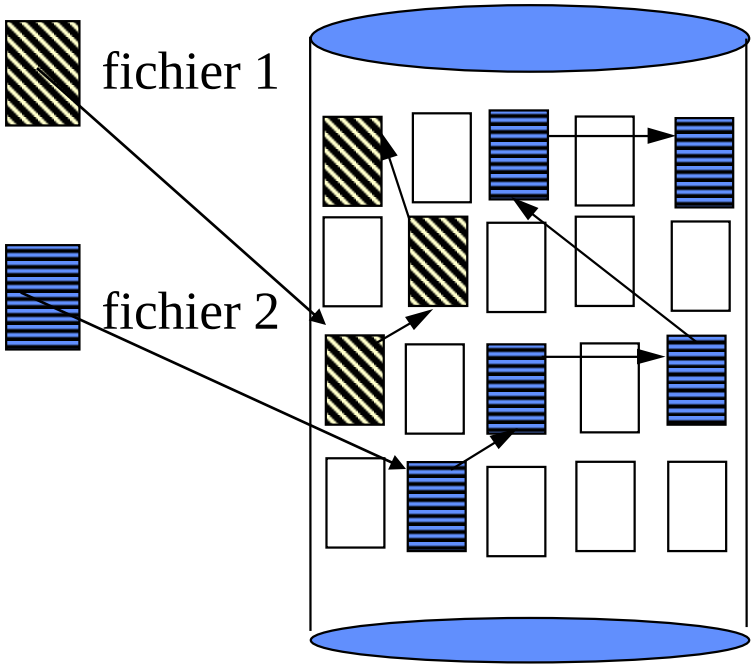
  - extension du fichier





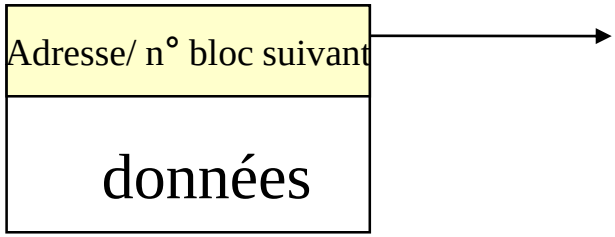
Allocation par bloc chainée

- Un fichier est constitué comme une liste chainée de blocs physiques, qui peuvent être dispersés n'importe où.



• Extension simple du fichier : allouer un nouveau bloc et le chainer au dernier
• Pas de fragmentation

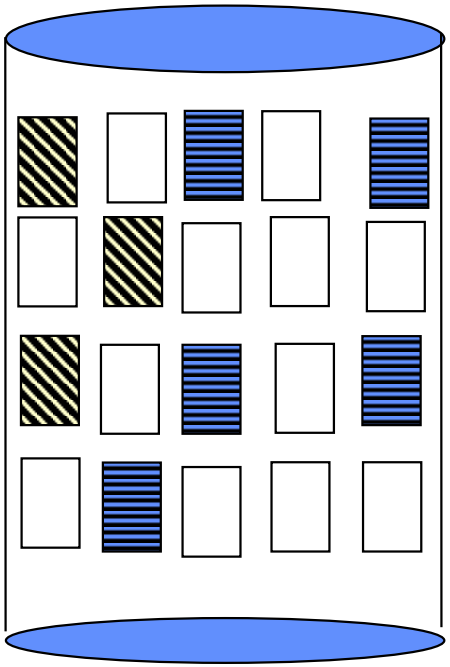
• Difficultés :
- le chaînage du bloc suivant occupe de la place dans un bloc

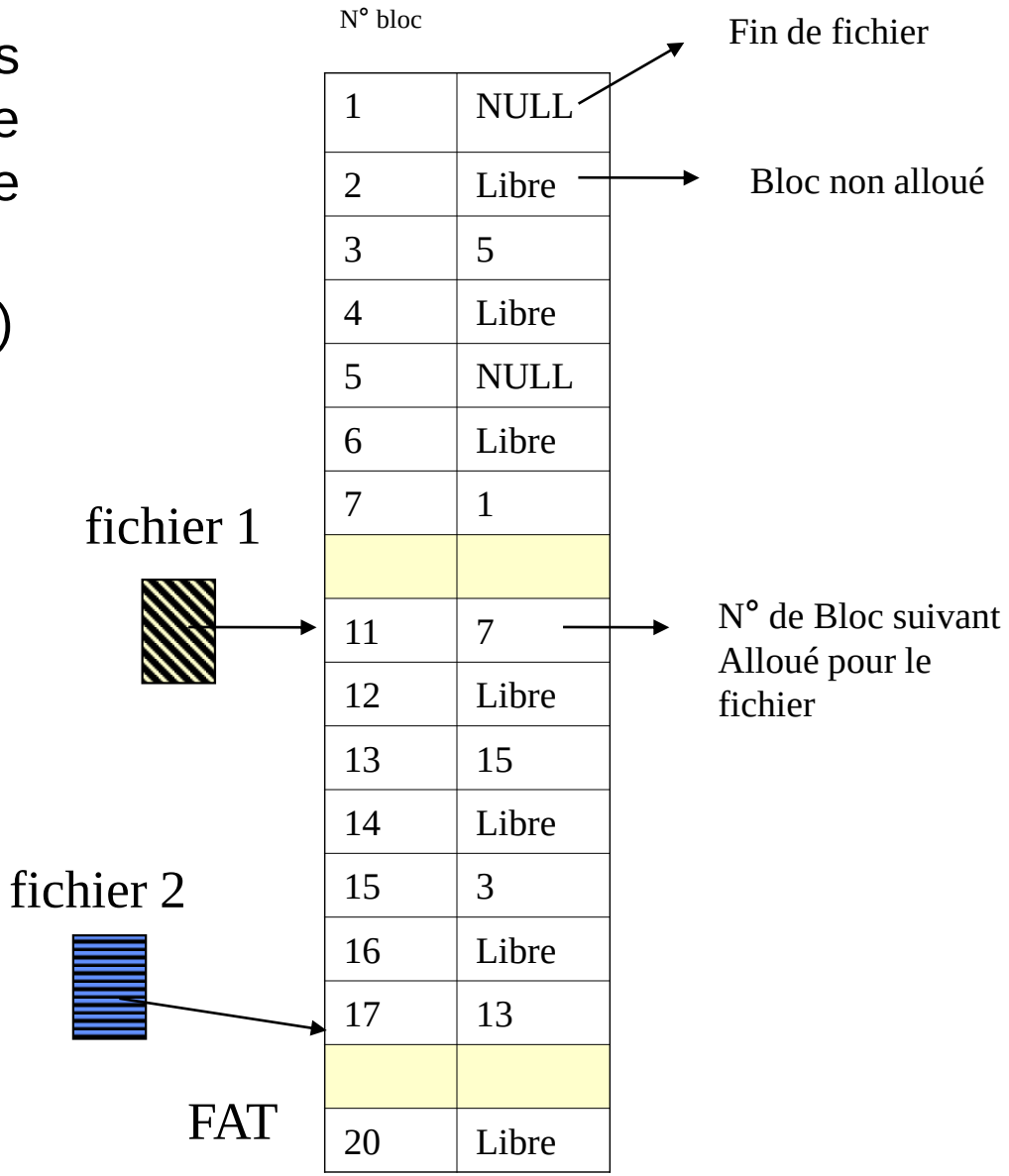


Allocation par bloc chainée : variante

- Une table d'allocation des fichiers (File allocation table - FAT) regroupe l'ensemble des chainages.

(exemple systèmes windows)

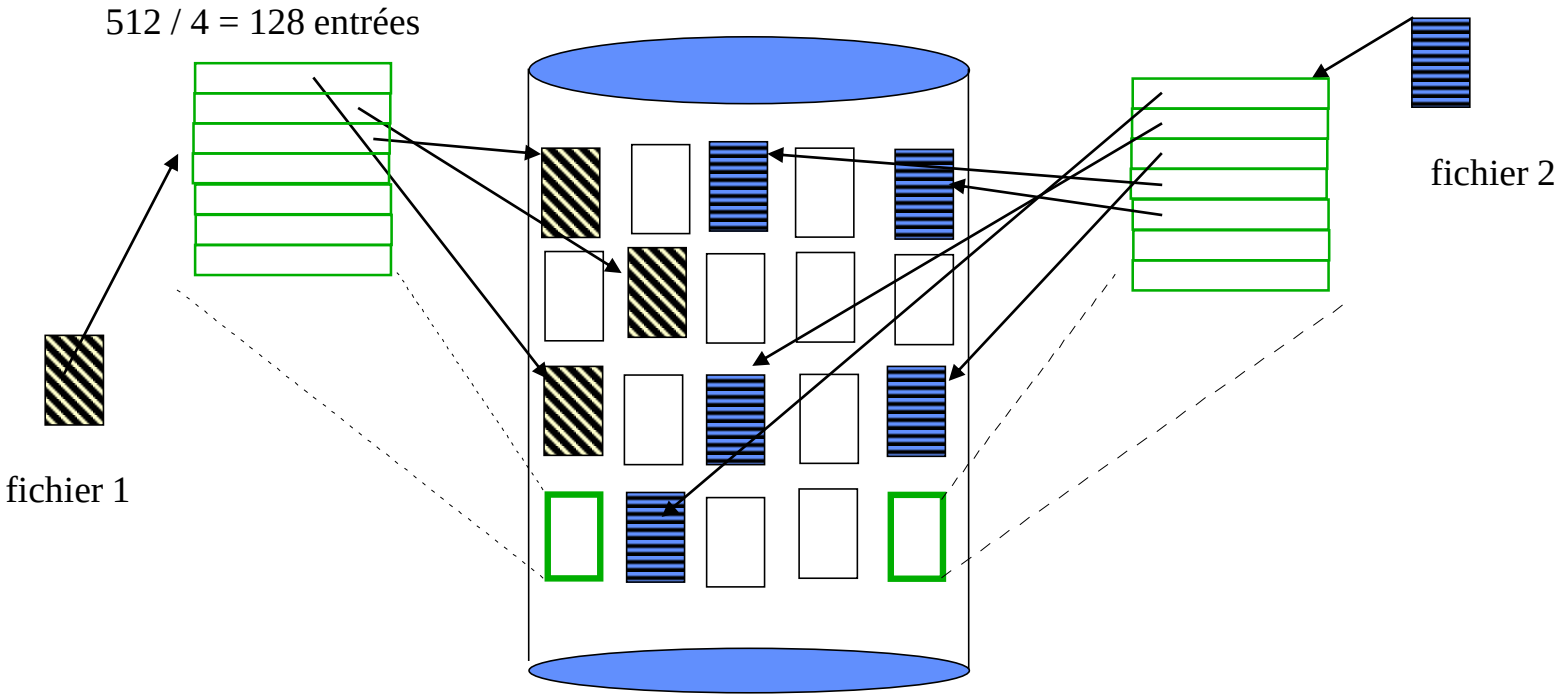




N° bloc	
1	NULL
2	Libre
3	5
4	Libre
5	NULL
6	Libre
7	1
11	7
12	Libre
13	15
14	Libre
15	3
16	Libre
17	13
20	Libre

Allocation indexée

- Les adresses des blocs physiques constituant un fichier sont rangées dans une table appelée index, elle-même contenue dans un ou plusieurs blocs disque



- Supporte bien l’accès direct
- « gaspillage de place » dans le bloc d’index

Système de gestion de fichiers LINUX

- Structure d'un fichier Linux
- Partition Linux

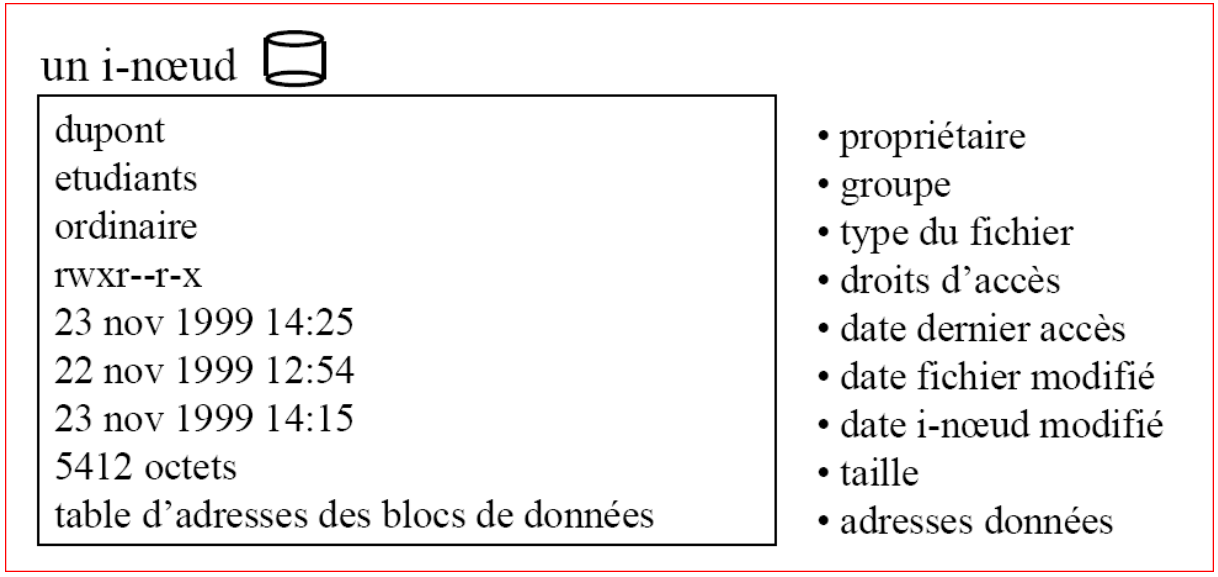
- Structure de gestion des fichiers dans un processus
- Mécanisme du buffer cache

Fichier Linux

- Identifié par un nom, sans structure logique (suite d'octets)
- La méthode d 'allocation mise en œuvre est de type allocation indexée.
- Un fichier Linux est composé d'un descripteur appelé « inode » et de blocs physiques, qui sont soit des blocs d'index, sont des blocs de données. Les blocs de données sont alloués au fur et à mesure de l'extension du fichier.
- Un bloc est identifié par un numéro codé sur 4 octets. La taille d'un bloc est un multiple de la taille d'un secteur (512 octets)

Fichier Linux : inode

- Structure stockée sur le disque, allouée à la création du fichier et répérée par un numéro
- Contient les attributs du fichier :
  - Nom
  - Type : fichiers normaux, répertoires, *périphériques, tubes nommés , sockets*
  - Droits d'accès
  - Heures diverses
  - Taille du fichier en octets
  - Table des adresses des blocs de données



Fichier Linux : structure

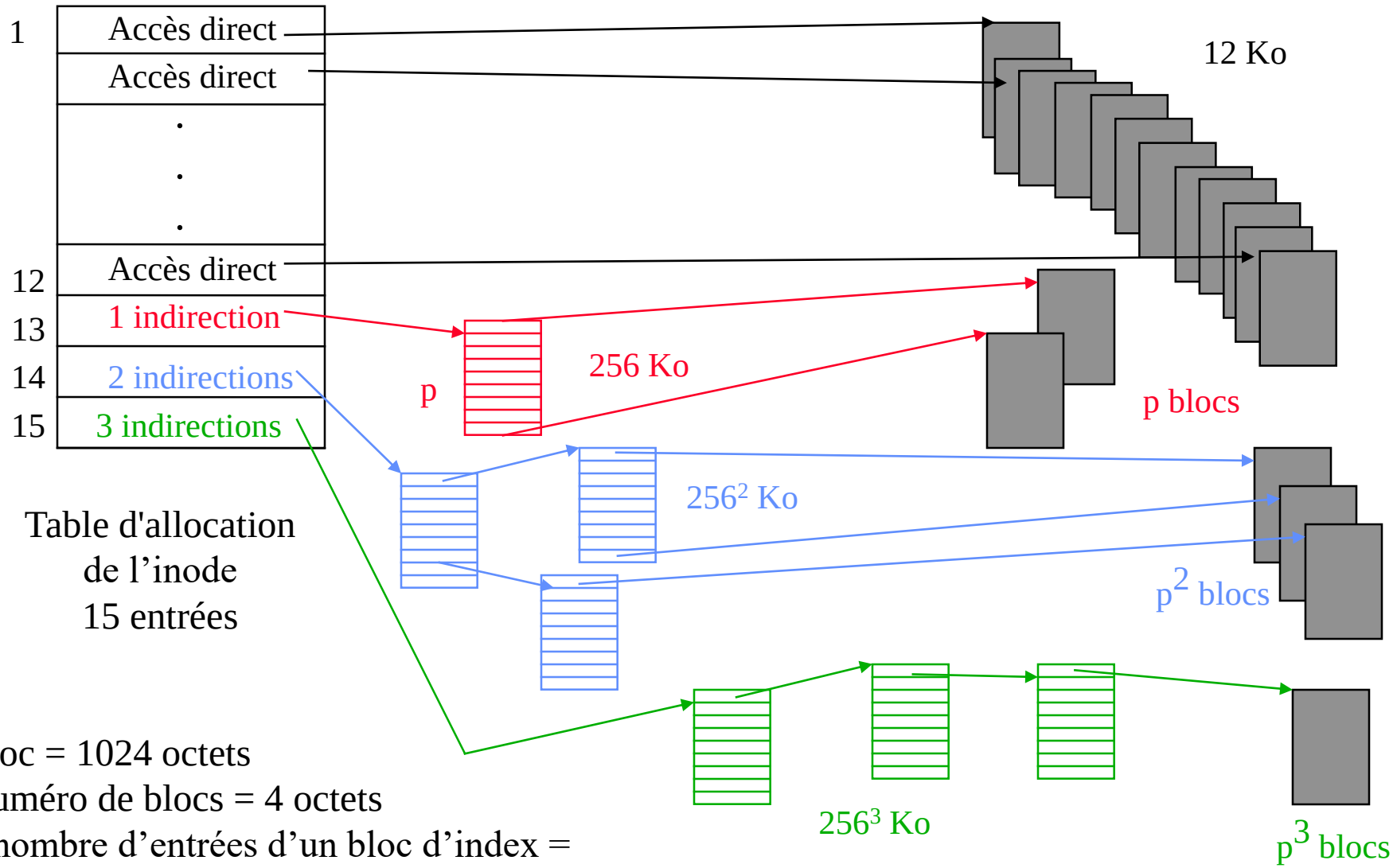
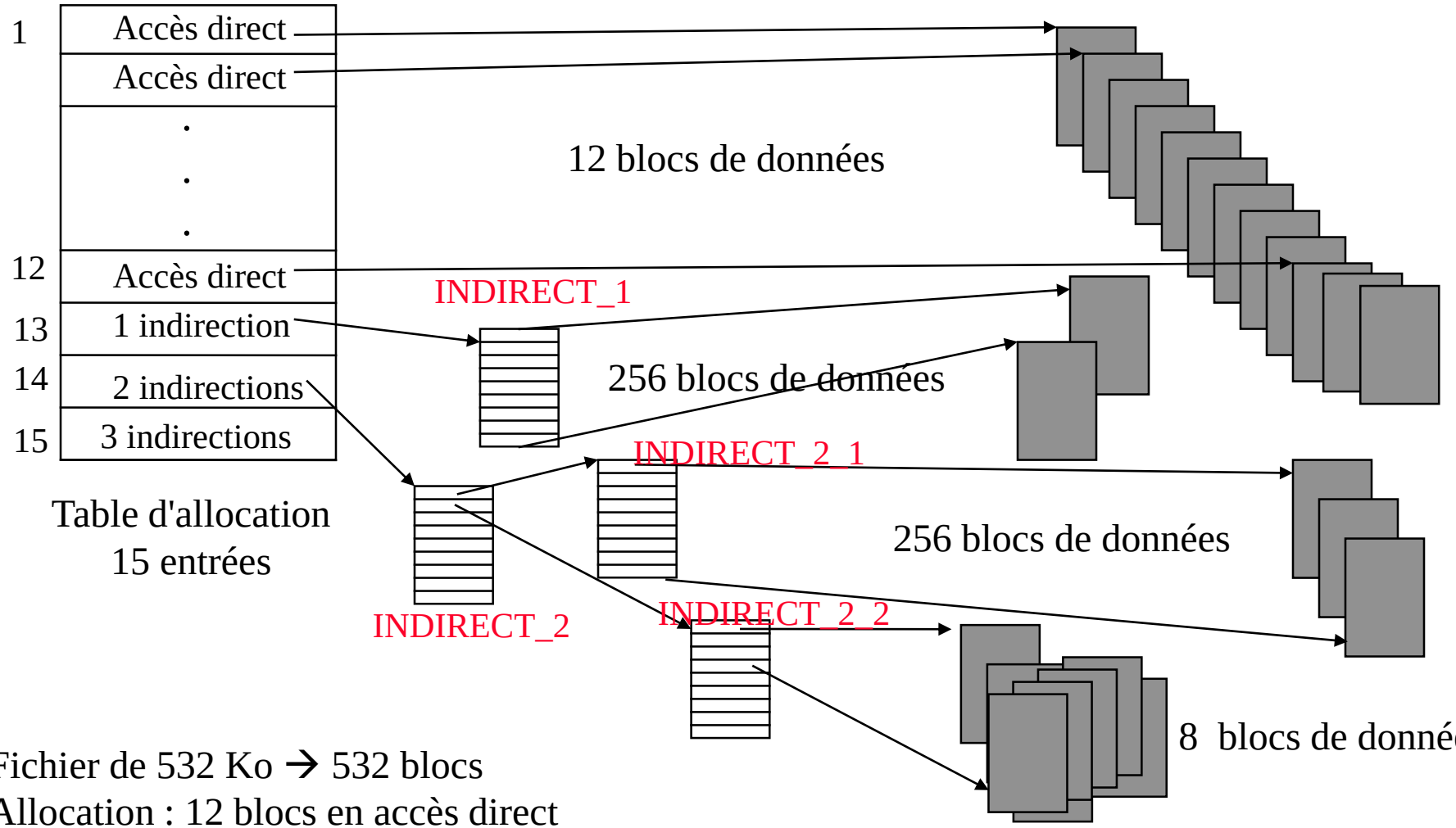


Table d'allocation
de l'inode
15 entrées

Bloc = 1024 octets
Numéro de blocs = 4 octets
p nombre d'entrées d'un bloc d'index =
taille bloc / taille numéro de bloc

EXEMPLE

Bloc = 1024 octets ; adresse de bloc = 4 octets → 256 entrées dans le bloc d'index



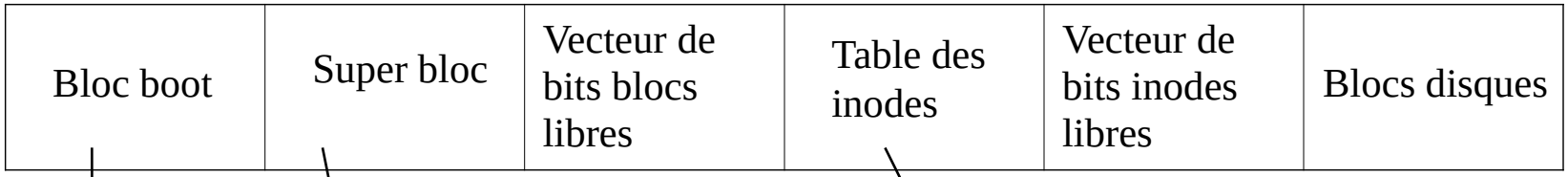
Fichier de 532 Ko → 532 blocs

Allocation : 12 blocs en accès direct

: 256 blocs de données pointés par le bloc index INDIRECT 1

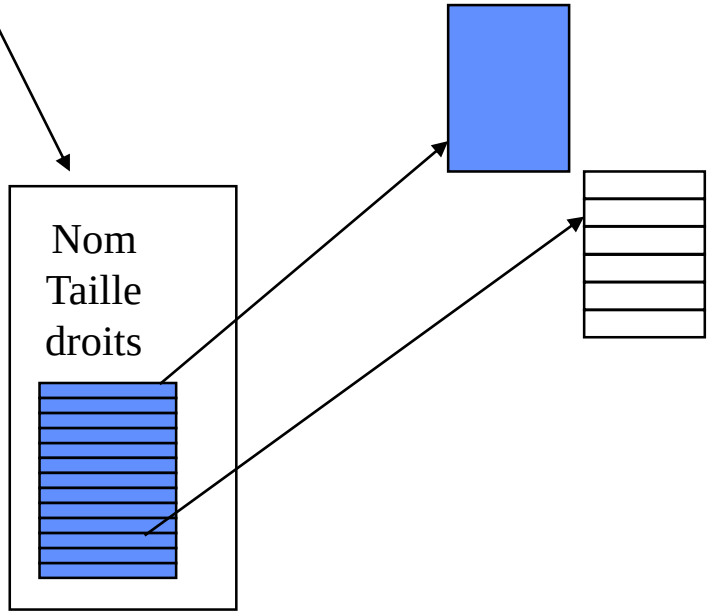
: restent 532 – 12 – 256 = 264 blocs . Tous ces blocs sont pointés à partir du bloc d'index INDIRECT_2. 2 blocs d 'index INDIRECT_2_1 et INDIRECT_2_2 sont nécessaires à ce niveau

Partition Linux : structure



1er secteur
contient l'amorçage du SE
et description de la partition

Nom de la partition
Taille de la partition
Taille d'un bloc
Nombre d'inodes
Nombre d'inodes libres
Nombre de blocs libres



inode

Lecture d'un fichier : le buffer cache

while(read (fd, &un_eleve, sizeof(un_eleve)) > 0);

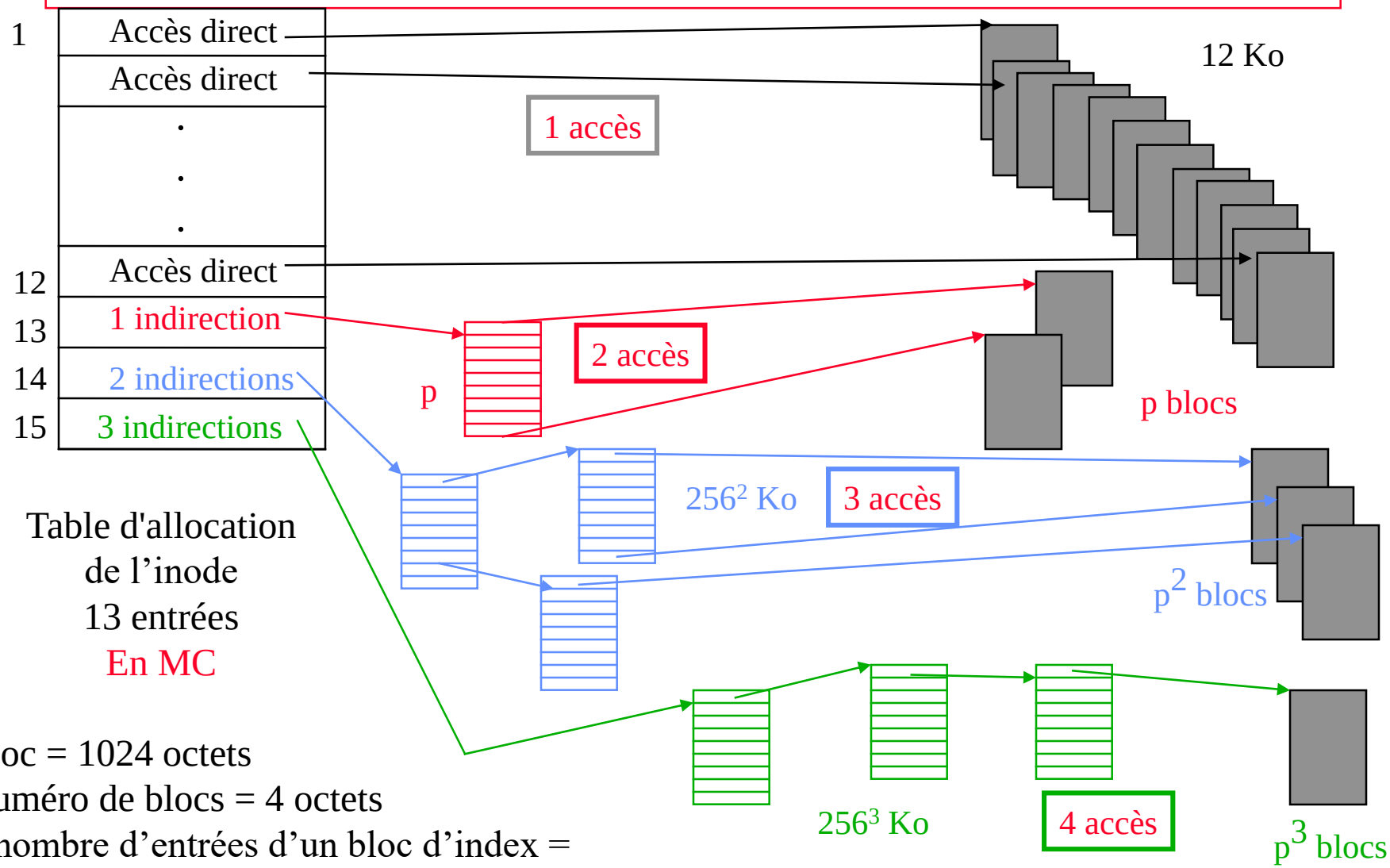
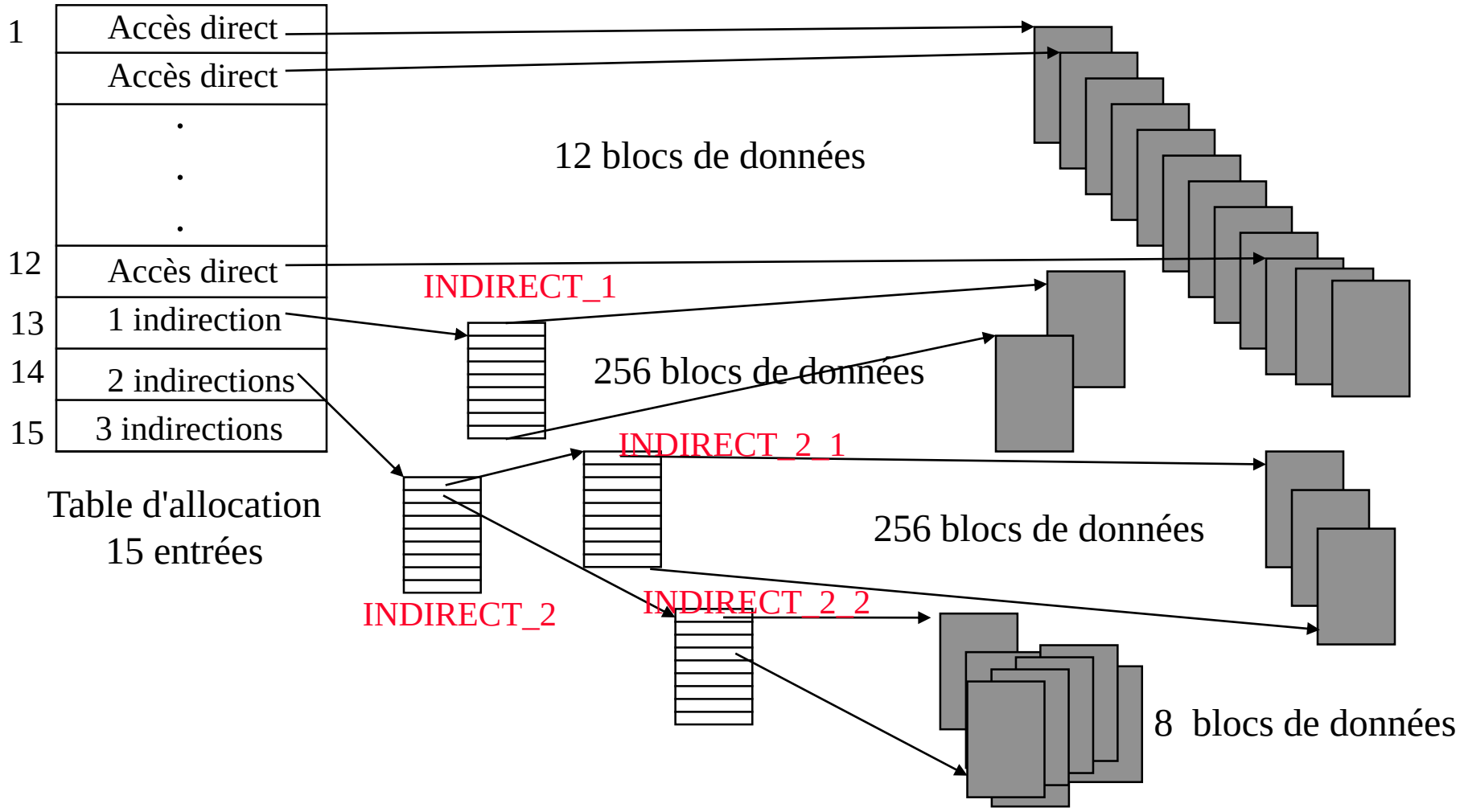


Table d'allocation
de l'inode
13 entrées
En MC

Bloc = 1024 octets
Numéro de blocs = 4 octets
p nombre d'entrées d'un bloc d'index =
taille bloc / taille numéro de bloc

EXEMPLE

Bloc = 1024 octets ; adresse de bloc = 4 octets → 256 entrées dans le bloc d'index



Nombre d'accès disque pour lire le fichier :

12 + (2 * 256) + (3 *256) + (3 * 8) = 1 316 AD

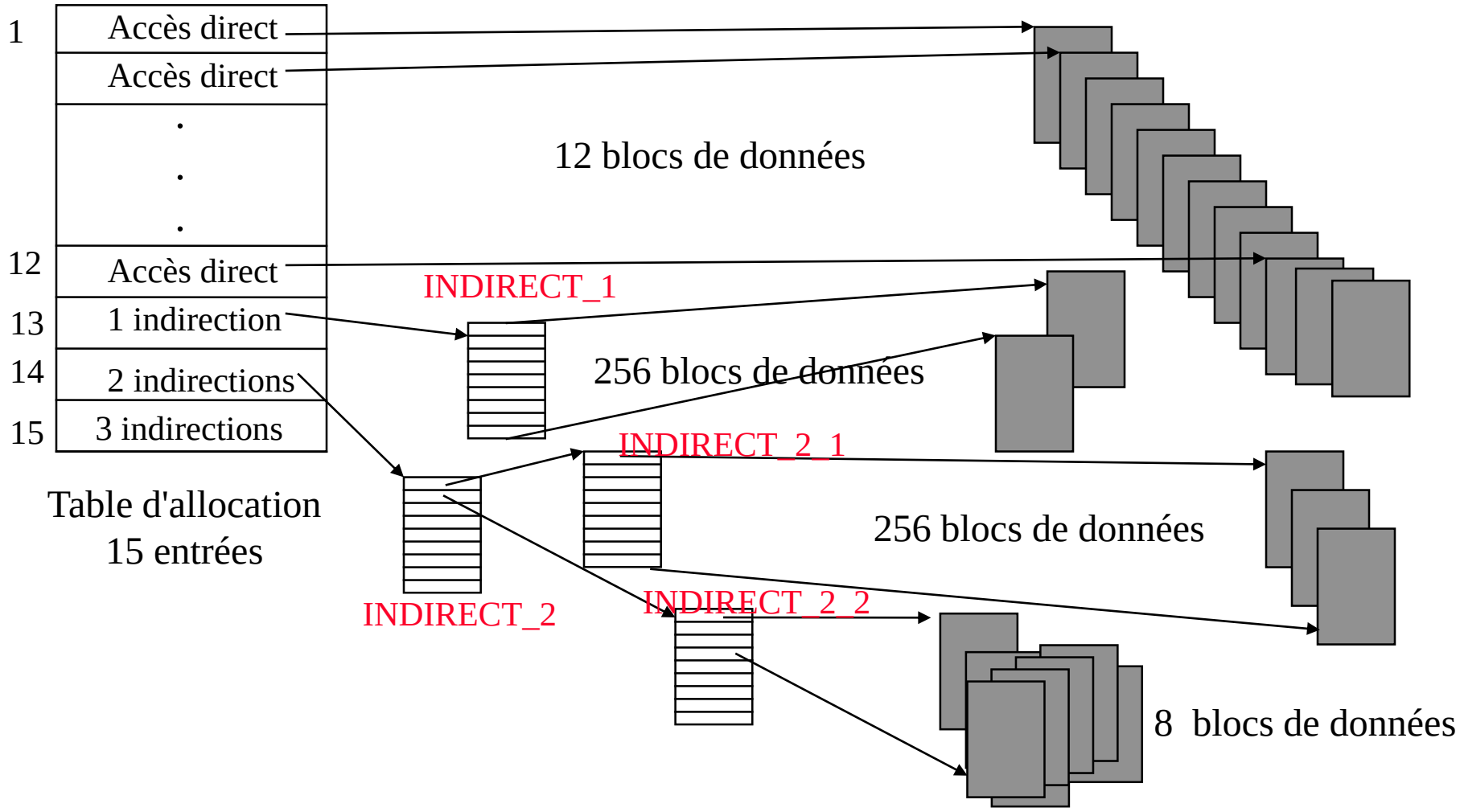
Temps de lecture = 1316 * 10 ms = 13160 ms = 13 s

Lecture d’un fichier : le buffer cache

- Le système maintient une liste de tampons mémoire qui joue le rôle de cache pour les blocs du disque et permet de réduire les entrées/sorties.
- La taille d'un tampon est égale à la taille d'un bloc disque. Il est identifié par un numéro de bloc physique et numéro de périphérique.
- Lorsque le système doit lire un bloc depuis le disque :
  - Il cherche d'abord si le bloc est déjà présent dans la liste des tampons mémoire
  - Si non, il prend un tampon libre et copie le bloc disque dans le tampon.
  - Si tous les tampons sont occupés, il libère un tampon en choisissant le moins récemment accédé.

EXEMPLE

Bloc = 1024 octets ; adresse de bloc = 4 octets → 256 entrées dans le bloc d'index



Nombre d'accès disque pour lire le fichier :

12 + (1 + 256) + (2 + 256) + (1 + 8) = 539 AD

Temps de lecture = 539 * 10 ms = 5390 ms = 5 s (divisé selon facteur 2)