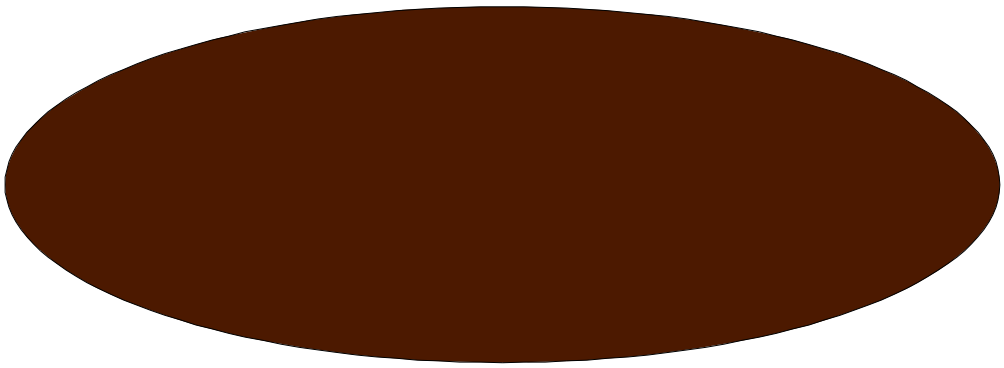
Cours « système d'exploitation »
2ème année
IUT de Caen, Département d'Informatique
Année 2000 − 2001
(François Bourdon)

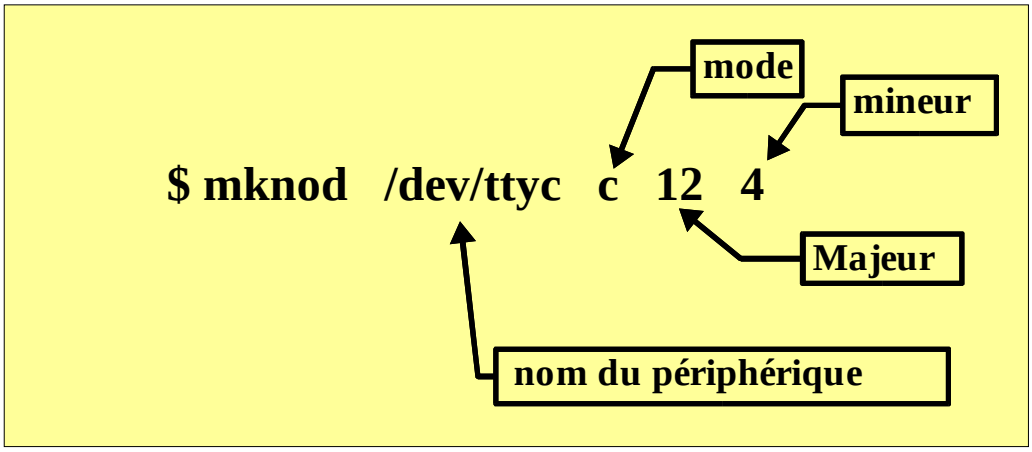


Plan

1.2 Les Entrées/Sorties et le SGF

L'accès aux périphériques tels que ports asynchrones, ports parallèles, unités de disques, bandes magnétiques, réseau ethernet, ou à des pseudo−périphériques comme l'espace mémoire mémoire, l'espace de pagination, est effectué sous UNIX par l'intermédiaire de fichiers spéciaux.

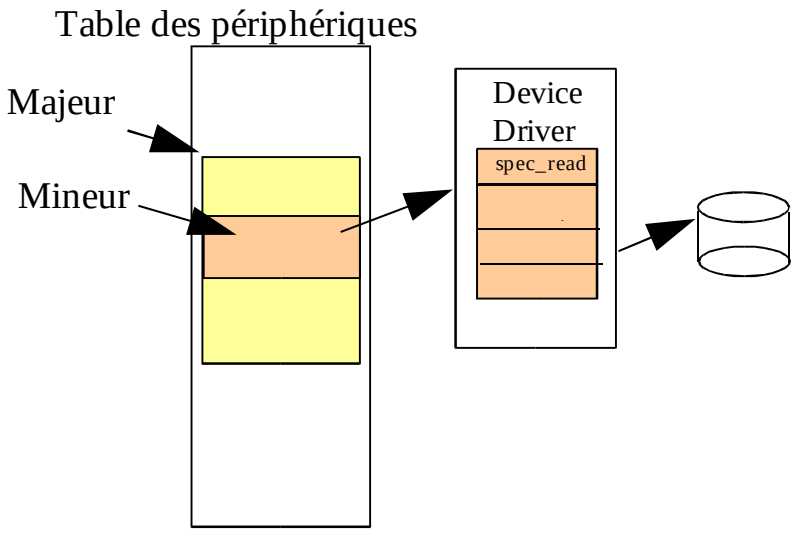
Ces fichiers sont situés dans le répertoire */dev* et sont créés à l'aide de la commande *mknod*.



Ils ne contiennent pas de données mais leur i−noeud (i−node) détermine l'accès au pilote d'entrée−sortie correspondant par deux numéros (*majeur* et *mineur*) qui correspondent à un index dans une structure du noyau pour les pilotes de type **bloc** ou pour ceux de type **caractère**.

Le **majeur** fait référence au type de périphérique, par exemple un lecteur de disquette, un lecteur de bande, un disque dur ou encore un terminal, ...

Le **mineur** permet d'adresser un périphérique particulier, par exemple un lecteur de disquette à un format particulier, une partition du disque numéro 0, une bande magnétique, le port série n°1, ...



Le numéro de **majeur adresse une table, le mineur est une information passée aux routines de ce driver, permettant de spécifier une entité gérée, éventuellement de passer des informations supplémentaires au driver (par exemple accéder à la bande magnétique numéro 0 sans rebobinage automatique).**

Dans UNIX il existe deux façons de gérer un périphérique de stockage (E/S) :

- Soit avec une interface de bufferisation gérée par le système (mémoire de cache). On dira que le périphérique est géré en **mode bloc**.

- Soit directement, sans aucune bufferisation. On dira alors que le périphérique est géré en **mode caractère**.

Les fichiers spéciaux **blocs** servent à modéliser les périphériques dont l'unité de transfert est le bloc ; c'est le cas généralement des disques.

Les fichiers spéciaux **caractères** servent à modéliser les périphériques dont l'unité de transfert est le caractère ; c'est le cas du clavier ou de l'imprimante.

La plupart des noms de fichiers spéciaux sont standard. En voici quelques exemples :

/dev/mem pour la mémoire vive physique,
/dev/kmem pour l'espace mémoire du noyau UNIX,
/dev/null pour le périphérique null,
/dev/ttyxx ou **/dev/term/xx** pour les ports asynchrones,
/dev/swap pour l'espace disque du swap,
/dev/drum pour l'espace disque de pagination,
/dev/pty[p−z][0−9a−f] et **/dev/tty[p−z][0−9a−f]** pour les pseudo−terminaux maîtres et esclaves.

Pour les disques, les noms ne sont pas standard mais on retrouve certaines règles. Par exemple */dev/sd0c* désigne la partition *c* du disque SCSI −0.

Un fichier spécial est donc caractérisé de façon unique par :

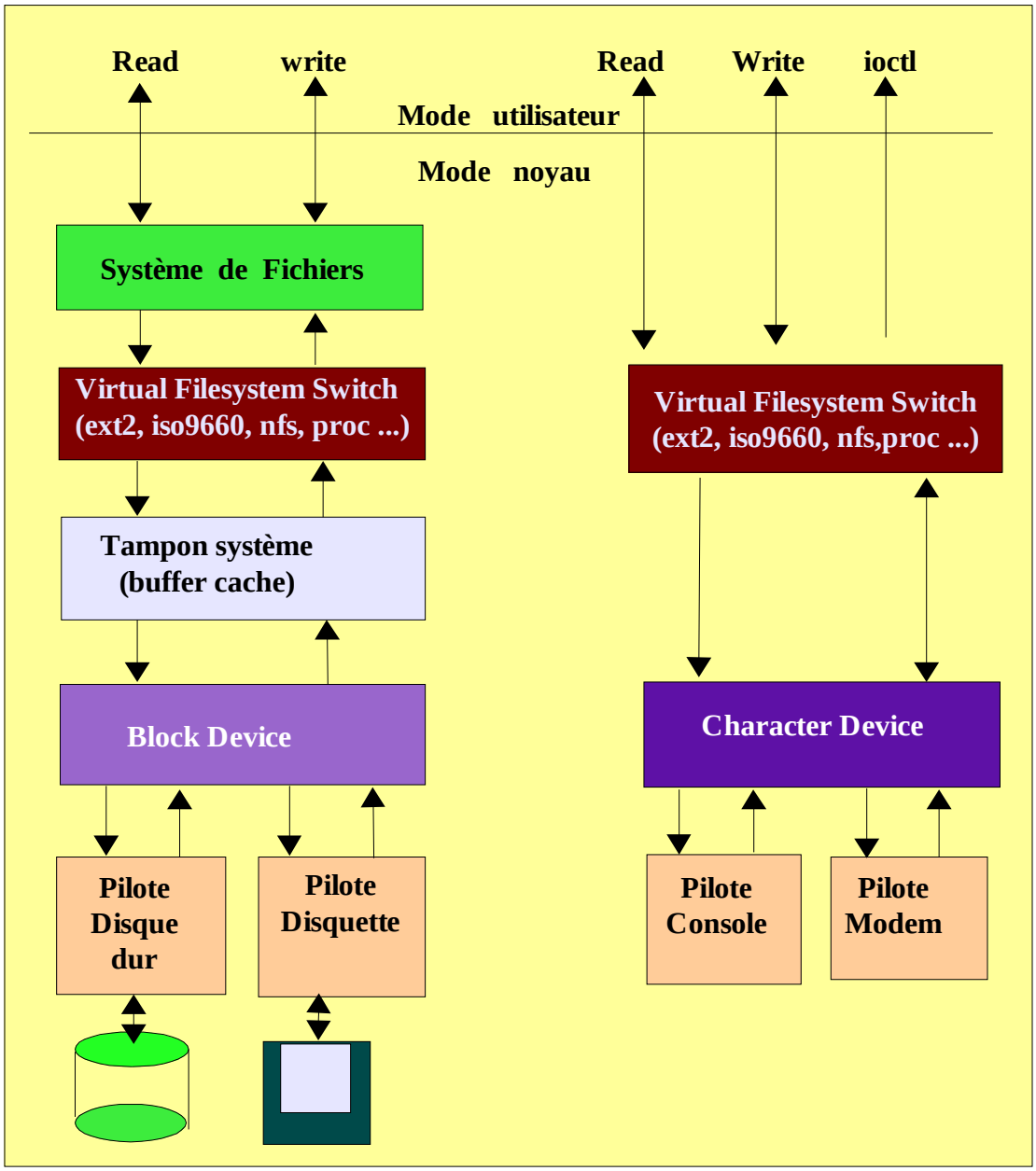
- Son type (bloc ou caractère).
- Son majeur (type de périphérique).
- Son mineur (une instance de ce périphérique).

Exemples :

```
crw-------  1  c1      cours     25, 2  Jul 6 15:36   /dev/term/2
crw-------  1  c2      cours     25, 3  Jul 6 15:37   /dev/term/3
brw-------  2  sysinfo sysinfo   1, 0  Dec 19 1995   /dev/dsk/0s0
brw-------  2  sysinfo sysinfo   1, 1  Dec 19 1995   /dev/dsk/0s1
crw-rw-rw-  1  root    sys       4, 2  Jul 6 15:45   /dev/null
```

/dev/term/2 est le fichier spécial associé au terminal numéro deux. Il est de type caractère comme l'indique la lettre '**c**'. Il n'a pas de taille, mais est caractérisé par le majeur **25** et le mineur **2**. **/dev/dsk/0s0** correspond à la partition **0** du disque **0**, géré en mode bloc ('**b**'). Il est caractérisé par le majeur **1** et le mineur **0**.

/dev/null est le fichier spécial associé à un pseudo−périphérique. Tout ce qui est écrit sur ce fichier disparaît et toute lecture sur ce fichier retourne toujours un code de fin de fichier.



Le système de fichiers est interfacé au−dessus de l'interface *bloc* qui utilise les tampons **cache** du système. Le rôle du **cache** est de différer les écritures et d'anticiper les lectures.

1.3 E/S tamponnées : le Buffer Cache

Le noyau maintient des fichiers dans des périphériques de stockage de masse tels que les disques, et il permet aux processus de ranger de nouvelles informations ou de rappeler des informations précédemmment rangées.

Quand un processus cherche à accéder aux données d'un fichier, le noyau ramène les données en mémoire principale où le processus peut les examiner, les altérer, et demander que les données soient de nouveau sauvegardées dans le système de fichiers.

Au moment où il ramène les données en mémoire, le noyau doit aussi ramener des données auxiliaires pour pouvoir les manipuler. Par exemple il lit un i−noeud en mémoire quand il veut accéder aux données d'un fichier et retourne l'i−noeud modifié au système de fichiers quand il veut mettre à jour la structure du fichier.

Le noyau manipule ces données auxiliaires sans la connaissance explicite ou sans requête du processus en exécution.

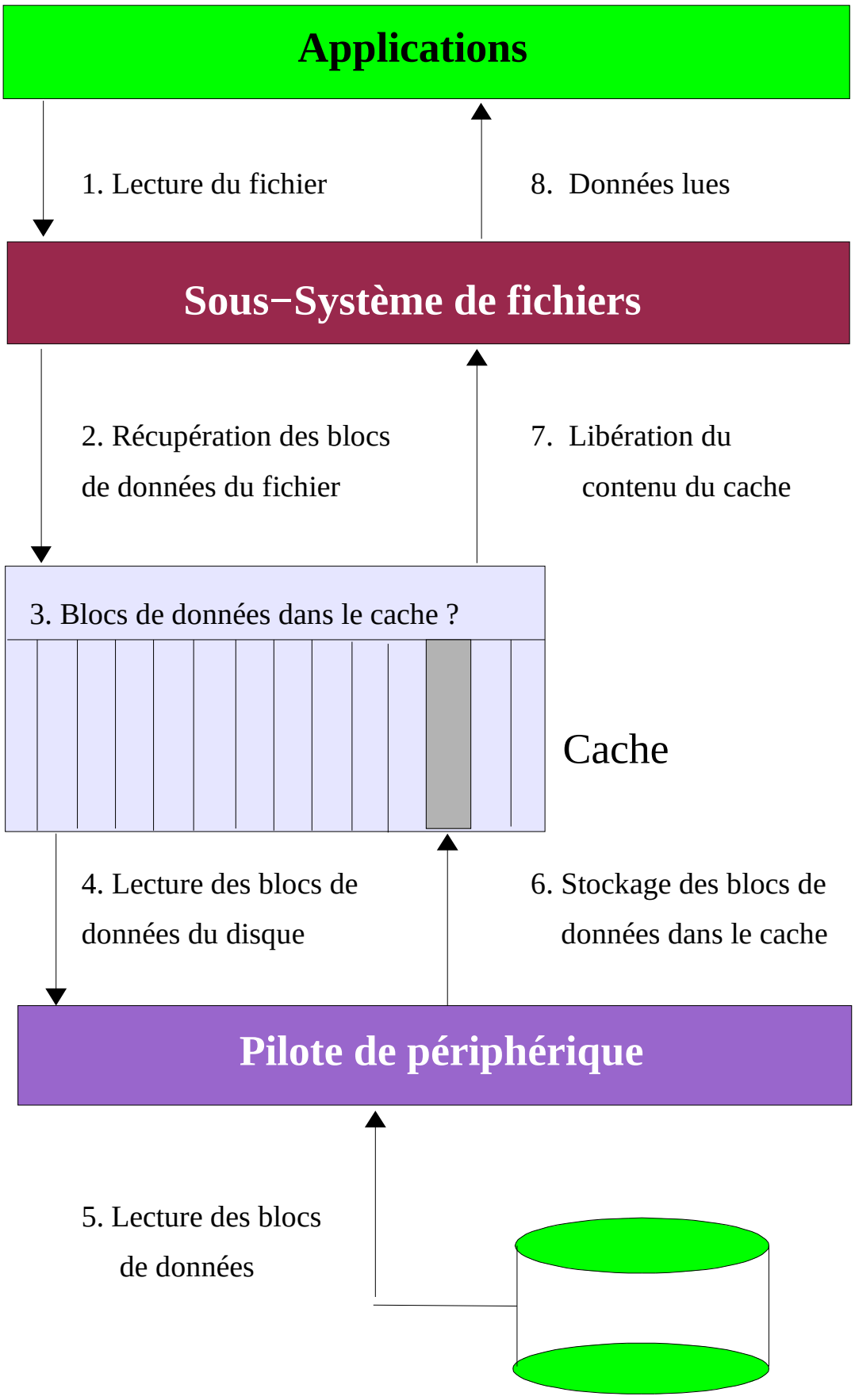
Le noyau tente de minimiser la fréquence d'accès au disque en gérant une réserve de tampons en données internes, appelée *buffer cache*, qui contient les données du disque les plus récemment utilisées.

Principe

- **Sur une lecture de données d'un disque, le noyau tente de lire depuis le *buffer cache*. Si les données sont déjà dans le cache, le noyau n'a pas à lire depuis le disque. Sinon, le noyau lit les données depuis le disque et les met dans le cache, en utilisant un algorithme qui essaye de mémoriser dans le cache autant de données que possible.**

- **De même les données devant être écrites sur le disque sont cachées pour qu'elles soient présentes si le noyau plus tard essayait de les relire.**

- **Le noyau tente aussi de minimiser la fréquence des opérations d'écriture sur le disque en déterminant si les données doivent vraiment être stockées sur le disque ou si ce sont des données transitoires qui seront bientôt remplacées.**


Applications
1. Lecture du fichier
8. Données lues
Sous−Système de fichiers
2. Récupération des blocs de données du fichier
7. Libération du contenu du cache
3. Blocs de données dans le cache ?
Cache
4. Lecture des blocs de données du disque
6. Stockage des blocs de données dans le cache
Pilote de périphérique
5. Lecture des blocs de données

Pour cela le cache est structuré en **tampons** constitués chacun d'eux des parties :

- **« en−tête »** pour l'identification du tampon et
- **« tableau mémoire »** pour les données du disque.

Un tampon est partageable par plusieurs processus, par contre un bloc disque ne peut appartenir qu'à un tampon à la fois.

L'état du tampon courant est une combinaison des états suivants :

- Le tampon est actuellement **verrouillé.**
- Le tampon contient des **données valides.**
- Le noyau doit écrire le contenu du tampon sur le disque avant de le réassigner ; cet état est connu sous le nom d'**écriture différée.**
- Le noyau est actuellement en train de lire ou d'écrire le contenu du tampon sur le disque **(actif).**
- Un processus est actuellement **en attente** que le tampon devienne libre.

Entête de TAMPON (simplifié)

N° (du pilote principal et secondaire) du périphérique dont le fichier provient
N° du bloc de données géré par le tampon
Etat courant du tampon (libre, occupé ...)
Position du bloc de données sur le disque
Pointeur sur le tampon suivant dans la files
Pointeur sur le tampon précédent dans la file
Pointeur sur le tampon suivant dans la liste des tampons libres
Pointeur sur le tampon précédent dans la liste des tampons libres

Pour OPTIMISER le fonctionnement du CACHE, les techniques de LISTES CHAINEES et de TABLE de HACHAGE (hash lists) sont combinées :

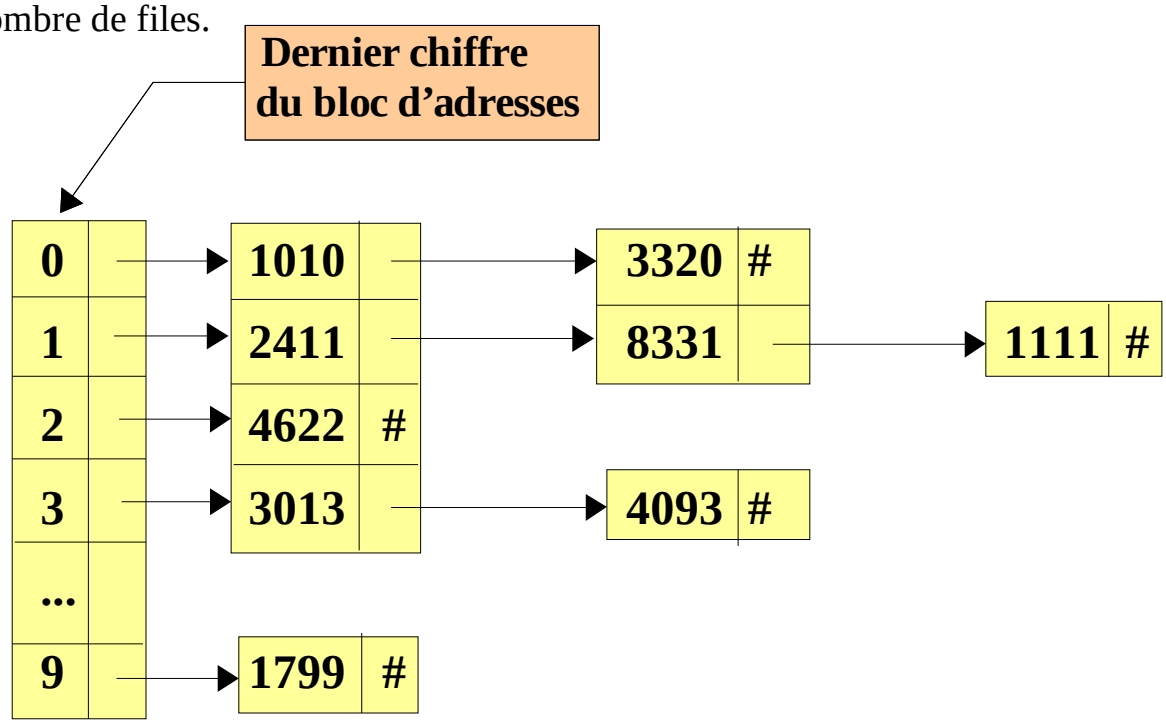
Accés aux blocs de données => TABLE de HACHAGE

Gestion des blocs libres => LISTES CHAINEES

Accés aux blocs de DONNEES (TABLE de HACHAGE)

Quand le noyau accède à un bloc disque présent dans le cache, il recherche le tampon qui contient ce bloc.

Plutôt que de parcourir en totalité la réserve de tampons, il organise les tampons dans des files séparées (listes partielles), accédées selon un adressage calculé (hash−coding ou table de hash) par une fonction paramétrée par les numéros de bloc et de périphérique. Les listes partielles (files) énumèrent les tampons appartenant à des blocs de données ayant une caractéristique commune. Par exemple, une telle liste ne cite que les tampons dont le numéro de bloc se termine par le chiffre 1. La liste suivante énumère ceux qui se termine par 2 et ainsi de suite. On peut aussi prendre le n° du bloc modulo le nombre de files.



Si aucun tampon correspond au bloc recherché on alloue un tampon libre (le moins récemment utilisé).

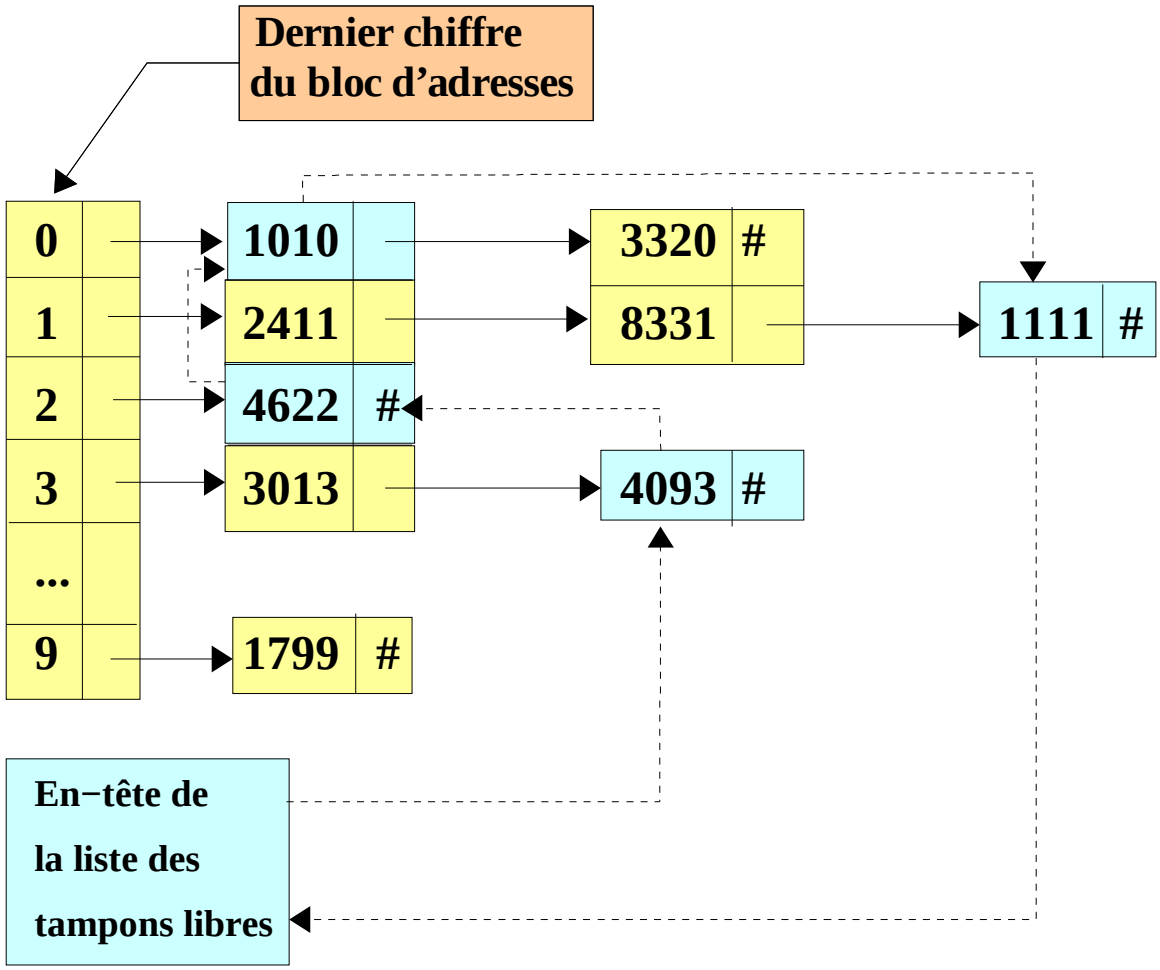
Le nombre de tampons dans une file varie suivant l'utilisation du système. Le noyau doit utiliser une fonction de *hashing* qui distribue les tampons uniformément dans l'ensemble des files, encore que cette fonction doit être simple pour ne pas ralentir le système.

L'administrateur peut définir le nombre de ces files à la génération du système.

L'en−tête d'un tampon contient deux ensembles de pointeurs, utilisés par les algorithmes d'allocation de tampons pour gérer l'ensemble de la structure associée à la réserve de tampons.

Les blocs tampons sont ainsi reliés entre eux par un double chaînage des blocs libres et des blocs occupés.

Le noyau gère donc une liste (circulaire, doublement chaînée) des tampons libres qui préserve l'ordre du moins récemment utilisé.



Si aucun tampon correspond au bloc recherché on alloue un tampon libre, qui correspond donc au moins récemment utilisé.

Les algorithmes sur les tampons combinent plusieurs idées simples pour fournir un mécanisme de cache sophistiqué.

Il existe plusieurs politiques de recopie des blocs du *buffer cache* sur le disque lorsqu'il ont été modifiés, en fonction d'un compromis entre la sécurité et la vitesse.

Dans UNIX :

- **les noeuds d'index et les répertoires sont recopiés immédiatement après leur modification ;**
- **les blocs de données ordinaires sont recopiés si leur tampon atteind la fin de la liste des LRU (blocs les plus utilisés) ;**
- **ou automatiquement par un démon toutes les 15 secondes. On peut déclencher ce démon par l'appel système : « SYNC ( ) »**

Les tampons du système MS−DOS sont à recopie immédiate.

Dans LINUX un CLUSTER est un ensemble de tampons mémoire dont les contenus sont contigüs en mémoire. Ainsi une lecture/écriture de plusieurs tampons mémoire peut être effectuée en une seule E/S.

Le noyau utilise le *buffer cache* de plusieurs façons :

- **Il transmet des données entre des programmes d'application et le système de fichiers via le *buffer cache*,**

- **et il transmet des données auxiliaires du système telles que les i−noeuds entre les algorithmes de plus haut niveau et le système de fichiers.**

- **Il l'utilise aussi quand il lit des programmes en mémoire en vue de leur exécution (cf. le chapitre sur « la gestion de la mémoire »).**

D'autres algorithmes qui cachent les i−noeuds et les pages de mémoire utilisent aussi des techniques similaires à celles décrites pour le *buffer cache*.

1.4 Le Système de Fichiers Virtuel (SFV)