Cours 2 : La gestion des fichiers

Plan du cours

Notion des Fichiers

Organisations Physiques

Système de Gestion de Fichiers SGF

Technique d'Allocation sur le Disque

Optimisation des Performances &Sécurité

Section 1 : Notion des Fichiers

Notion de Fichiers

Définition

Un fichier est une unité de stockage logique de l'information

Des contenus pouvant être de différents types, en particulier

- Texte
- Son
- Image
- Vidéos

- Nommage de fichier =Nom Fichier + Extension Fichier
- **Ex**. Etudiant.doc

Attributs des fichiers:

- Création, Ecriture/lecture, Supression, Concatenation,..

Fichiers comme abstraction de ressources

Un fichier:

- Abstraction des propriétés physiques des dispositifs de stockage
- Les fichiers sont utilisés pour représenter des données sur des dispositifs de stockage (par exemple un disque)
- La correspondance est établie par un SE

Les opérations sur les Fichiers

Fichier : Point d'entrée-sortie

Informations sur les fichiers

- stat(), fstat()

Création ou acquisition des droits de lecture/ écriture d'un fichier existant et attribution d'un Identifiant unique { descripteur de fichier}

- open()

Opérations lecture/ écriture, positionnement

- read(), write(), lseek()

Partage/verrouillement d'un fichier

- dup2(), fcntl()

Fermeture

- close()

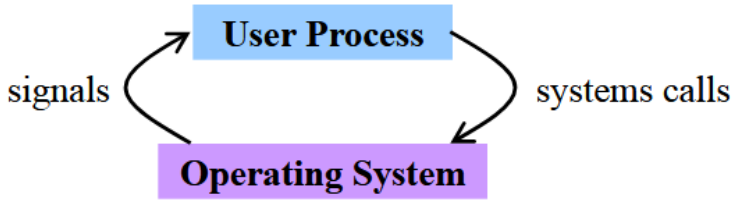
Destruction d'un fichier

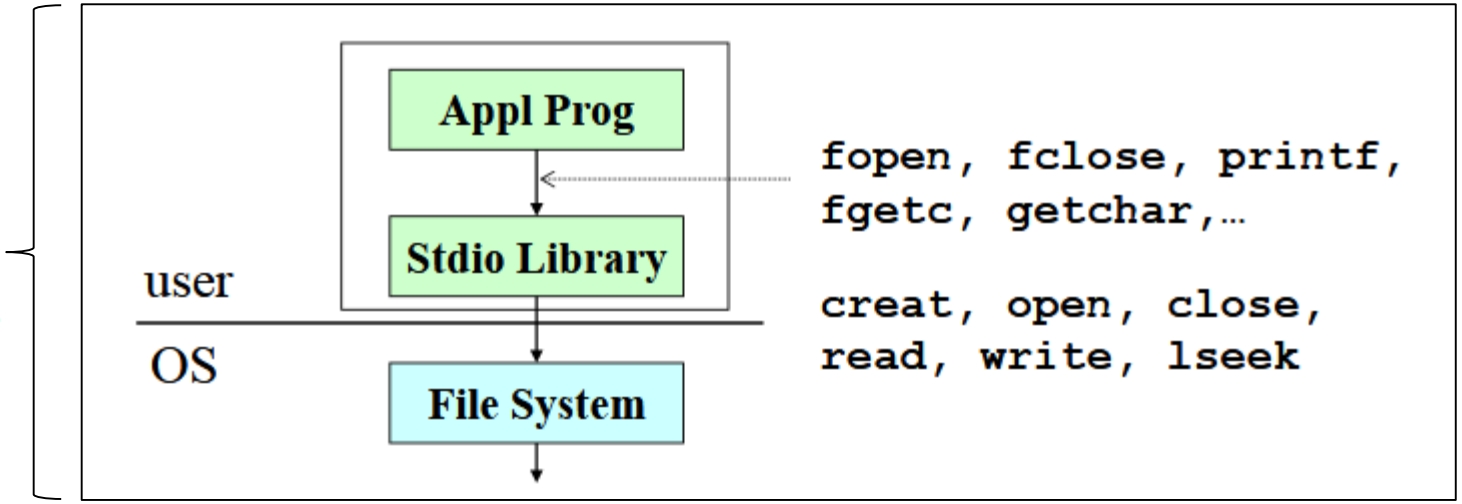
- unlink()

Bibliothèque d'E/S standard

Bibliothèque d'E/S: gère des appels système liés aux E/S.

- **Portabilité**
  - Prise en charge des E/S génériques pour les programmes C
  - Implémentations spécifiques pour différents systèmes d'exploitation hôte
  - Invoque les appels système spécifiques au SE pour les E/S
- **Abstraction pour les programmes C**
  - Concept de flux, Entrée ligne par ligne, Sortie formatée
- **Optimisation supplémentaire**
  - E/S bufférisées
  - Écriture sécurisée





Notion de Fichier

Question Quelle abstraction le programme utilise-t-il pour accéder à un fichier sur le disque?

a) Le chemin d'accès du fichier et son nom

b) Une séquence de blocs contigus sur le disque

c) Un ensemble de blocs non-contigus sur le disque

d) Un numéro de secteur et de cylindre sur le disque

e) FAT-16

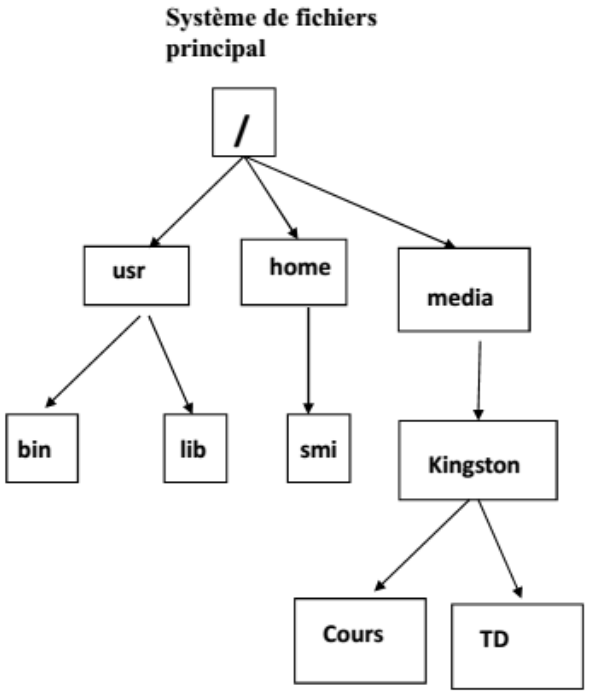
f) FAT-32

g) Aucune de ces réponses

Section 2 : Organisations physiques

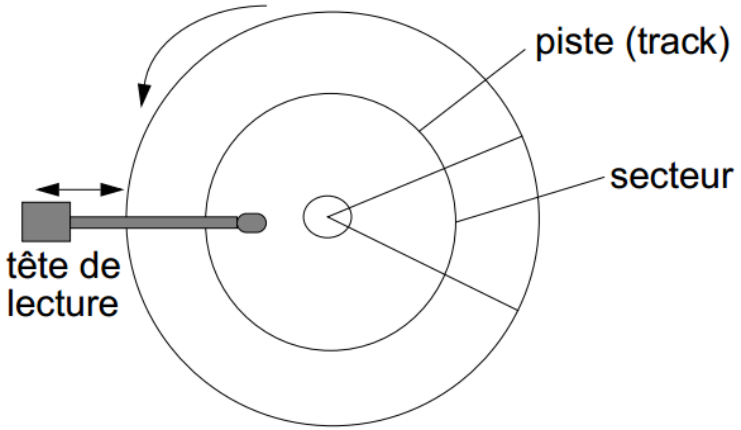
Vue logique vs Vue physique

- **Abstraction**



- Le **SGF** est vue comme une arborescence de dossiers/ fichiers

- **Implémentation ??**
  - Représentation des fichiers
  - Il faut un élément hardware **->** persistance



Exp. Secteur= 512 KB

Question: Comment implémenter ce type d'abstraction par le SE

Allocation du disque : le bloc physique

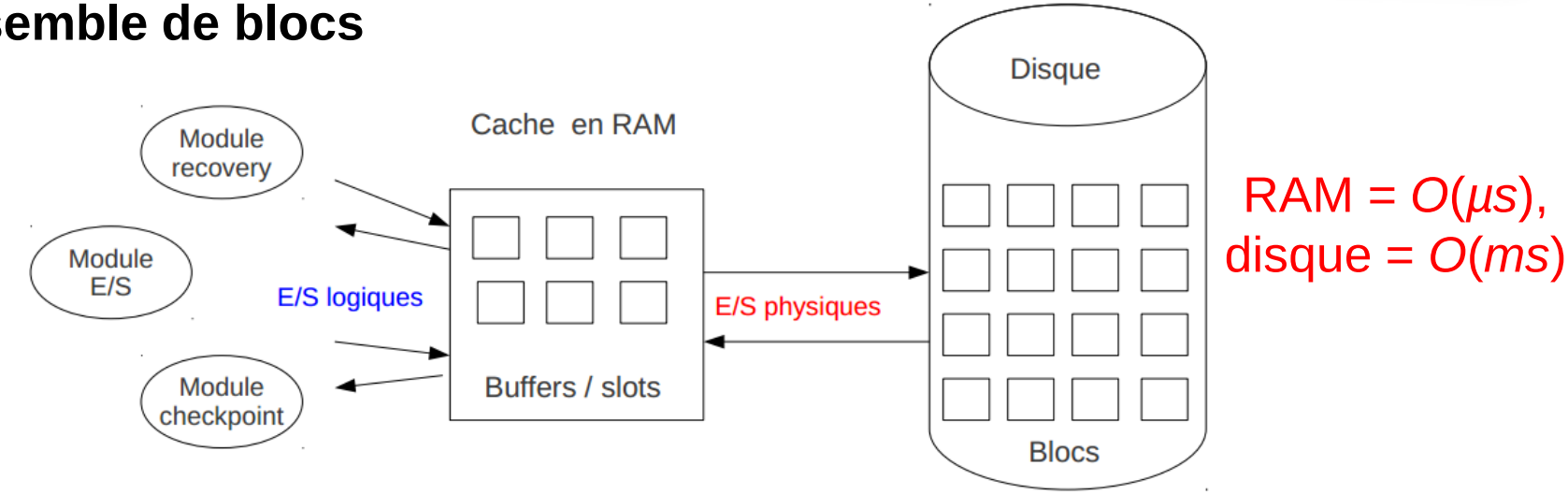
L'unité d'allocation sur le disque dur est le bloc physique.
Un bloc physique est composé de *1* à *n* secteurs
exemple :

1 bloc = 2 secteurs de 512 octets soit 1 kB.

Les opérations de lecture et d'écriture du SGF se font bloc par bloc.

Organisations physiques: Le gestionnaire du cache -

- **Fichier = ensemble de blocs**

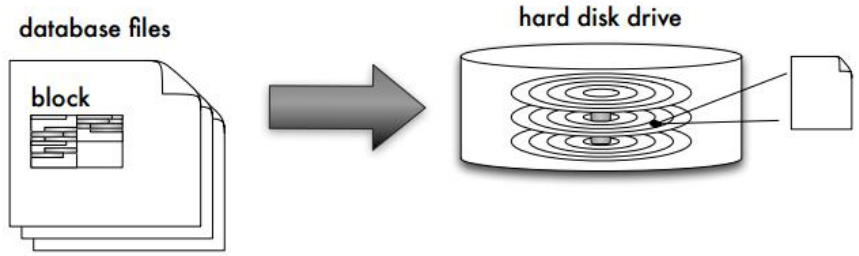


RAM = $O(\mu s)$, disque = $O(ms)$

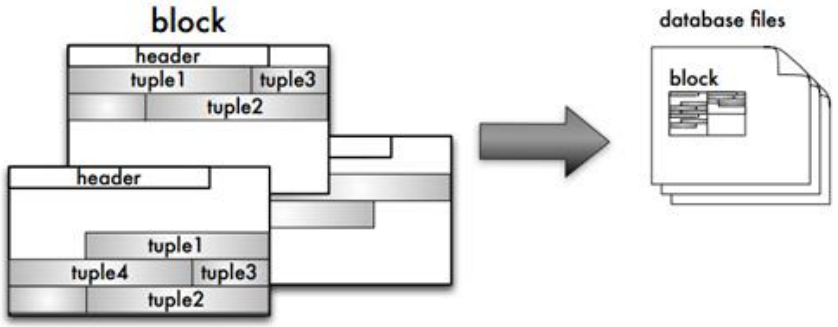
- **Chargement d'un bloc b:**
  - si le bloc b est dans le cache, retourner son numéro de slot.
  - sinon s'il y a un slot libre dans le cache, chargé le bloc du disque
  - sinon choisir un slot à vider (l'écrire sur disque) et le remplacer par b
- **Stratégies de remplacement:** LRU, FIFO, ...

Organisations physiques
Cas des données structurées

- **Fichier** = ensemble de blocs
  - ➢ chaque bloc renferme un ensemble d'enregistrements un enregistrement peut être de taille fixe ou variable (longueur de champs, nombre de champs)
- **Facteur de blockage**: Nombre enregistrement par bloc



- **Clustering:** rassembler dans le même bloc, les enreg susceptibles d'être traités ensembles.

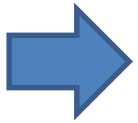


tuple identfier (TID) concept [N° Bloc , Offset]

Besoin des utilisateurs

1. Chaque utilisateur doit pouvoir créer, Supprimer, lire, écrire et modifier des fichiers
2. Chaque utilisateur peut avoir un accès contrôlé aux fichiers

… Manipulation des fichiers et des dossiers



Système de gestion de fichiers (SGF)

– Implements file abstraction ***(including APIs for file types)***
  – Implements directory structure for organizing files
  – File management *involves operations on external devices & memory*

Organisations physiques:

Le problème:

- Comment l'implémenter ?
  - Disque → partition
    Début de disque → table des partitions
    Chaque partition à un **SGF**
  - **Premier région**: gère les données
    Fichier
    Données → organisation en blocs
  - **Deuxième région** : gère les métadonnées
    Stocker dune manière persistante dans le disque
    Réserver par le SE

Organisations physiques

Section 3 : Système de gestion de fichiers (SGF)

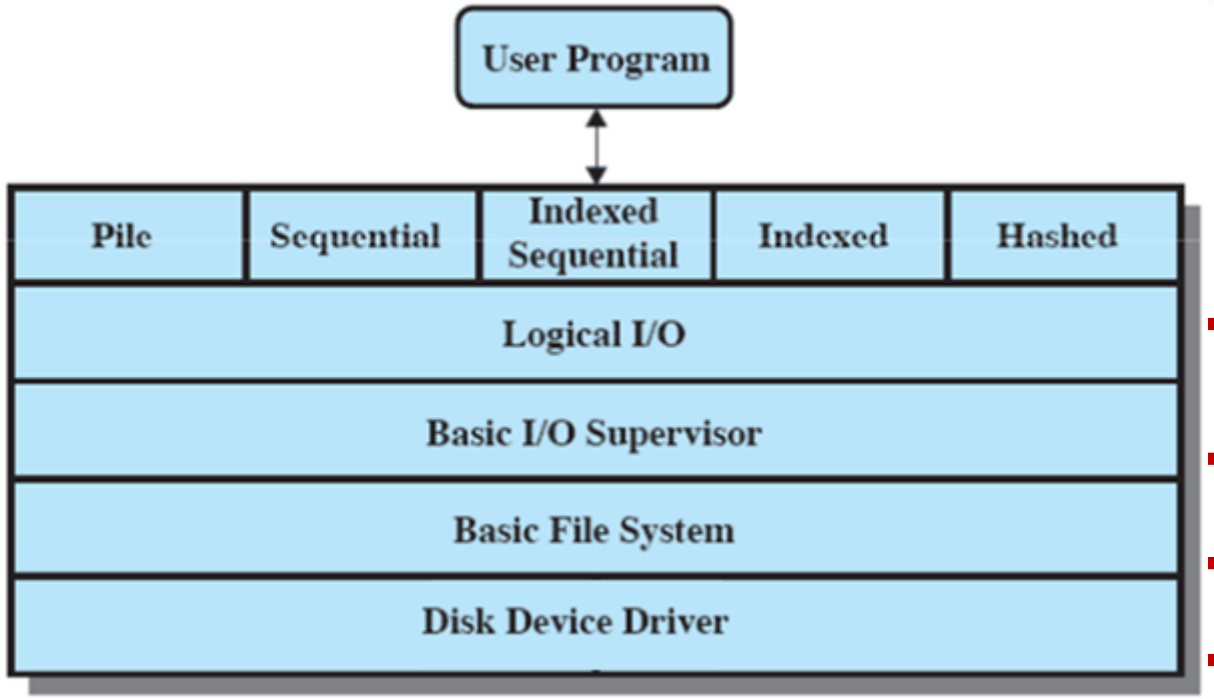
C'est quoi un SGF?

- Un Système de gestion de fichiers abrégé FS (**file system)** est une structure qui permet de stocker, d'organiser des données sur un support : Disque magnétique (HDD) , SSD, RAM, USB, DVD,…

Les tâches d'un SGF

- Interface conviviale pour manipuler les fichiers
- Le stockage des fichiers dans le disque dur
- La gestion d'espace libres sur le disque dur
- La gestions des fichiers dans un environnement muti-utilisateur
- Les données d'utilitaire pour le diagnostic, la récupération en cas d'erreurs, l'organisation des fichiers

Composantes du SGF



Mapping. < *Demandes E/S* avec *inodes , files, directories, superblocks (partitions)* >

Echange entre Buffer /Disque

Ordonnancement
Allocation du buffer

Communiquer directement
avec le matériel d'E / S

Système de gestion de fichiers (SGF)

Improving Linux System Performance with I/O Scheduler Tuning

- **Changing the I/O Scheduler**

`/sys/block/<disk device>/queue/scheduler` file.

```
# cat /sys/block/sda/queue/scheduler
noop [deadline] cfq
```

`/sys/block/<disk device>/queue/scheduler` file with the new I/O scheduler selection.

```
# echo "cfq" > /sys/block/sda/queue/scheduler
# cat /sys/block/sda/queue/scheduler
noop deadline [cfq]
```

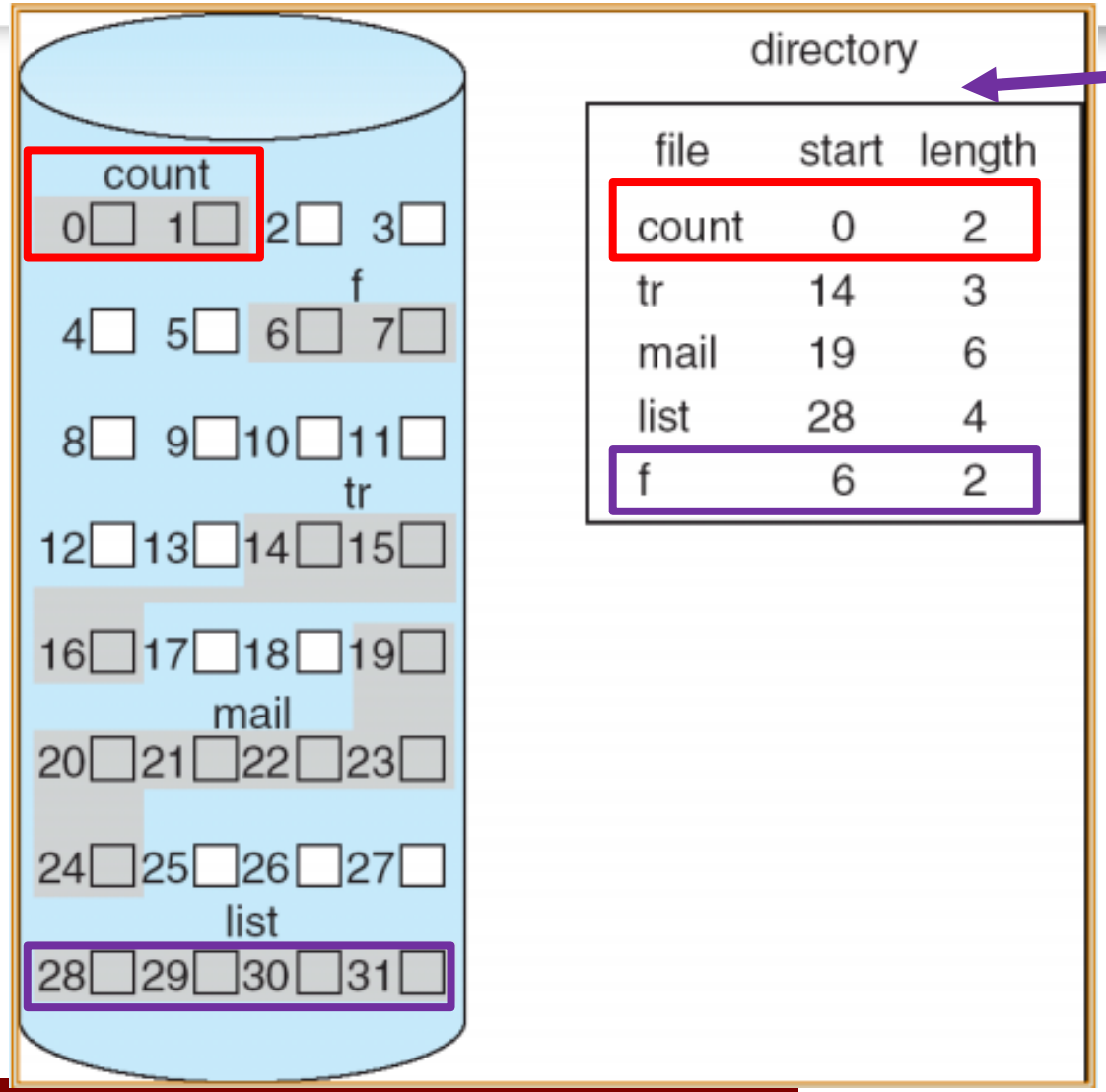
Source: https://blog.codeship.com/linux-io-scheduler-tuning/

Section 4 : Techniques d'allocation des blocs physiques

Différentes méthodes pour allouer un fichier physique

- Allocation contiguë (séquentielle simple) ;
- Allocation par blocs chaînés ;
- Allocation indexée.
- Direct ou par Hachage

Organisation - blocs contigus



Entrée de répertoire :

- adresse du premier bloc
- et longueur (en nombre de blocs)

L'entrée d'un répertoire dans un système qui applique l'allocation séquentielle a la structure suivante:

Nom du fichier	Attributs	Adresse du fichier	Longueur

Allocation contigüe
Avantages et inconvénients

Problème de création d'un fichier

- Comment gérer plusieurs fichiers qui grandissent simultanément?
- Nécessité de réserver un nombre fixe de blocs à la création.
- Combien de blocs réserver?

– Avantage : accès rapide,

– Désavantage : fragmentation du disque, nécessite de rouler un "défragmenteur" pour pouvoir trouver de l'espace pour une nouvelle création de fichier.

Si on veut créer un fichier de 7 octets, il faut déplacer au moins 2 fichiers.

Allocation contiguë
Résumé

Principe

- Fichiers stockés par blocs contigus sur le disque
- Temps de positionnement des têtes est minimal
- **Entrée de répertoire** : adresse du premier bloc et longueur (en nombre de blocs)
- **Accès direct et séquentiel facile à implémenter** : il suffit de mémoriser l'adresse du premier bloc
- **Gestion de l'espace libre** : Techniques : Fragmentation externe, Bit map, compactage etc.

Problème majeur : fichiers de taille variable

- Trop d'espace : fragmentation interne
- Pas assez d'espace : déplacement (coûteux) du fichier. Pas toujours possible.

Utilisation actuelle : CD / DVD-ROM

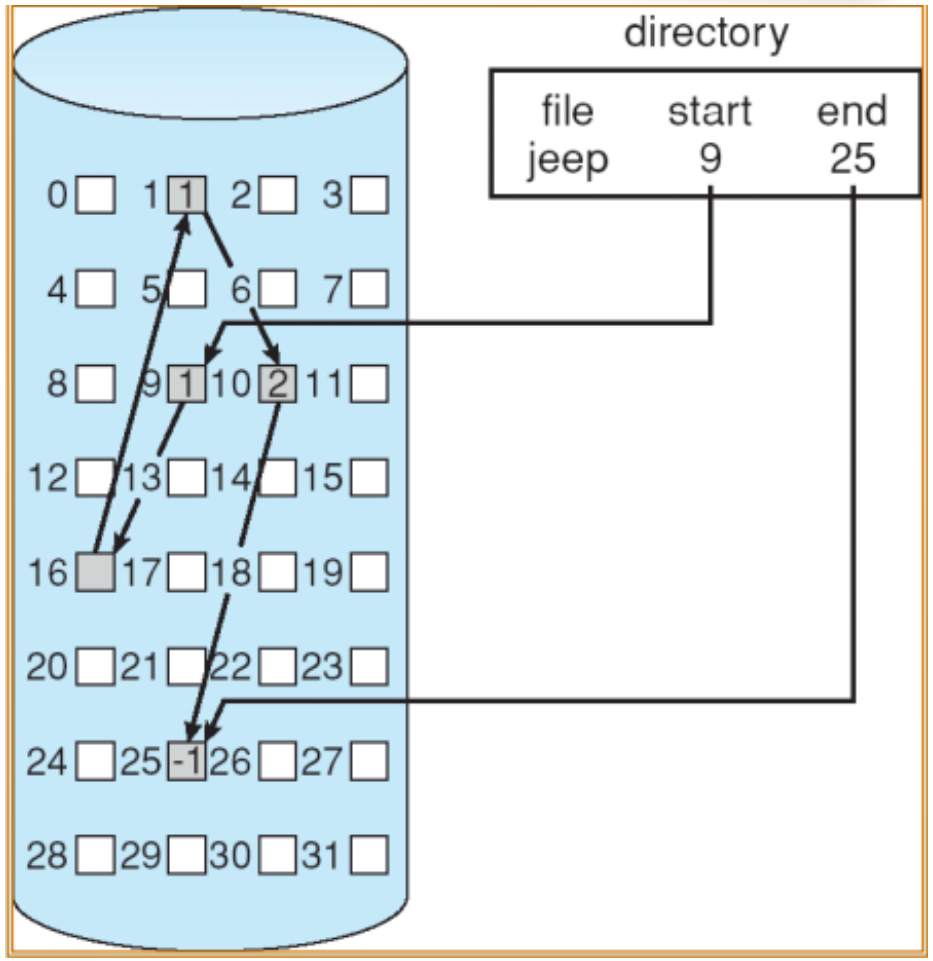
Allocation chainée

Définition

- Elle consiste à allouer des blocs chaînés entre eux aux fichiers.
- Un fichier peut désormais être éparpillé sur le disque puisque chaque bloc permet de retrouver le bloc suivant.

Principe

- Fichier = chaîne non contiguë de blocs disque
- Chaque bloc se termine par un pointeur sur le bloc suivant
- Une entrée de répertoire contient un pointeur sur le premier bloc



Allocation chainée

Avantages

Pas de fragmentation externe puisque tous les blocs peuvent être alloués

Pas de limite de taille

Inconvénients

Accès au fichier est séquentiel: On commence le parcours à partir du début du fichier même si on a récemment chargé les blocs

Fiabilité : perte de pointeur critique **->** on se retrouve dans une autre zone mémoire

Solutions : **listes doublement chaînées**, reproduction du nom de fichier et numéro de bloc dans chaque bloc etc. Coûteux dans tous les cas.

Organisation - index de blocs hiérarchisés

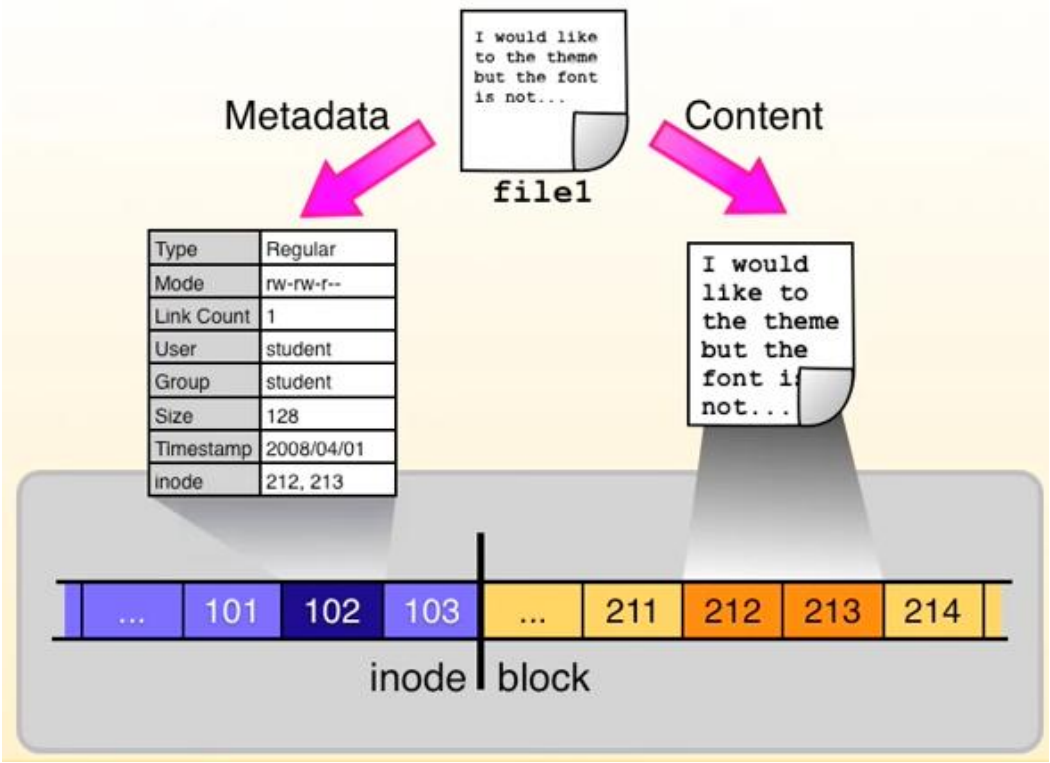
Notion de inode

L' i-node (Index NODE ou nœud d'index en Français),

Concept très ancien (année 50)

- **Appliquer dans tous les SF**
  - Inode
    - Date
    - droits d'accès
    - Taille de ficher
    - ……
    - Référencer les différents blocs (pointeurs)
- **Allouer un fichier**
    - Allouer i-node
    - Allouer un ficher

Structure d'un I-noeuds



Organisation - index de blocs hiérarchisés

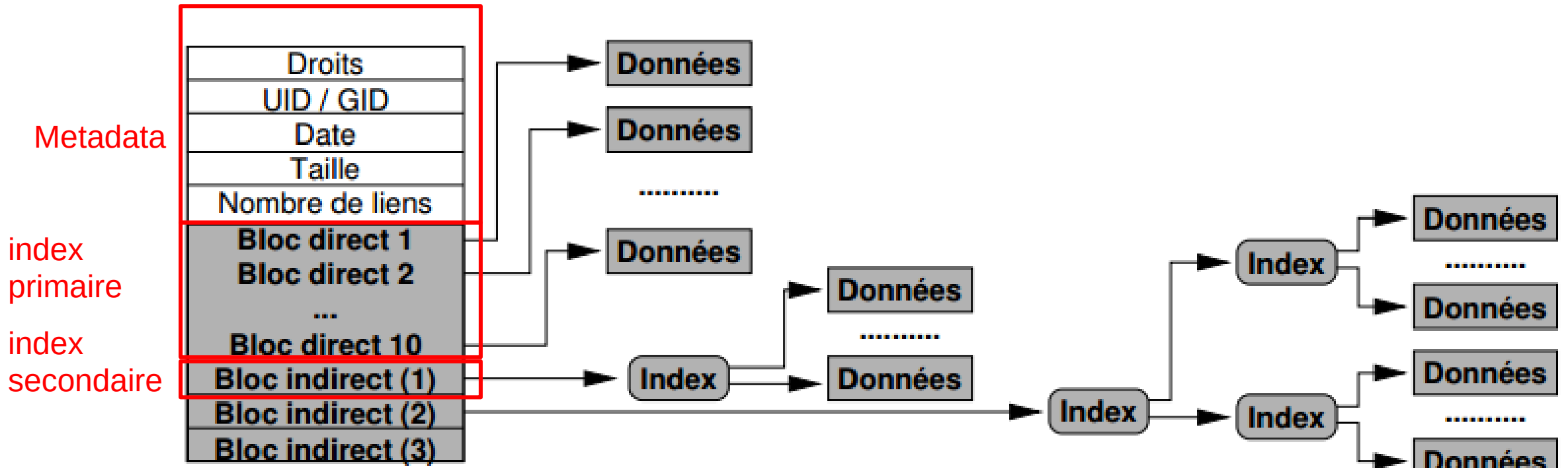
Un i-node est une structure représentant :

a) un processus
b) un espace d'adressage
c) un fichier
d) une partition
e) un système de fichiers
f) un périphérique

Structure d'un *inode* :
Metadata

– le type (le type du fichier, un entier dont la valeur est 0 pour un fichier et 1 pour un répertoire,
– Le nombre de liens (voir après),
– UID (User Identification): numéro d'utilisateur du propriétaire
– GID (Group Identification) : numéro du groupe propriétaire
– Taille du fichier en octets
– Adresses des blocs de données (qui contiennent le fichier)
– Droits du fichier
– Date du dernier accès
– date de la dernière modification
– date de création

Structure des i-node
Exemple: Unix



Question ?

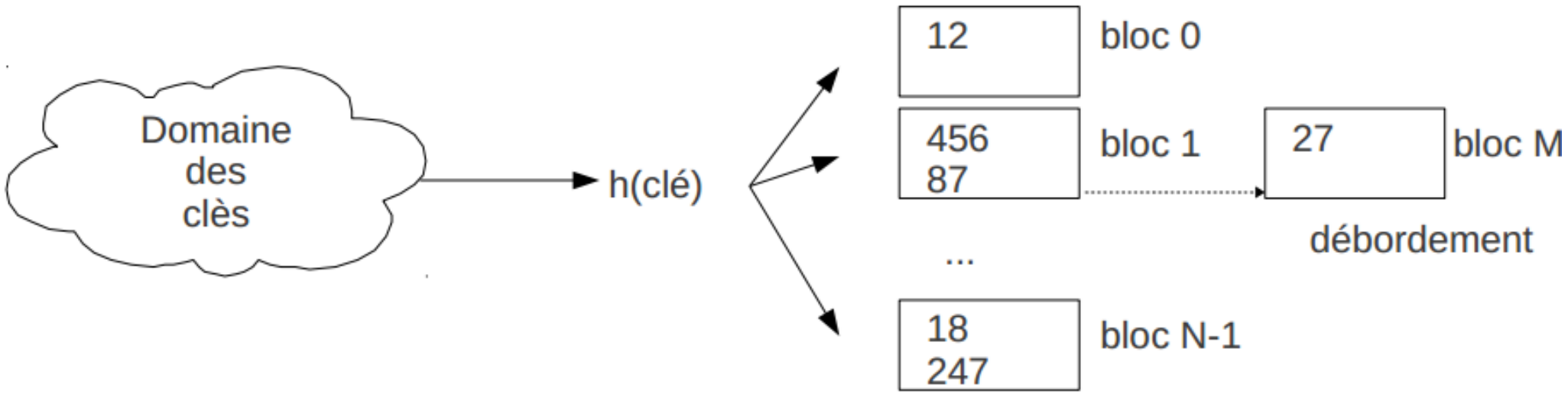
Comment implémenter les répertoires ?

On rappelle que les *inodes* sous BSD permettent d'adresser 12 blocs, 3 tables de niveau 1, 1 table de niveau 2 et 1 table de niveau 3. Chaque table a 1024 entrées, et occupe elle même 1 bloc.

☞Quelle est, en nombre de blocs, la quantité maximale des données que peut contenir un fichier ?

Organisations physiques: - Hachage -

- Utilisation d'une fonction ***h*** pour le calcul du n° de bloc



Organisations physiques: - Hachage -

On *calcule* la position d'un enregistrement d'après la clé.

Utilisation d'une fonction h pour le calcul du n° de bloc

- une *fonction de hachage* h associe des valeurs de clé à des adresses de bloc
- h doit répartir uniformément les enregistrements dans les n blocs alloués à la structure
- recherche d'un enregistrement => calcul de l'endroit où il se trouve.

hachage : Exemple (1/2)

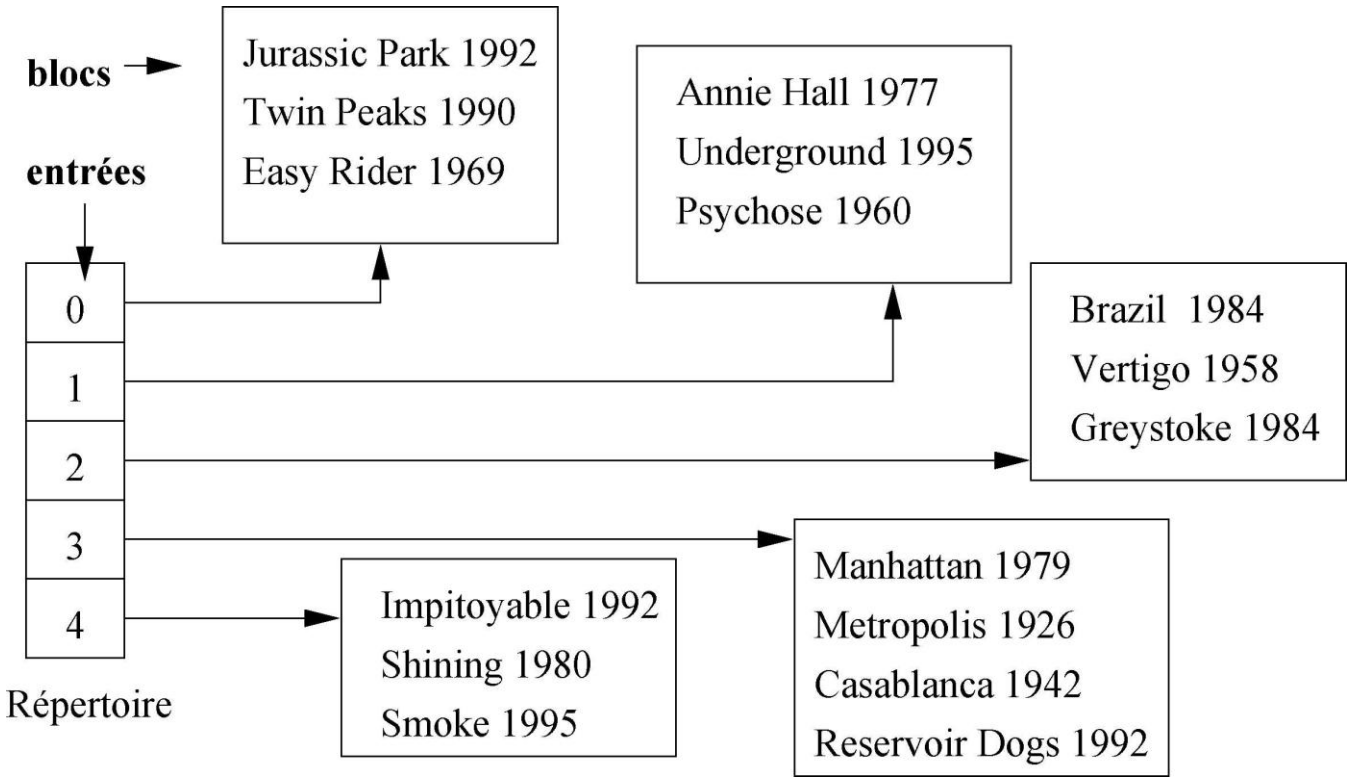
On veut créer une structure de hachage pour nos 16 films

(hyp. : 4 enregistrements par bloc)

- on alloue 5 blocs (pour garder une marge de manœuvre
- un répertoire à 5 entrées (0 à 4) pointe vers les blocs
- On définit la fonction

$$h(titre) = rang(titre[0]) \bmod 5$$

hachage : Exemple (2/2)



h(titre) = rang(titre [0]) *mod 5* *h(nouveau titre) = (18 mod 5) =3*

⇒ Problème de collision

Techniques d'allocation des blocs physiques

Section 5 : Optimisation des Performances & Sécurité

Optimisation des Performances

Optimisation des Performances

Technique des performances E/S

Objectif: Réduire les opérations E/S disque

Classique

Algorithme d'ordonnancement

- **FCFS, SSTF, SCAN**
- **C-SCAN, C-L OOK**

Moderne

- Programmer les E/S
- Buffer
- RAID
- …

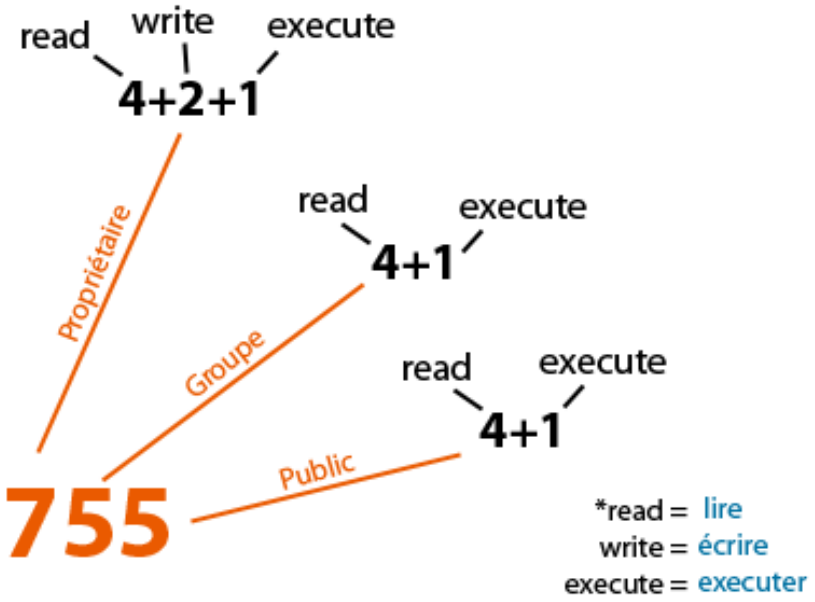
Fiabilité

Objectif : La Sécurité et l'intégrité des données

- Deux aspects:
  - Journal des transactions
    - Fichier log
    - recovery backup
  - Vérification de la cohérence

Sécurité

Octal	Decimal	Permission	Representation		
000	0(0+0+0)	No Permission	-	-	-
001	1(0+0+1)	Execute	-	-	x
010	2(0+2+0)	Write	-	w	-
011	3(0+2+1)	Write + Execute	-	w	x
100	4(4+0+0)	Read	r	-	-
101	5(4+0+1)	Read + Execute	r	-	x
110	6(4+2+0)	Read + Write	r	w	-
111	7(4+2+1)	Read + Write + Execute	r	w	x



- Difference between a Directory and File Permissions:
- The first character identifies **the file type** : dash **(-)** indicates a normal file and **d** denotes it is a directory.
- **-**rw-r--r-- **d**rwxr-xr-x

Optimisation des Performances & Sécurité

Quel ensemble de permissions permet à tous les utilisateurs d'afficher le contenu d'un répertoire?

a) rwx--x--x
b) rwxr--r—
c) rwx-w—w
d) rw-rw-rw
e) r--r--r—

Atelier – Etudes des commandes de gestion de fichiers

Commande usuelles (répértoires, fichiers) :

1-Fichiers sous Unix:

Commande	Utilisation
Chmod	Change les droits d'accès à un fichier
Cmp	comparaison de fichiers
Diff	affiche la différence entre deux fichiers
Basename	affiche le nom du fichier
File	donne le type de fichier
Find	chercher fichier
Getfalc	affiche les informations du fichier
Grep	cherche texte dans fichier
Head	affiche les premières lignes d'un fichier
Ln	création de lien vers un fichier
Mv	renomme ou déplace un fichier
Rm	supprime un fichier
Tail	affiche la fin d'un fichier
Wc	compte le nombre de lignes, de mots et de caractères dans un fichier

2- Commandes des répertoires :

Commandes	**Utilisation**
Dirname	extrait le chemin d'une commande
Dir	affiche la liste des fichiers et répertoires
Ls	affiche la liste des fichiers et répertoires
Cd	changer de répertoire
Dircmp	compare deux répertoires
Rewinddir	Revenir au début du répertoire(déplacement)
Rmdir	supprime répertoire
Mkdir	création de répertoire
Pwd	Retrouver le répertoire courant
Seekdir	permet de sauter à une position donnée
Telldir	renvoie la position courante
Scandir	sélectionne une partie du contenu d'un répertoire, la trie et fournit son contenu dans une table allouée automatiquement

Bibliographie

Bibliographie

- **LIVRES ET COURS :**

Systèmes d'exploitation et programmation système par Yves PAGNOTTE
Programmation système en C sous Linux par Christophe Blaess
Le système de fichiers Ext2 par Rémy Card ;Remy.Card@linux.org
Initiation au système Unix par D.E. Menacer
Programmation SystèmeUNIX Rev G.1, Janv 99
Document non révisable SI-220RevG.1
Guide étudiant Solaris 2.x
GUIDE du KornShell sous UNIX (HP-UX 8.0, SUNOS 5.1, AIX 3.2) par Jean François Pujol août 93
Les systemes de gestion de fichiers par Ivan Boule (2007/2008)
Introduction au système UNIX :C. Méhat -D. Mascré - M. Mayero, Département R&T,IUTdeVilletaneuse
Introductionau système UNIXGTR 2001-02 :Emmanuel Viennet ,viennet@lipn.univ-paris13.fr
A Fast File System for UNIX Marshall Kirk McKusick, William N. Joy†,Samuel J. Leffler‡, Robert S. Fabry
SGF de S. Laporte
Linux System Programming by Robert Love
TP Archi & Syst d'Exploitation Par Volatiana RALAMBONDRAINY – Samy FOUILLEUX

- **Liens internet :**

http://www.linux-france.org/article/sys/ext3fs/
http://www.freebsd.org/.
http://www.gentoo.org/doc/en/articles/afig-ct-ext3-intro.xml
http://en.wikipedia.org/wiki/Unix_File_System
http ://www-gtr.iutv.univ-paris13.fr/Cours
http://www.africacomputing.org