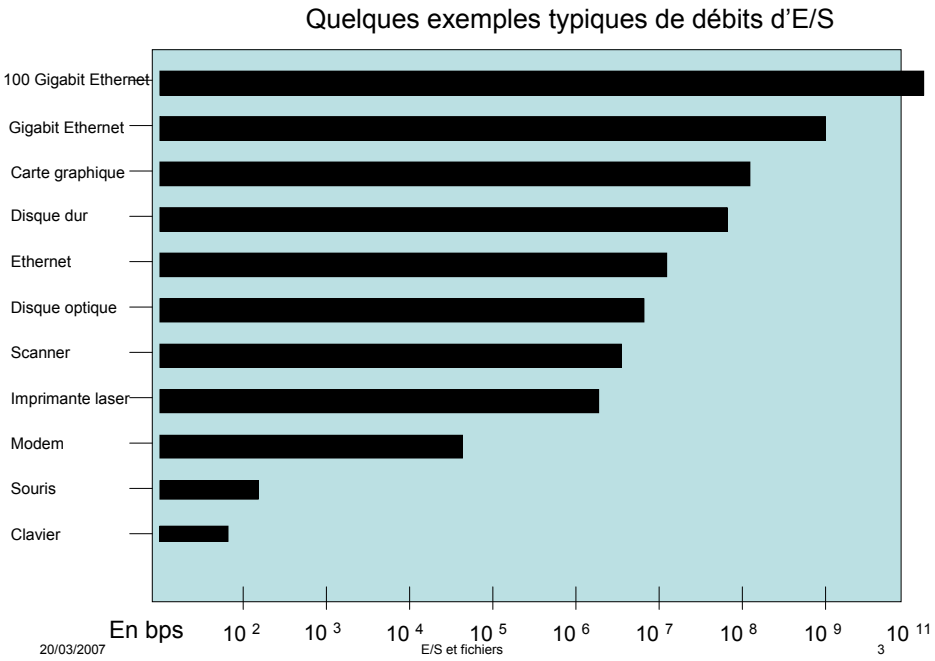
Gestion des E/S et des fichiers

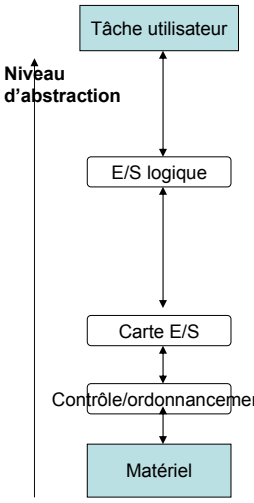
Niveau	Nom	Objets impliqués	Exemple d'opérations
13	Shell	Env utilisateur	Langage shell
12	Processus uitilisateur	Processus	Fork, quit, kill, suspend, resume
11	Répertoire	Répertoire	Mkdir, remove, attach, detach
10	Périphériques	Imprimante, écran, clavier	Open, close, read, write
9	Système de Gestion de fichiers	Fichiers	Create, destroy, open, close, read, write
8	Communications	Pipes	Create, destroy, open, close, read, write
7	Mémoire virtuelle	Segments, pages	Read, Write, Fetch
6	Mémoire de masse	Blocs de données, canaux d'E/S	Read, Write, Allocate, free
5	Processus, tâches et flots	Tâches, sémaphores, liste de tâches	Suspend, Resume, Wait, Signal
4	Interruptions	Invocation pilote (handler)	Invoke, mask, unmask, retry
3	Procédures	Appel et retour, contexte	Call, return
2	Jeu d'instructions	Pile d'évaluation, gestion des UF, microprogrammes	Load, Store, add, addf, branch, jsr, rts, rti
1	Carte et circuits	Registres, UF, bus …	Clear, activate, transfert …



Organisation des E/S

- E/S programmées
  - Le processeur émet une commande d'E/S sous le contrôle d'une tâche vers un module d'E/S ; le pilote du périphérique est activé, la tâche attend que l'opération se termine.
- Par interruption
  - Le processeur émet une commande d'E/S sous le contrôle d'une tâche vers un module d'E/S ; le pilote du périphérique est activé, la tâche poursuit l'exécution de ses instructions puis est interrompue par le contrôleur du périphérique lorsque l'E ou la S est terminée ; la tâche est ou non suspendue suivant les dépendances entre la tâche et le pilote.
- Par DMA
  - Le processeur émet une commande d'E/S sous le contrôle d'une tâche vers un DMA qui engendre une interruption vers la tâche lorsque le transfert est réalisé.

Structures logiques de la fonction d'E/S

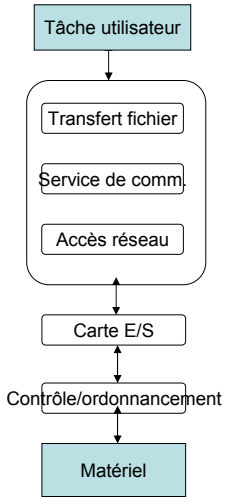


Non concerné par les détails d'accès au périphérique
Identifie le périphérique (nom, méthode d'accès)

Les opérations demandées sont transformées en séquences d'instructions d'E/S et de commandes sur les canaux. Des tampons permettent d'améliorer l'utilisation

Gestion de l'adressage, des its, de l'état, et contrôle l'échange par le pilote du périph.

Périphérique local

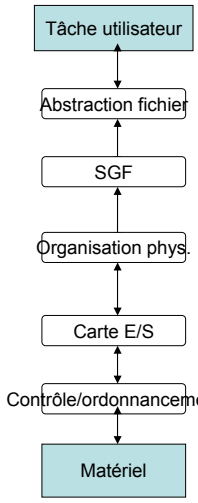


Architecture de communication
- Socket
- TCP (Transmission Control Protocol) ou UDP (user datagram protocol)
- IP (Internet Protocol)

Gestion de l'adressage et du routage

Gestion des caractéristiques du medium de transmission, la nature du signal, le débit…

Port de communication



Définition du type fichier, des attributs, des méthodes d'accès

Transformation nom- logique – descripteur ; contrôle des accès

Fournit une adresse de blocs ; gère les secteurs du disque

Effectue les opérations de positionnement, de lecture ou d'écriture : commande de l'échange, gestion des dispositifs matériels

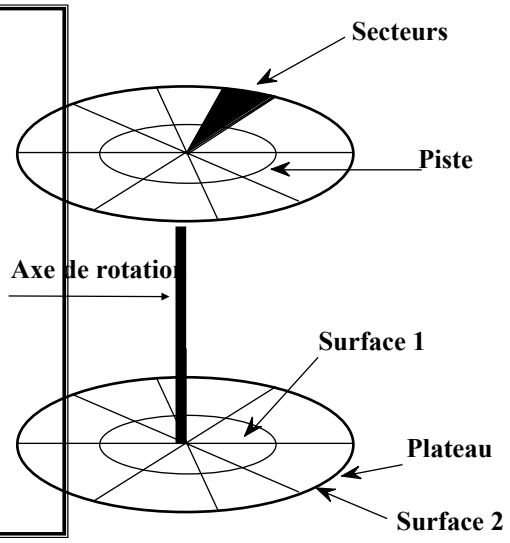
Commande des têtes sur les disques, établissements des signaux

Périphériques de stockage

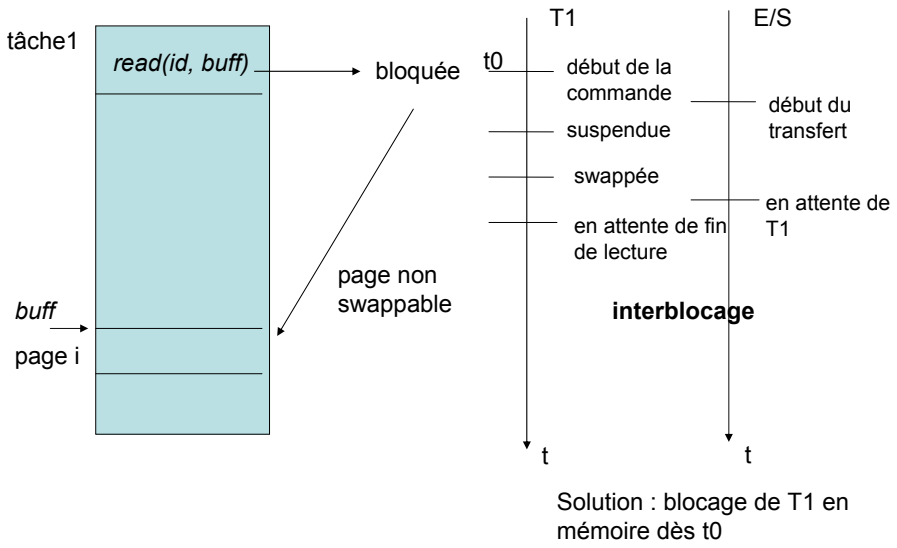
Le disque

Un exemple de disque :

- Diamètre 5.25 inch.
- Capacité 36 GO
- Cylindres 18 479
- Pistes/cylindre 8
- Secteurs/piste 485
- Octets/secteur 512
- Vitesse de rotation (T/mn) 15000
- Recherche moyenne en mms 3,6
  - minimale 0,3
  - maximale 22.5
- Débit 500..700 Mo/s



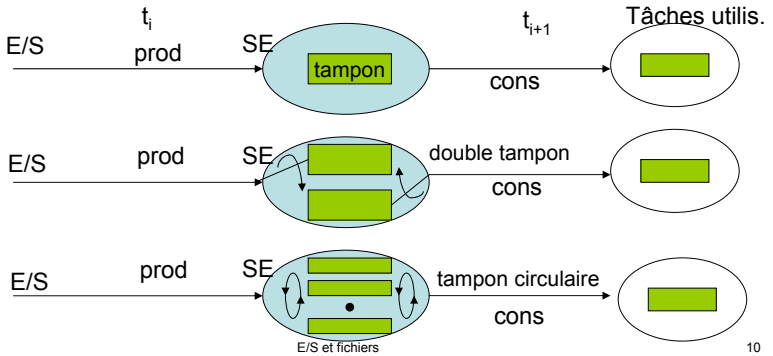
Tampons d'E/S



Solution : blocage de T1 en mémoire dès t0

Différents schémas de tampons

- E/S orientées blocs et E/S orientées flots
  - Bloc type disque ou autre référencé par un n°.
  - Flots = succession d'octets (terminaux, ports de com).



Ordonnancement des accès disque : Etude de performance

- Temps de recherche moyen + délai de rotation moyen + temps de transfert = temps d'accès moyen

Temps de recherche moyen (Ts) = valeur fonction de la technologie (< 10 ms)
Délai de rotation = ½ du temps total de révolution (r)
Temps de transfert :

$$T_t = \frac{b}{r * N}$$

Nombre d'octets à transférer
Nombre d'octets sur la piste
Délai de rotation
Temps de transfert

Temps d'accès moyen total = $Ts + 1/2r + T_t$

Exemple

- Un seul disque ainsi défini :
  - Temps de recherche moyen 4 ms
  - Vitesse de rotation 15 000 t/mn (250 t/s)
  - 500 secteurs par piste
  - 512 octets/secteurs
- Lecture d'un fichier de 2500 secteurs représentant un total de 1,28 Moctets.

  Cas N° 1 : le fichier est stocké séquentiellement sur 5 pistes adjacentes (5 pistes * 500 secteurs/pistes) = 2500 secteurs

Lecture de la première piste :

Temps de recherche moyen	**= 4 ms**
Délai de rotation	**= 4 ms**
Lecture des 500 secteurs	**= 4 ms**
	total = 12 ms

Lecture des pistes successives :

Délai de rotation = 4 ms
Lecture des 500 secteurs = 4 ms
total = 8 ms

Lecture du fichier en entier : 12 + 4 * 8 ms = 0,044 s

– Cas N° 2 : le fichier est stocké aléatoirement
  • Lecture de chaque secteur :
    » **Temps de recherche moyen = 4 ms**
    » **délai de rotation = 4 ms**
    » **lecture d'un secteur = 0,008 ms**

TOTAL = 8,008 ms

Lecture du fichier en entier : 500 * 8,008 ms = 4,004 s.

Conclusion : grande importance de l'ordre dans lequel les secteurs sont lus sur le disque

Algorithmes d'ordonnancement des accès disque

- Réduction du temps de positionnement
- Algorithmes
  – FIFO
  – SSTF (Shortest service time first) :minimise les déplacements du bras en ordonnant les n° de blocs. Déplacements + ou –
  – SCAN : le bras se déplace dans une seule direction
  – FSCAN : précédente sur deux files.
- RAID 0
  – Parallélisation des accès en distribuant les accès séquentiels sur plusieurs disques.

Fichiers et système de gestion

- Organisation et accès
- Répertoires
- Partage de fichiers
- Accès partagés
- Gestion des espaces
- Exemples : UNIX et Windows

Interface utilisateur

fd = open (chemin d'accès, type) ;

L, Lecture
E, Ecriture
A Ajout

La valeur de fd sera ensuite utilisée pour référencer ce fichier : cette valeur permet d'accéder à un descripteur qui contiendra toutes les info associées au fichier

fd = creat (chemin d'accès, mode) ;

Permissions :
RWE

Création d'un nouveau fichier, d'une entrée dans la table des descripteurs de fichiers, création d'une entrée dans le répertoire, initialisation de la structure de données « fichier »

close (fd) ;

Suppression de l' entrée dans la table des descripteurs de fichiers pour la tâche en cours

number = write (fd, buffer,nombre) ;

fd : référence du fichier
buffer : adresse mémoire de la zone de données à écrire
nombre : nombre d'octets à écrire
number : nombre d'octets effectivement écrits
Si (number != nombre) alors erreur

number = read (fd, buffer,nombre) ;

fd : référence du fichier
buffer : adresse mémoire de la zone de données à lire
nombre : nombre d'octets à lire
number = 0 alors end_of_file sinon si -1 erreur sinon pas d'erreur

Programme qui recopie l'entrée standard sur la sortie standard dans UNIX

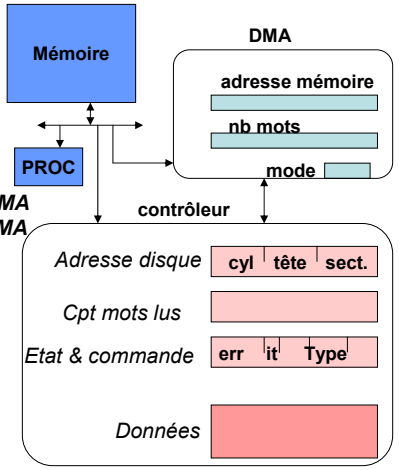
```
char buf [8192] ;
while (( n = read (0, buf, sizeof(buf)) > 0) write (1, buf, n) ;
```


Exemple : mise en œuvre d'une écriture

number = write (fd, buffer,nombre) ;

En utilisant fd, recherche de l'entrée dans un répertoire qui fournit les informations pour charger le Registre Adresse Disque (cyl, tête, secteur)

```
MOVE    #buffer, R1
MOVE    nombre, R2
SWI     DMAdisque
```

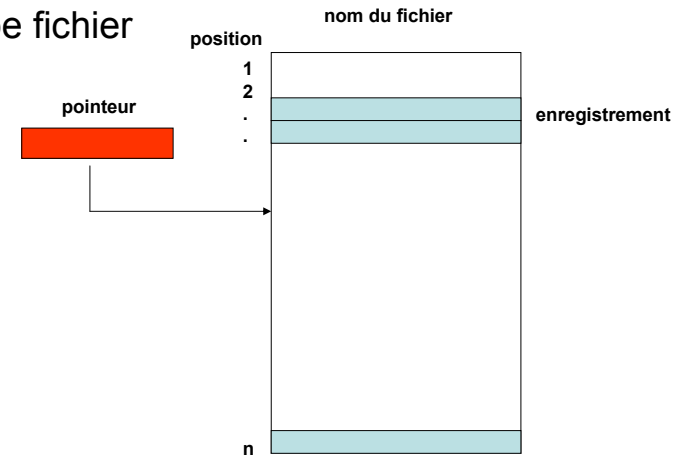


DMAdisque : Transfert R1 -> Registre adresse du DMA
Transfert R2 -> Registre nb mots du DMA
Mode = 0 (+ sur adr, - sur nb)
Lancer le transfert
fin

itDMAdisque : Lecture compte_de_mots -> R0
R0 ->number
RTI

Type fichier et système de gestion des fichiers

- Le type fichier



Méthodes d'accès

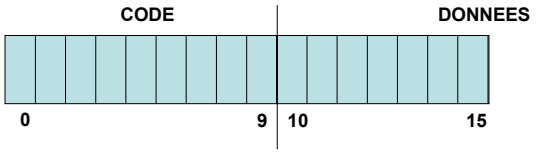
- Direct
  - Le contenu de pointeur est géré par le programmeur : les instructions d'accès (read, write, position) contiennent explicitement une valeur de position.
- Séquentiel
  - La gestion du pointeur est implicite : l'accès s'opère à partir de la valeur de position qu'il contient ; position vaut 0 au début et chaque nouvelle exécution d'une instruction d'accès fait +1 sur la valeur du pointeur.
- Séquentiel indexé
  - Le pointeur désigne un fichier d'index qui fournira un point d'entrée sur le fichier de données.

Implantation

- Accès direct
  - descipteur-fichier = open (nom_de_fichier) ; // création du pointeur
  - read (descripteur_fichier, adresse de composition, position) ;
  - write (descripteur_fichier, adresse de rangement, position) ;
- Accès séquentiel
  - rewind (descripteur_fichier) ; // pointeur <- 0 ;
  - reads (descripteur_fichier, adresse de composition) ;
  - writes (descripteur_fichier, adresse de rangement) ;
- Accès séquentiel indexé
  - read (descripteur_fichier, adresse de composition, clé) ;
  - write (descripteur_fichier, adresse de rangement, clé) ;
  - close(descripteur_fichier) ; // suppression du pointeur et de l'entrée dans la table des descripteurs de fichiers.

Exemple

- Soit un fichier de 125 000 enregistrements :



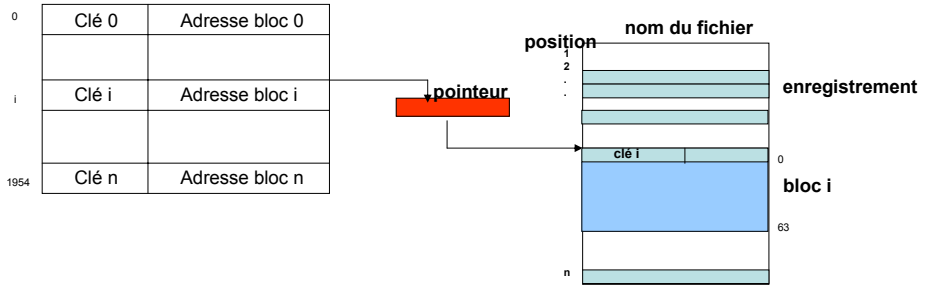
La taille totale du fichier est de 2 millions d'octets ; pb : recherche d'un élément à partir de son code :

Accès séquentiel : position en début de fichier ; lecture et comparaison des valeurs des codes lus et comparaison avec le code recherché. Complexité algorithmique de l'opération ?

optimisation : maintien d'un ordre (croissant ou décroissant) sur les codes, puis recherche dichotomique. Complexité algorithmique de l'opération ?

<u>Accès direct</u> : Calcul de la valeur de l'adresse physique (n° de secteur) à partir de la valeur de position et de la taille du fichier.
Ex : où se situe l'enregistrement n° 520 000 ?
8192 octets/secteurs => 512 enregistrements/secteurs => n° secteur = sup(520000/512) = 1016.
Les éléments sont rangés dans une topologie connue qui permet d'exécuter un algorithme pour trouver directement l'adresse d'implantation. (généralement ordre croissant sur les codes)

<u>Accès séquentiel indexé</u> : accès direct sur un fichier d'index, accès séquentiel sur le bloc identifié par l'index.



Si 64 enregistrements/bloc , le fichier de données contient 125 000 / 64 = 1954 index.

Contenu des fichiers répertoire

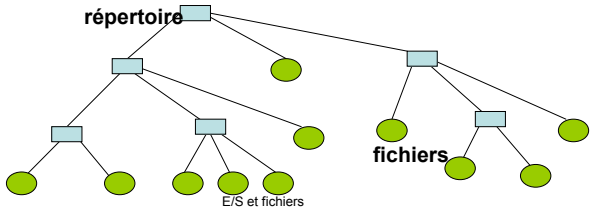
- Nom
- Type
- Organisation
- Volume
- Adresse d'implantation (cylindre, piste, n° de bloc)
- Taille courante
- Taille max.
- Id du propriétaire
- Information d'accès
- Droits d'accès

Contenu des fichiers répertoire (2)

- Date de création
- Id du créateur
- Date du dernier accès
- Identité du dernier accès
- Dernière date de modification
- Identité du dernier en écriture
- Date de la dernière sauvagarde
- Informations sur l'utilisation courante du fichier

Opérations

- Recherche de fichiers
- Création de fichiers – création d'une entrée dans le répertoire
- Suppression du fichier – suppression d'une entrée
- Lister le contenu du répertoire
- Mise à jour du répertoire lors d'un accès à un fichier



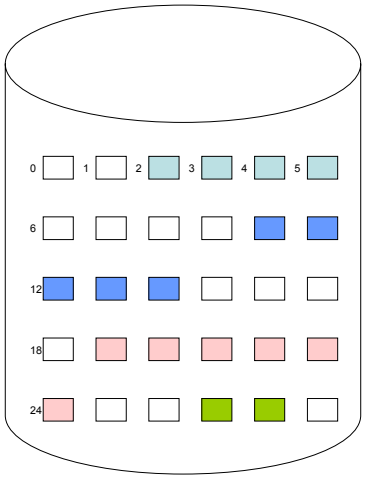
Accès partagés aux fichiers

- Verrous sur fichiers permettent à une tâche de verrouiller un fichier et de protéger les accès concurrents d'autres tâches
  - Un verrou partagé peut être accédé concurremment par plusieurs tâches (accès en lecture)
  - Un verrou exclusif permet à une tâche et une seule d'accéder à un fichier.
- Les SE fournissent soit des mécanismes directifs (Windows) soit consultatif (Unix)
  - directif : une tâche acquiert un verrou exclusif et empêche toutes les autres d'accéder au verrou
  - consultatif : le SE n'interdit rien ; c'est à l'utilisateur de s'assurer que les verrous sont correctement utilisés dans son application.

Méthode d'allocation des fichiers

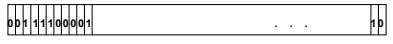
- Position du problème
  - A la création d'un fichier, l'espace nécessaire doit-il être créé complètement ?
    - Préallocation statique lorsque la taille du fichier est connue ou estimé en maximum
    - Allocation dynamique par bloc au fur et à mesure des besoins
  - Quelle est la taille du bloc logique constituant la granularité minimale d'accès sur le disque ?
    - Du bloc à tout le fichier
      - Séquentialité améliore la performance
      - Un grand nombre de blocs de faible taille augmente la taille de la table
      - Des blocs de taille fixes simplifient la gestion de l'allocation
      - Des blocs de tailles faibles ou variables minimisent la place perdue
  - Comment gérer les blocs de données alloués pour un fichier ?
    - Contigüe
    - Chainée
    - Indexée
  - Comment gérer l'espace libre ?

Allocation contigüe



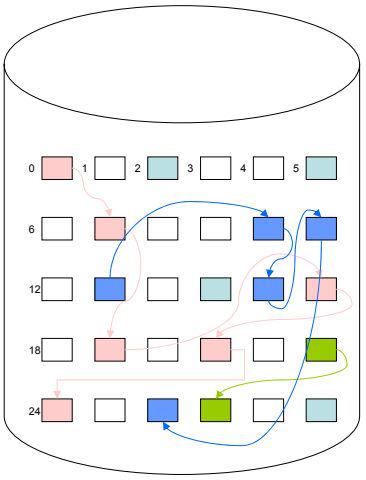
Nom de fichier	premier bloc	longueur
Fichier A	2	4
Fichier B	10	5
Fichier C	19	6
Fichier D	27	2

Vecteur de bits modélisant l'occupation des secteurs



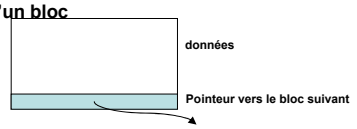
Pb. Trouver l'espace libre pour la création d'un fichier, fragmentation du disque, nécessité d'une compaction périodique

Allocation chaînée



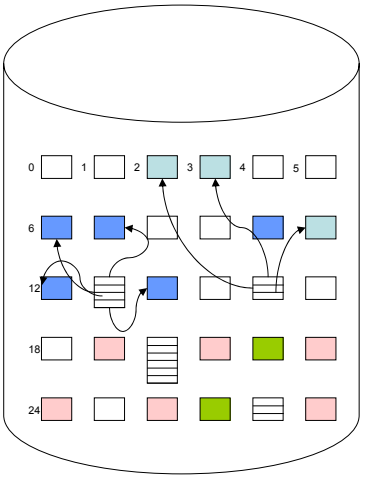
Nom de fichier	premier bloc	longueur
Fichier A	2	4
Fichier B	13	5
Fichier C	0	6
Fichier D	23	2

Structure d'un bloc



Pb. Perte d'efficacité, car absence de localité ; nécessité de réorganiser les blocs pour introduire de la localité. Accès séquentiels

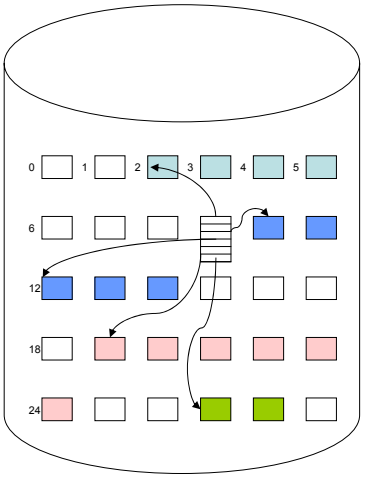
Allocation indexée (1)



Un fichier d'index séparé par fichier : résout le pb de la fragmentation du disque mais pas le pb de la localité

Nom de fichier	index bloc
Fichier A	16
Fichier B	13
Fichier C	20
Fichier D	28

Allocation indexée (2)



avec des blocs de taille variable :

Nom de fichier	index bloc	adresse de début	longueur
Fichier A	16	2	3
Fichier B	13	6	5
Fichier C	20	19	6
Fichier D	28	22	2

Rajoute de la localité en résolvant les autres pbs.
Peu efficace pour des petits fichiers.

Exemple : UNIX

- Unix distingue 6 types de fichiers
  - Ordinaire

  Contient des données dans 0 ou n blocs de données ; pas de structure d'enregistrements mais une séquence d'octets terminée par EOF
  - Répertoire

  Noms de fichiers et un pointeurs vers un i-node (index node) qui décrit le fichier ; le répertoire est organisé en arbre n-aire.
  - Spécial

  Associé un périphérique pour que celui-ci puisse être géré comme un fichier.
  - « pipes » nommés

  Permet la communication entre tâches ; gestion FIFO du fichier associé
  - Liens

  Un nouveau nom pour un fichier existant, pouvant être accédé différemment
  - Liens symbolique

  Fichier qui contient le nom du fichier auquel il est lié.

Les i-nodes

- Tous les fichiers UNIX sont administrés par le système par l'intermédiaire d'une structure de 64 octets, l'i-node.

Le chemin d'accès au trouve se trouve dans le fichier répertoire.
L'inode contient :

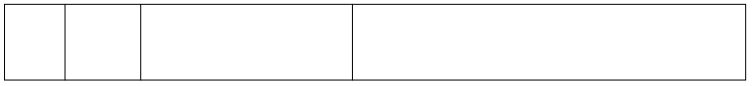
- **le nom du propriétaire du fichier**
- **l'identification du groupe associé à ce fichier**
- **type (d,f, c ou b, « pipe »)**
- **permission d'accès**
  - **propriétaire**
  - **groupe**
  - **autres**
- **nombre de références (de noms différents) sur cet i-node**
- **la date de la dernière modification du fichier**
- **la date du dernier accès**
- **la date de la dernière modification de l'inode**
- **la liste des adresses disque des blocs de données alloués en discontinuité (39 octets : 13 mots)**
- **la taille du fichier.**

La copie mémoire d'un i-node lorsqu'un fichier est ouvert contient en outre :

- **état (bloqué si un accès en écriture sur le fichier en cours)**
- **la liste des tâches en attente de déblocage**
- **le n° de l'i-node dans la liste des i-nodes sur disque**
- **le nb de références faites sur le fichier (nb d'opens)**
- **pointeurs vers d'autres i-nodes**
- **Le n° logique du SGF qui contient ce fichier**

Modifier le contenu d'un fichier => modification du contenu de l'i-node, mais une modification de l'i-node peut intervenir sans modification du fichier (création, accès en lecture, création de liens, changement de droits..)

Un file system UNIX réside sur un disque logique et contient les infos suivantes :



boot | super bloc | liste des i-nodes | blocs de données

Code pour booter le système

Taille totale du SGF
Nb max de fichiers
Début de l'espace libre pour les données

Les répertoires sont des fichiers -> représenté par un i-node -> désigne un bloc de données

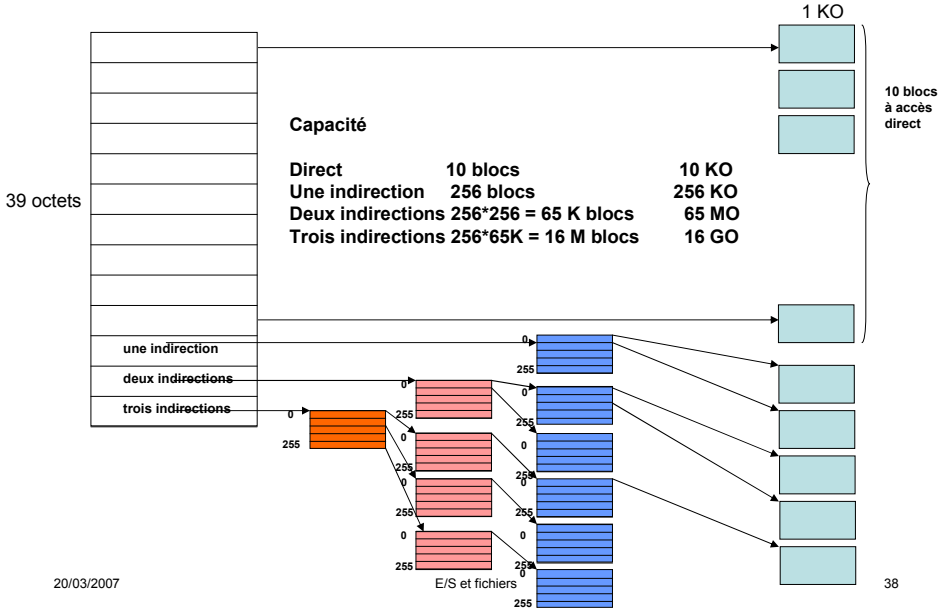
/etc

nom de fichier , n° d'inode

nom de fichier	n° inode	
.	83	0
. .	2	16
fsk	1276	32
clri	85	
motd	1268	
mount	1799	
mknod	88	
passwd	2114	
unmount	1771	128

déplacement en octet dans le boc de données

Accès à un bloc de données



Structures de données du noyau

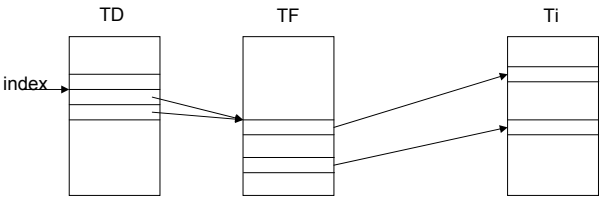


Table des descripteurs de fichiers locales à une tâche. Chaque fichier ouvert fournit une entrée

Table des fichiers ouverts gère les droits d'accès : structure globale

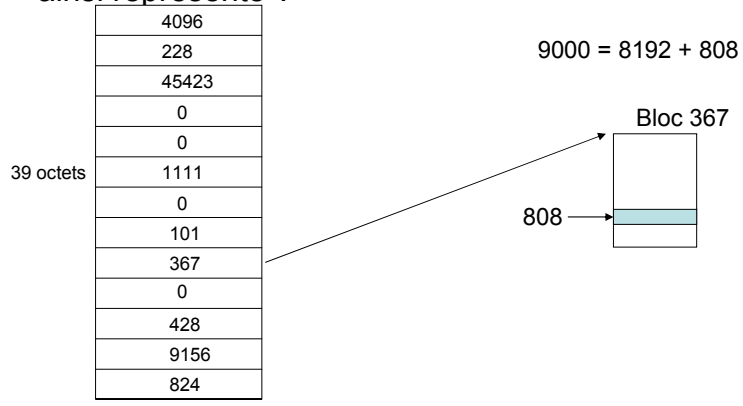
Table des n° d'i-nodes

Sur un create ou open il y a :
- **allocation d'une entrée dans chacune des tables**
- **affectation de la valeur d'index retourné en résultat de la fonction**

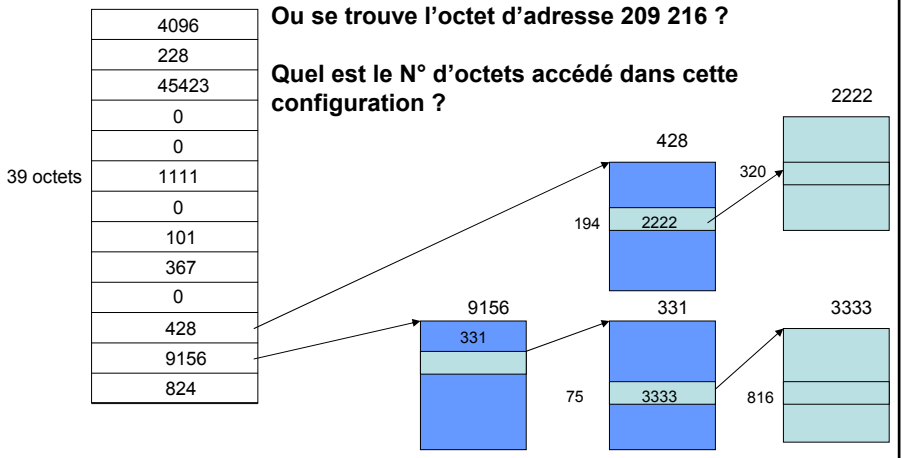
Sur un read ou write il y a :
- **accès à TD avec index**
- **accès à TF et vérification des droits**
- **accès à ti et détermination de l'adresse du bloc de données**

Exemples

- Où se trouve l'octet de position 9000 d'un fichier UNIX ainsi représenté ?



Suite exemple



Windows (NTFS)

- Secteur
  - L'unité minimale sur le disque (512 octets)
- Cluster
  - Un ou plusieurs secteurs contiguës (sur la même piste) (**2)
- Volume (max 2**64 octets)
  - Une partition logique sur le disque, constituée de un ou plusieurs clusters et utilisé par le SGF pour allouer de l'espace aux fichiers.
  - Un volume consiste en :
    - Un système de fichiers
    - Une collection de fichiers
    - Un espace libre.

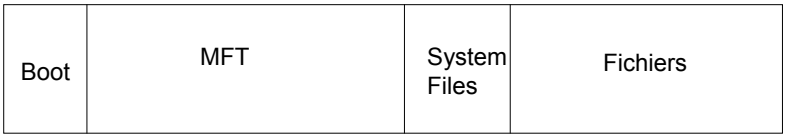
C'est une partie d'un seul disque ou il peut être déployé sur plusieurs disques

- L'utilisation de clusters pour l'allocation rend NTF indépendant de la taille des secteurs.
  - Exemple : secteur 512 octets , deux secteurs par clusters.
    - Création d'un fichier de 1600 octets => 2 clusters (2Ko)
    - Rajout de 1600 octets => 2 clusters supplémentaires soit 8 secteurs pour stocker 3200 octets
- La taille du cluster utilisé pour un volume particulier est établie au formattage du disque

Partitions NTFS et taille des clusters

Taille du volume	Nb secteurs/clusters	Taille clusters
<= 512 octets	**1**	**512 octets**
512 à 1 G octets	**2**	**1K**
1 à 2 G octets	**4**	**2K**
2 à 4 G octets	**8**	**4K**
4 à 8 G octets	**16**	**8K**
8 à 16 G octets	**32**	**16K**
16 à 32 G octets	**64**	**32K**
> 32 G octets	**128**	**64K**

Représentation d'un volume NTFS



Boot : (1 à 16 secteurs)

Master File Table : contient toutes les informations sur les fichiers et répertoires de ce volume, ainsi que la description de l'espace libre.

System files : 1 MO ;

MFT2 : un miroir des 3 premières lignes de MFT pour redondance
Fichier de log : une liste de transactions pour récupérer le File System
Cluster bit map : topologie des clusters en service
Table de définition d'attributs : définis les types supportés par ce volume

MFT : descripteurs de fichiers

Table d'enregistrements de taille variable ; une entrée par fichier ou répertoire

- Information standard
  - Attributs d'accès (read-only, read/write, etc…) ; dates (création, modif, accès, nb liens)
- Liste d'attributs : Utilisé lorsque tous les attributs ne contiennent pas dans une seule entrée.
- Nom de fichier
- Descripteur de sécurité : propriétaire et droits d'accès
- Données : contenu du fichier
- Racine de l'index : utilisé pour les répertoires
- Allocation de l'index : utilisé pour les répertoires
- Information sur le volume : version et nom du volume
- Bitmap : enregistrements utilisés de la MFT

Tolérance au fautes : principes du RAID

- La plupart des systèmes utilisent RAID 1, 3 ou 5
- Nombreux paramètres de réglages (taille des tranches…)
- Problèmes de performances dans certaines configurations