UNIVERSITE DE GHARDAIA

DEPARTEMENT DES MATHEMATIQUES ET INFORMATIQUE

COURS D'INFORMATIQUE

SYSTEME D'EXPLOITATION 1

EXERCICES ET ENONCE DE TP

2EME ANNEE INFORMATIQUE

Continuité du cours : Système d'exploitation I

La mémoire virtuelle

Algorithmes de remplacement de page

Suite à un défaut de page (absence de la page en mémoire vive), le système d'exploitation doit ramener en mémoire vive la page référencée à partir du disque (la mémoire virtuelle).

S'il n'y a pas de cadres libres en mémoire vive, il doit retirer une page (la page victime) pour la remplacer par celle référencée.

Si la page à retirer (page victime) a été modifiée depuis son chargement en mémoire vive, il faut la réécrire sur le disque (mémoire virtuelle).

Quelle est la page à retirer ?

Le choix de la page à remplacer peut se limiter aux pages du processus qui ont provoqué le défaut de page (si la stratégie d'allocation est locale)

Le choix de la page à remplacer peut s'étendre à l'ensemble des pages en mémoire (si la stratégie d'allocation est globale).

En général, l'allocation globale produit de meilleurs résultats que l'allocation locale.

Plusieurs algorithmes de remplacement de page victime ont été proposés.

-Ces algorithmes mémorisent les références passées aux pages.

-Le choix de la page à retirer dépend des références passées.

1-L'algorithme de remplacement FIFO

Cet Algorithme mémorise dans une file de type FIFO (premier entré, premier sorti) les pages présentes en mémoire.

Lorsqu'un défaut de page se produit, il retire la plus ancienne (tête de la file).

Exemple :

La mémoire contient 3 cadres (m),

La suite de références(w) ={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1}

L'algorithme produit 15 défauts de page, comme le montre la figure suivante.

m \ w	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0	0	0	0	0	**7**	7	7
1		**0**	0	0	0	**3**	3	3	**2**	2	2	2	2	**1**	1	1	1	1	**0**	0
2			**1**	1	1	1	**0**	0	0	**3**	3	3	3	3	**2**	2	2	2	2	**1**

L'algorithme est rarement utilisé car il provoque beaucoup de défauts de page.

2-L'algorithme de remplacement de la page la moins récemment utilisée (LRU Least Recently Used)

Cet algorithme mémorise dans une liste chaînée toutes les pages en mémoire. La page la plus utilisée est en tête de liste et la moins utilisée est en queue. Lorsqu'un défaut de page se produit, la page la moins utilisée est retirée. Pour minimiser la recherche et la modification de la liste chaînée, ces opérations peuvent être réalisées par le matériel. Cet algorithme est coûteux.

Exemple :

La mémoire contient 3 cadres (m),

La suite de références (w) ={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1}

L'algorithme produit 12 défauts de page comme le montre la figure suivante.

m \ ω	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	**7**	7	7	**2**	2	2	2	**4**	4	4	**0**	0	0	1	1	1	1	**1**	1	1
1		**0**	0	0	0	0	0	**0**	0	**3**	3	3	3	3	3	**0**	0	0	0	0
2			**1**	1	1	**3**	3	3	**2**	2	2	2	2	2	2	2	2	**7**	7	7

Le problème avec cet algorithme est la difficulté d'implémentation, qui requiert un support du hardware. Il faut une manière de mémoriser le temps à chaque fois qu'une page est référencée.

On peut aussi utiliser une technique de vieillissement, où un registre de n bits est associé à chaque page. Le bit le plus significatif est mis à 1 chaque fois que la page est référencée. Régulièrement, on décale vers la droite les bits de ce registre. Lorsque qu'on doit expulser une page, on choisit celle dont la valeur est la plus petite. On pourrait aussi mettre une page au-dessus d'une pile chaque fois qu'elle est référencée. On expulsera la page qui se trouve au fond de la pile.

3-L'algorithme de remplacement de la page non récemment utilisée (NRU Not Recently Used)

A chaque page est associé deux bits (R, M).

R : Positionnée chaque fois que la page est référencée (soit en lecture ou en écriture).

M : Positionnée chaque fois que la page est Modifiée.

Au lancement d'un processus, le système d'exploitation met à Zéro les bits R et M de toutes les pages.

-Périodiquement à chaque top horloge, le bit R et remis à Zéro pour différencier les pages qui n'ont pas été récemment référencées des autres pages.

Lors d'un défaut de page,

Le système retire une page avec hasard dont la valeurs RM est plus petite.

RM = 00 (Non référencée, non modifiée)

RM =01 (Non référencée, modifiée)

RM =10 (référencée, non modifiée)

RM =11 (référencée, modifiée)

4-L'algorithme de remplacement de la page la plus récemment utilisée (MRU Most Recently Used)

Lorsqu'un défaut de page se produit, c'est la dernière page utilisée qui sera retirée.

Le MRU permet de remédier aux inconvénients de LRU (trop gourmands en temps de calcul et difficiles à implémenter)

Le MRU est Facile à implémenter : nécessite de mémoriser seulement la dernière page référencée

Le MRU est assez efficace dans la pratique.

Exemple :

La mémoire contient 3 cadres,

La suite de références ={7,0,1,2,0,3,0,4,2,3,0,3,2,1,2,0,1,7,0,1}

L'algorithme produit 16 défauts de page comme le montre la figure suivante.

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
0	7	7	7	7		7	7	7		7	7	7	7	7	7	7	7		0	
1		0	0	0		3	0	4		4	4	4	4	4	4	4	4		4	
2			1	2		2	2	2		3	0	3	2	1	2	0	1		1	
Faut	X	X	X	X		X	X	X		X	X	X	X	X	X	X	X		X	

5-L'algorithme de vieillissement

C'est une amélioration logicielle de l'algorithme LRU en vu de le rendre moins coûteux.

-A chaque entrée d'une page dans la table de pages, on rajoute un champ compteur R qui est incrémenté de 1 quand une page est référencée.

-En réalité :

-R est rajouté au compteur à chaque interruption de l'horloge

-R est remis à 0 ensuite.

Avant chaque RAZ (remise à zéro), le système d'exploitation décale le compteur de 1 bit à droite (division par 2) et additionne R au bit du poids fort.

Quand un défaut de page se produit,

Si on remplace la page dont le compteur est le plus faible, on risque d'avoir la situation suivante :

-Une page accédée à maintes reprises il y a quelques temps sera considérée plus récente qu'une page accédée moins fréquemment et plus récemment.

Exercices TD

Exercice 01 (avec solution)

Appliquer l'algorithme FIFO sur la chaîne de références

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

sur un systèmes à 03 cadres (frames) puis à 04 cadres

Solution

1er cas : Trois (03) frames disponibles.

	1	2	3	4	1	2	5	1	2	3	4	5
Frame 1	1	1	1	4	4	4	5	5	5	5	5	5
Frame 2		2	2	2	1	1	1	1	1	3	3	3
Frame 3			3	3	3	2	2	2	2	2	4	4
Défaut de page	D	D	D	D	D	D	D			D	D	
Taux de défaut de page							(9/12) * 100 = 75%					

2eme cas : Trois (04) frames disponibles.

	1	2	3	4	1	2	5	1	2	3	4	5
Frame 1	1	1	1	1	1	1	5	5	5	5	4	4
Frame 2		2	2	2	2	2	2	1	1	1	1	5
Frame 3			3	3	3	3	3	3	2	2	2	2
Frame 4				4	4	4	4	4	4	3	3	3
Défaut de page	D	D	D	D			D	D	D	D	D	D
Taux de défaut de page							(10/12) * 100 = 83,33%					

Exercice 02

Appliquer l'algorithme LRU sur la chaîne de références

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

sur un systèmes à 03 cadres (frames) puis à 04 cadres

Exercice 03

Appliquer l'algorithme MRU sur la chaîne de références

1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

sur un systèmes à 03 cadres (frames) puis à 04 cadres

Exercice TP

Ecrire un programme en Java ou python qui permet de réaliser le travail de remplacement de pages des Algorithmes suivants :

FIFO

LRU

-Définir le nombre de cadre mémoire

-Définir la chaîne de références

-Appliquer FIFO

-Appliquer LRU

-Afficher la séquence de remplacement de chaque algorithme

-Afficher le nombre de défaut de page de chaque algorithme

Gestion des Fichiers (introduction)

Un processus en cours d'exécution peut enregistrer une quantité d'informations dans son espace d'adressage mais cette façon de faire présente trois inconvénients :

1-**La capacité de stockage est limitée à la mémoire vive**, Cette taille peut convenir pour certaines applications, mais elle est beaucoup trop faible pour d'autres (applications des banques, réservations, téléchargement p2p,...)

2-**Les informations stockées en mémoire vive sont perdues**, lorsque le processus se termine ou lors du plantage de l'ordinateur. Pour un grand nombre d'applications, les informations doivent être conservées pendant des durées importantes.

3-**Il ne peut pas y avoir d'accès simultané à ces informations**, Un répertoire téléphonique stocké dans l'espace d'adressage d'un processus ne peut être examiné que par ce seul processus (pour les raisons de protection des données expliquées lors de l'étude des processus), de telle sorte qu'on ne peut rechercher qu'un seul numéro à la fois.

Solution :

Pour résoudre ce problème, il faut rendre l'information indépendante d'un processus donné.

Trois caractéristiques sont donc requises pour le stockage des informations à long terme :

1-Il faut pouvoir stocker des informations de très grande taille ;

2-Les informations ne doivent pas disparaître lorsque le processus qui les utilise se termine ;

3-Plusieurs processus doivent pouvoir accéder simultanément aux informations.

La solution (utiliser les fichiers)

consiste à stocker les informations dans des fichiers sur des disques ou d'autres supports.

De cette façon,

-Les processus peuvent alors les lire ou en écrire de nouvelles.

-Les informations stockées dans des fichiers doivent être permanentes, c'est-à-dire non affectées par la création ou la fin d'un processus.

-Un fichier ne doit disparaître que lorsque son propriétaire le supprime explicitement.

Gestion des Fichiers

Un fichier est un ensemble d'informations stockées sur le disque.

Le système de fichiers est la partie du système d'exploitation qui est chargée de gérer les fichiers.

La gestion consiste en la création (identification, allocation d'espace sur disque), la suppression, les accès en lecture et en écriture, le partage de fichiers et leur protection en contrôlant les accès.

Qu'est-ce qu'un fichier ?

Un fichier est une suite d'octets. Par contre, les utilisateurs peuvent donner des significations différentes au contenu d'un fichier (suites d'octets, suite d'enregistrements, arbre, etc.).

Les fichiers sont généralement créés par les utilisateurs. Toutefois certains fichiers sont générés par les systèmes ou certains outils tels que les compilateurs. Afin de différencier les fichiers entre eux, chaque fichier a un ensemble d'attributs qui le décrivent.

Chaque fichier est identifié par un nom auquel on associe un emplacement sur le disque (une référence) et possède un ensemble de propriétés :

-Ses attributs.

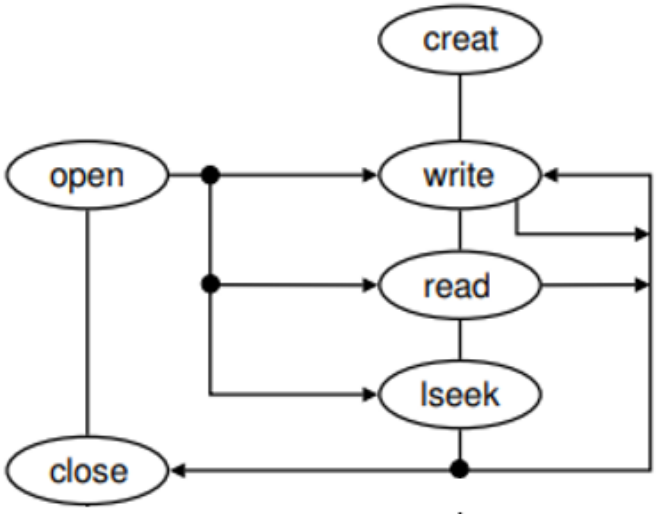
-Le nom est en général, composé de deux parties séparées par un point (la partie qui suit le point est appelée extension : prog.c, entete.h, fich.doc, archivo.pdf, etc.).

-L'extension peut être de taille fixe, comme dans MS-DOS ou variable comme c'est le cas d'Unix/Linux ; obligatoire ou non.

-L'extension est nécessaire dans certains cas. Par exemple, le compilateur C rejettera le fichier prog.txt même si son contenu est un programme C. Le nom d'un fichier peut être sensible à la typographie : ainsi en Unix/Linux Archivo ≠ archivo

Cycle de vie d'un fichier Les fichiers

Comme tous les autres composants qui ont un cycle de vie. Les fichiers sont créés (ou ouverts), modifiés (écrire), on lits à partir d'eux et finalement, ils meurent (sont effacés). Ceci est illustré par la figure suivante.



Creat : Creation de fichier,

Open : Ouverture d'un fichier,

Wrire : Ecriture dans le fichier,

Read : Lecture du fichier,

Iseek : Recherche dans le fichier,

Close : Fermeture de fichier,

Le système de gestion de fichiers

Le système de gestion de fichiers (SGF) est la partie la plus visible d'un système d'exploitation qui se charge de gérer le stockage et la manipulation de fichiers (sur une unité de stockage : partition, disque, CD, disquette. Un SGF a pour principal rôle de gérer les fichiers et d'offrir les primitives pour manipuler ces fichiers.

Le SGF est généralement composé de plusieurs niveaux différents. La figure suivante donne un exemple d'architecture de SGF :

------Programme d'Application------

Le système de fichier logique
Le module d'organisation de fichiers
Le système de fichiers de base
Le contrôle des E/S

------Les périphériques------

Les couches d'un SGF

1-Le contrôle des E/S : est constitué de drivers et des handlers (routines d'interruption) pour transférer l'information entre la mémoire et le système de disques.

On peut considérer le driver comme un traducteur : ses entrées consistent en des commandes de haut niveau comme « récupérer le bloc 123 ». Ses sorties sont des instructions de bas niveau, spécifiques au matériel, utilisés par le contrôleur du matériel qui relie le périphérique d'E/S au reste du système. Le driver écrit généralement des configurations binaires spécifiques dans des emplacements spéciaux de la mémoire du contrôleur d'E/S afin de lui indiquer sur quel emplacement du périphérique agir et quelles actions entreprendre.

2-Le système de fichiers de base : doit seulement émettre des commandes génériques pour le driver approprié afin de lire et d'écrire les blocs physiques sur le disque.

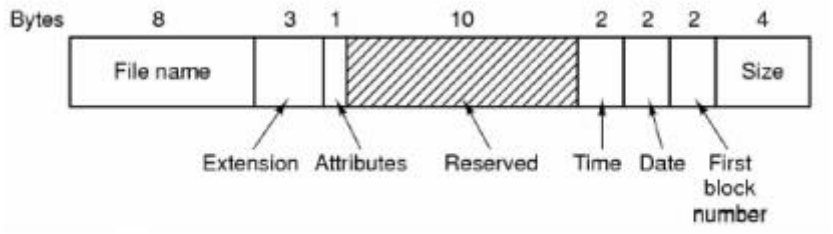
3-Le module d'organisation de fichiers : connaît les fichiers et leurs blocs logiques, ainsi que les blocs physiques. En connaissant le type d'allocation de fichiers employé et l'emplacement du fichier, ce module peut traduire les adresses des blocs logiques dans les adresses des blocs physiques pour que le système de transfert de base les transfère. Les blocs logiques du fichier sont numérotés de 0 à N, tandis que les blocs physiques contenant ces données ne correspondent généralement pas aux numéros logiques, il faut donc une traduction pour localiser chaque bloc. Le module d'organisation de fichiers comprend également le gestionnaire de l'espace libre, qui suit la piste des blocs disponibles et les fournit au module d'organisation de fichiers quand celui-ci les demande.

4-Le système de fichier logique : utilise la structure de répertoires pour proposer au module d'organisation de fichiers l'information dont ce dernier a besoin, pour un nom donné de fichier logique est également responsable de la protection et de la sécurité.

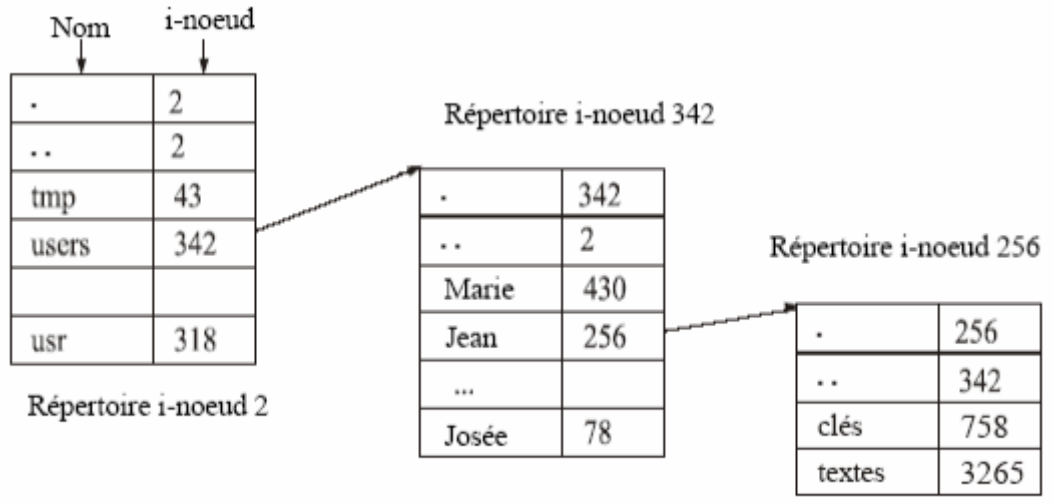
Pour créer un nouveau fichier, un programme d'application appelle le SGF. Ce dernier connaît le format de la structure des répertoires. Il lit le répertoire approprié dans la mémoire, l'actualise avec la nouvelle entrée et le réécrit sur disque. Une fois que le répertoire a été actualisé, le SGF peut l'utiliser pour exécuter les E/S.

Qu'est-ce qu'un répertoire ?

Un répertoire est une entité crée pour l'organisation des fichiers. En effet on peut enregistrer des milliers, voire des millions de fichiers sur un disque dur et il devient alors impossible de s'y retrouver. Avec la multitude de fichiers créés, le système d'exploitation a besoin d'une organisation afin de structurer ces fichiers et de pouvoir y accéder rapidement. Cette organisation est réalisée au moyen de répertoires également appelés catalogues ou directory. Un répertoire est lui-même un fichier puisqu'il est stocké sur le disque et est destiné à contenir des fichiers. Du point de vue SGF, un répertoire est un fichier qui dispose d'une structure logique : il est considéré comme un tableau qui contient une entrée par fichier. L'entrée du répertoire permet d'associer au nom du fichier (nom externe au SGF) les informations stockées en interne par le SGF. Chaque entrée peut contenir des informations sur le fichier (attributs du fichier) ou faire référence à (pointer sur) des structures qui contiennent ces informations.



Structure d'un répertoire : cas de MS-DOS (32 octets)



Structure d'un répertoire : cas d'UNIX (14 octets)

Dans ce cas, chaque fichier à un i-noeud

On distingue plusieurs structures pour les répertoires :

• La structure plate à un niveau : organisée en plusieurs répertoires mais chacun d'eux ne peut contenir que des fichiers. Aujourd'hui absurde, cette approche existait à l'époque des premiers systèmes d'exploitation car le nombre de fichiers était limité.

• La structure à deux niveaux : chaque utilisateur dispose de son propre répertoire dans lequel il peut conserver des fichiers et des répertoires.

• La structure arborescente : contient un nombre arbitraire de niveaux et chaque répertoire peut contenir des fichiers et des sous répertoires.

Le nom complet d'un fichier est formé d'une liste des répertoires qu'il faut traverser à partir du haut de la hiérarchie (le répertoire racine (root directory)) plus le nom_du_fichier. Les répertoires sont séparés par un caractère qui dépend du système d'exploitation : " >" pour Multics, "/" pour UNIX, " \" pour Dos et Winxx et " : " pour MacOS. Un tel chemin (exprimé à partir de la racine) est appelé chemin absolu. Voici un exemple de chemin absolu sous MS-DOS **c:\cours\chapitre4.txt** et sous Unix **/home/user1/rapport.txt**. Par contre, un chemin qui ne commence pas par la racine est un **chemin relatif**. Ces deux concepts de fichier et de répertoire sont considérés par le système d'exploitation comme une seule entité différentiable par un bit à rajouter aux attributs.

En Unix, le répertoire racine (le répertoire /) contient les sous répertoires suivants :

/bin	commandes binaires utilisateur essentielles (pour tous les utilisateurs)
/boot	fichiers statiques du chargeur de lancement
/dev	fichiers de périphériques
/etc	configuration système spécifique à la machine
/home	répertoires personnels des utilisateurs
/lib	bibliothèques partagées essentielles et modules du noyau
/mnt	point de montage pour les systèmes de fichiers montés temporairement
/proc	système de fichiers virtuel d'information du noyau et des processus
/root	répertoire personnel de root (optionnel)
/sbin	binaires système (binaires auparavant mis dans /etc)
/sys	*état des périphériques (model device) et sous-systèmes (subsystems)*
/tmp	fichiers temporaires

La gestion de l'organisation de l'espace disque Sur le disque,

Un fichier est sauvegardé sur un ensemble de clusters, appelés également blocs. Le SGF manipule alors des blocs numérotés de 0 à N-1 (N = taille du disque/taille d'un bloc). Chaque fichier (ordinaire ou répertoire) d'un système de fichiers est stocké sur l'unité de stockage du système de fichiers. Ses données sont dans des blocs de taille fixe (512, 1024, ou 2048 octets, …) et à chaque fichier est alloué un nombre de blocs. La lecture ou l'écriture d'un élément d'un fichier impliquera le transfert vers la mémoire du bloc entier qui contient cet élément.

Techniques d'allocation des blocs sur le disque,

On distingue trois manières d'organiser les blocs d'un fichier : contiguë, chaînée et indexée.

Allocation contiguë :

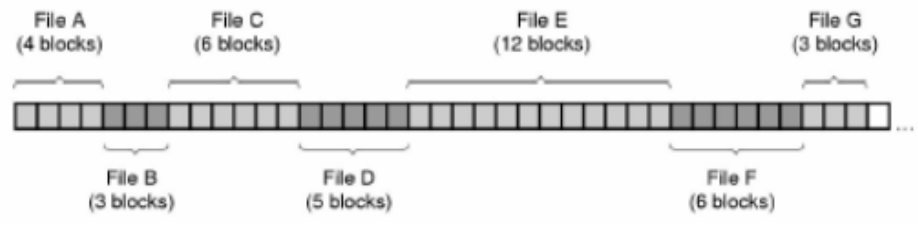
Pour chaque fichier à enregistrer, le système recherche une zone suffisamment grande pour accueillir le fichier. Le fichier sera alors constitué de plusieurs blocs contigus. Cette méthode présente l'avantage de la rapidité de l'accès (les blocs étant contigus, on limite les déplacements de la tête le lecture/écriture, coûteux en temps). Cependant, elle présente un grand nombre d'inconvénients :

• Le dernier bloc a toutes chances d'être sous-utilisé et ainsi, on gaspille de la place. Le pourcentage de place perdue est d'autant plus grand que la taille moyenne des fichiers est faible, ce qui est la réalité

• Il est difficile de prévoir la taille qu'il faut réserver au fichier : un fichier est amené à augmenter de taille, par conséquent il faut prévoir de l'espace libre après le dernier secteur alloué. Si le fichier est agrandi, il faudra le déplacer pour trouver un nouvel ensemble de blocs consécutifs de taille suffisante.

• La perte d'espace sur le disque : si on prévoit trop d'espace libre, le fichier risque de ne pas l'utiliser en entier. En revanche, si on prévoit trop peu d'espace libre, le fichier risque de ne pas pouvoir être étendu.

• Problème de fragmentation externe : c'est l'espace perdu en dehors des fichiers. On peut effacer des données ou supprimer des fichiers ce qui libère des blocs sur le disque. Au fil de l'utilisation, il peut se créer un grand nombre de petites zones dont la taille ne suffit souvent pas pour allouer un fichier mais dont le total correspond à un espace assez volumineux.



Table

Fichier	Début	Fin
A	0	3
B	4	6
C	7	12
D	13	17
E	18	29
F	30	35
G	36	38

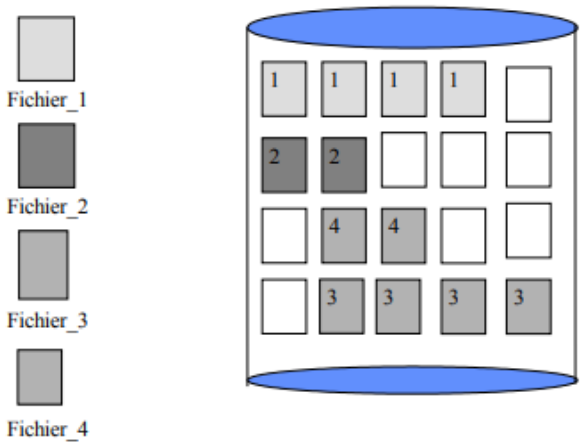
Allocation contiguë d'espace disque pour 7 fichiers

Il existe plusieurs algorithmes utilisés pour mettre un fichier sur le disque. Dans tous les cas, il faut trouver un espace suffisamment grand pour le fichier et pour sa croissance anticipée. Comme pour l'allocation contigüe de mémoire pour les processus, on retrouve les algorithmes de "first-fit", "best-fit" et "worst-fit".

Note : L'allocation contigüe implique l'existence d'une table qui donne l'emplacement de départ de chaque fichier sur le disque.

Exercice

Soit le schéma suivant représentant l'allocation du disque C sur un système pour lequel l'allocation est gérée selon la méthode d'allocation contigüe. Les blocs sont numérotés de 1 à 20, ligne à ligne, de gauche à droite. Les rectangles blancs sont des blocs libres. Les numéros apparaissant dans les blocs colorés correspondent au numéro du fichier auquel le bloc appartient (exemple : 1 pour fichier_1).



Question 1 : Donnez une représentation possible de l'ensemble des blocs libres du disque.

Question 2 : On souhaite allouer un nouveau fichier (Fichier_5) d'une taille de 3 blocs. Quels sont les blocs alloués à ce fichier : - si l'allocation réalisée est de type First Fit ? - si l'allocation réalisée est de type Best Fit ?

Question 3

Ayant réalisé l'allocation Best Fit précédente, on souhaite à présent étendre le fichier Fichier_4 pour lui ajouter un bloc supplémentaire. Que se passe-t-il ? Proposez une solution éventuelle.

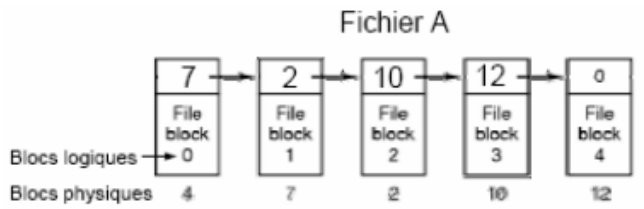
Allocation chaînée (non contiguë) :

Le principe est d'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. Lorsque le fichier change de taille, la gestion des blocs occupés est simple. Il n'y a donc aucune limitation de taille, si ce n'est l'espace disque lui-même.

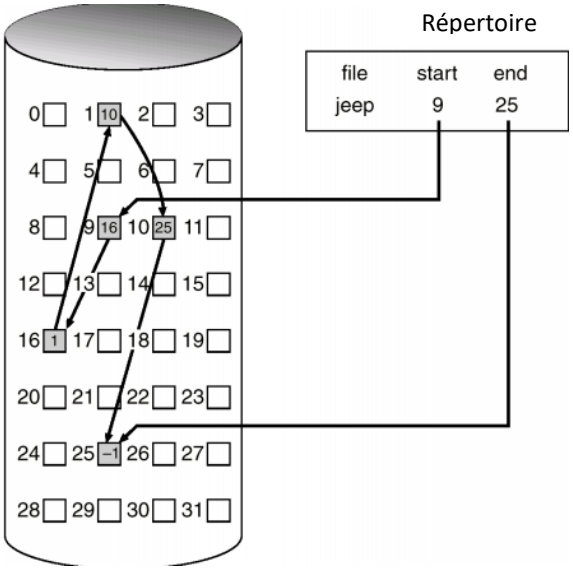
Cette méthode présente l'avantage de l'élimination du problème de fragmentation externe. Aussi le fait de ne pas nécessiter une structure spéciale pour sa mise en place, constitue un autre avantage. En revanche, les inconvénients ici aussi sont multiples :

• L'accès au fichier est totalement séquentiel, on doit toujours commencer le parcours du fichier à partir du début.

• La perte d'un chaînage entraîne la perte de tout le reste du fichier. Pire encore, il suffit qu'une valeur soit modifiée dans un pointeur pour qu'on se retrouve dans une autre zone de la mémoire.



Allocation chaînée



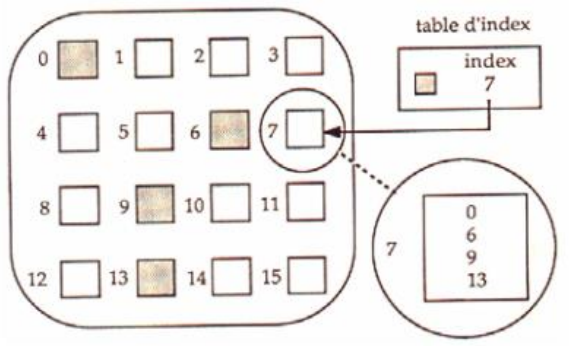
Une autre représentation

Allocation indexée :

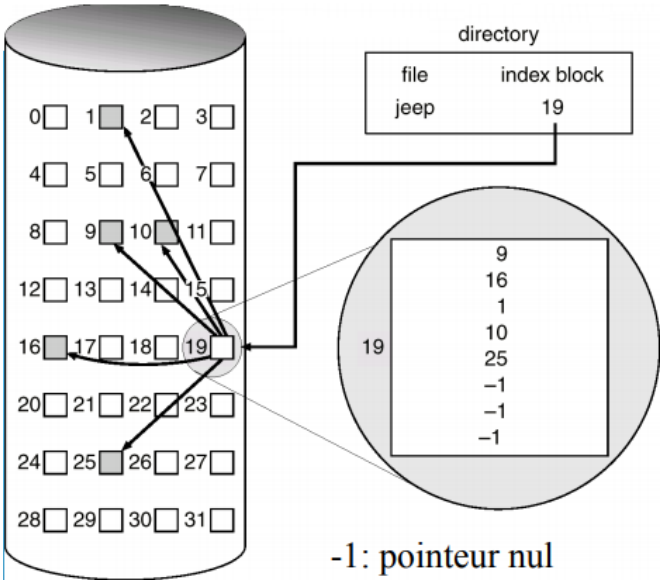
Tous les inconvénients de l'allocation chaînée peuvent être résolus de la manière simple suivante :

Il suffit de retirer les pointeurs des blocs et de les placer dans une structure de données gardée en mémoire centrale, ainsi, les informations sur les numéros de blocs peuvent être obtenue à tout moment.

L'allocation chaînée résout les problèmes de fragmentation, cependant elle ne peut pas supporter l'accès direct de façon efficace, car les pointeurs vers les blocs ainsi que les blocs eux-mêmes sont dispersés dans tout le disque et ils doivent être récupérés dans l'ordre. L'allocation indexée résout ce problème en rangeant tous les pointeurs dans un seul emplacement : le bloc d'index. Chaque fichier possède son propre bloc index, qui est un tableau d'adresse de blocs disque. La ième entrée dans le bloc index pointe sur le ième bloc du fichier. Le répertoire contient l'adresse du bloc index. Pour lire le ième bloc, on utilise le pointeur de la ième entrée du bloc index afin de trouver et de lire le bloc désiré.



Allocation indexée



Une autre représentation

Quand on crée le fichier, tous les pointeurs du bloc index sont fixés à nul. Quand le ième bloc est écrit pour la première fois, on obtient un bloc du gestionnaire de l'espace libre et son adresse est placée dans la ième entrée du bloc index.

Inconvénient : L'espace occupé par le bloc d'index.

Chaque fichier doit posséder un bloc d'index dont il est souhaitable qu'il soit le plus petit possible. Cependant s'il est trop petit, il ne pourra pas ranger des pointeurs en nombre suffisant pour un grand fichier et il faudrait un mécanisme pour traiter ce détail.

Pour cela plusieurs solutions ont été proposées, dont :

1-Schéma chaîné : Pour des fichiers importants, on peut chaîner les blocs d'index entre eux.

2-Index multiniveaux : Cette solution consiste à utiliser un bloc d'index séparé pointant sur des blocs d'index.

La création de fichier par le SE

Un fichier ou un répertoire sont tous les deux créés suivant les mêmes étapes qui sont les suivantes :

• La création d'une structure de données pour décrire le fichier, Les attributs du fichier sont sauvegardés dans cette structure de données.

• La création du fichier proprement dit. Il s'agit d'allouer au fichier un certain nombre de blocs sur le disque selon sa taille. Les blocs d'un fichier vont contenir des données sans aucune structure particulière. Les blocs d'un répertoire ont par contre une structure bien particulière, en effet ils contiennent des noms et des attributs de fichiers et de sous répertoires.

La gestion de l'espace libre sur le disque

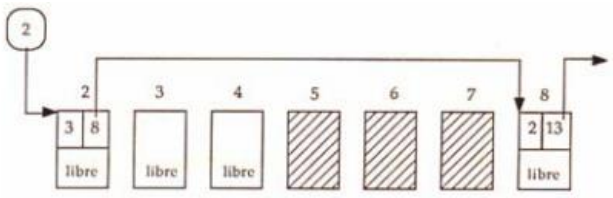
Les systèmes d'exploitation utilisent essentiellement deux approches pour mémoriser l'espace libre : une statique et une dynamique.

• **Bitmap** : Approche statique utilise une table de bits (vecteur de bits n blocs) comportant autant de bits que de blocs sur le disque. A chaque bloc du disque, correspond un bit dans la table, positionné à 1 si le bloc est occupé, à 0 si le bloc est libre (ou vice versa).

Vecteur de bits à n bloc

Si les blocs 3, 4, 5, 9, 10, 15, 16 sont libres : 11100011100111100… Cette solution est utilisée pour trouver n blocs contigus.

• **Liste chaînée** : Approche dynamique utilise une liste chaînée constituée d'éléments, chacun mémorisant des numéros de blocs libres. Tous les blocs libres sont liés ensemble par des pointeurs.



Exemple d'utilisation d'une liste chaînée pour la gestion de l'espace libre

Protection

Quand on maintient de l'information dans un système informatique, l'un des principaux problèmes est sa protection contre les dégâts (fiabilité) et les accès non autorisés (protection). On peut fournir de la protection de plusieurs manières différentes : protection par type, protection par groupe d'accès, … etc.

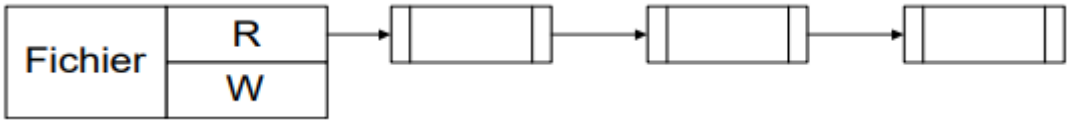
Protection par type

Le besoin de protéger des fichiers est une conséquence directe de la possibilité d'y accéder. Dans les systèmes dont l'accès aux fichiers des autres utilisateurs n'est pas permis toujours, on fournit un accès contrôlé. Les mécanismes de protection fournissent un accès contrôlé en limitant les types d'accès possibles aux fichiers. On autorise ou on refuse un accès selon plusieurs facteurs, l'un d'eux est le type d'accès demandé. On peut contrôler différents types d'opérations : Lecture, Ecriture, Exécution, Ajout, Destruction, Enumération.

Protection par groupe d'accès

L'approche la plus commune au problème de la protection consiste à rendre l'accès dépendant de l'identité de l'utilisateur. Divers utilisateurs peuvent avoir besoin de types d'accès différents à un fichier ou à un répertoire. Le schéma le plus général pour implémenter l'accès dépendant de l'identité consiste à associer à chaque fichier et répertoire **une liste d'accès**, en spécifiant le nom de l'utilisateur et les types d'accès autorisés à chaque utilisateur. Quand un utilisateur demande à accéder à un fichier particulier, le SE examine la liste d'accès associée à ce fichier. Si cet utilisateur se trouve dans la liste, l'accès est autorisé, sinon il se produit une violation de la protection.

Le principal problème concernant les listes d'accès est leur longueur. Si on désire autoriser tout le monde à lire un fichier, on doit construire une liste de tous les utilisateurs ayant le droit de le lire ; ce qui peut constituer une tâche pénible pour le système.



Afin de réduire la longueur de la liste d'accès, plusieurs systèmes reconnaissant trois types d'utilisateurs en liaison avec chaque fichier :

- Propriétaire : L'utilisateur qui a créé le fichier en est le propriétaire.

- Groupe : Le groupe est un ensemble d'utilisateurs partageant le fichier et ayant des besoins d'accès semblables.

- Univers : Tous les autres utilisateurs du système constituent l'univers. La protection par groupe ne peut fonctionner correctement que si l'appartenance au groupe est contrôlée.

Dans le système Unix, les groupes ne peuvent être chargés ou créés que par l'administrateur du système. Avec cette classification, il faut seulement 3 bits pour définir la protection ; chacun d'eux autorise ou interdit l'accès. Par exemple, le système Unix définit 3 champs **rwx** pour respectivement : la lecture, l'écriture, et l'exécution ou maintient un champ séparé pour le propriétaire, le groupe et les autres.

Exemple :

	Propriétaire	Groupe	Les autres
Student.doc	rwx	rwx	rwx
Programme.c	rw-	r--	r--