Système d’exploitation I

Plan

I. Introduction aux systèmes d'exploitation
- Les composantes du Système d'Exploitation
- Les fonctions du Système d'Exploitation
- Les classes du système d'exploitation

II. Système de Gestion de Fichiers
- Notions préliminaires
- Le concept de fichier
- Structure d'un système de fichiers
- La gestion de l'organisation de l'espace disque
- La gestion de l'espace libre sur le disque
- Quelques Systèmes de Gestion de Fichiers
- Système de Gestion de Fichiers

III. Gestion de la mémoire
- Gestion sans recouvrement ni pagination
- Gestion avec recouvrement ("va et vient" ou "swapping") sans pagination
- Gestion avec recouvrement, avec pagination ou segmentation

Chapitre 1

I. Introduction aux systèmes d’exploitation

- Les composantes du Système d’Exploitation
- Les fonctions du Système d’Exploitation
- Les classes du système d’exploitation

Introduction

- Le système d'exploitation est le logiciel le plus important d'un ordinateur.

- Le regroupe un ensemble de programmes qui permettent l'utilisation de l'ordinateur et de ses périphériques (écran, imprimante, clavier…). Sans système d'exploitation, l'ordinateur n'est pas capable de gérer les périphériques.

- Il permet tout d'abord le démarrage de l'ordinateur et est indispensable à la mise en œuvre des autres programmes présents sur l'ordinateur.

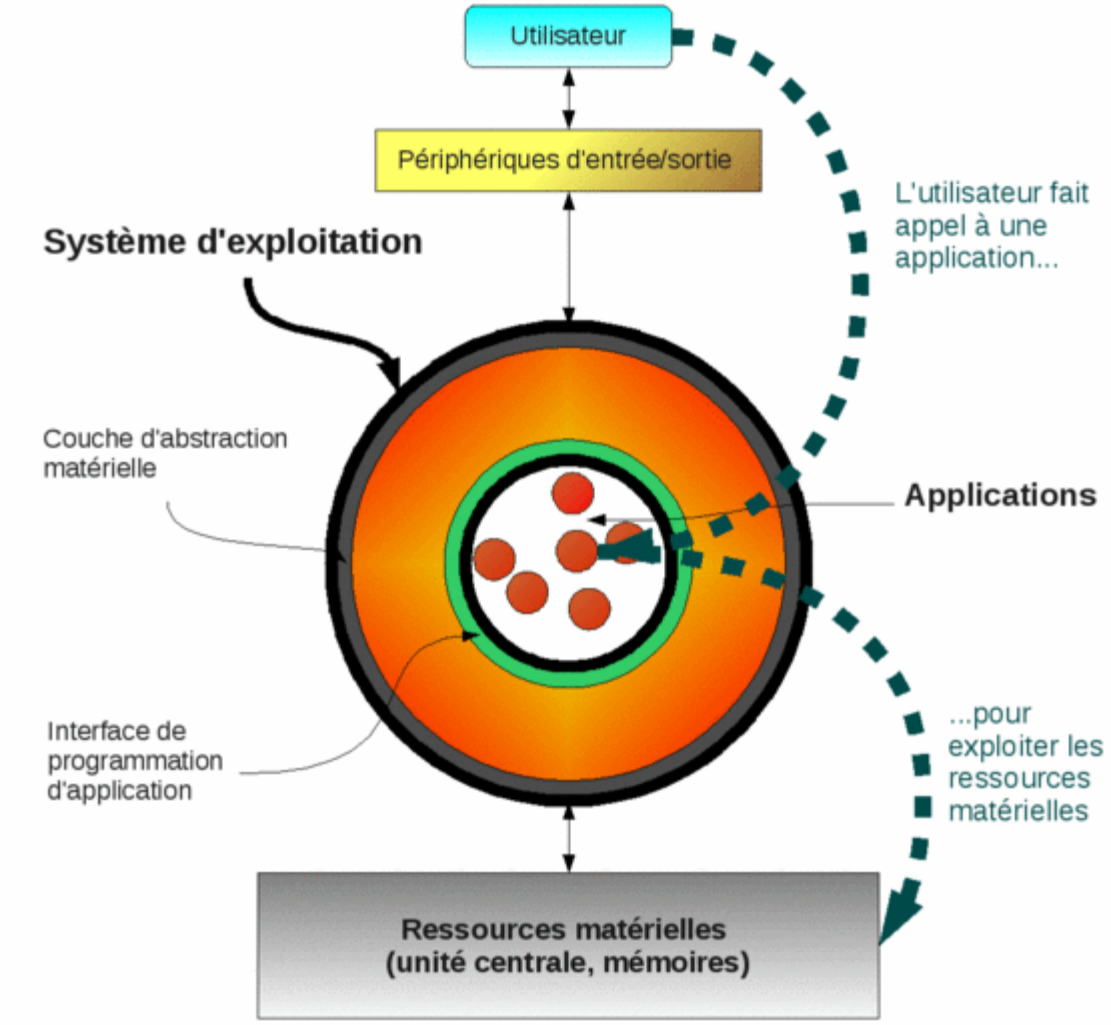
=> Il assure donc son fonctionnement général.

Introduction

- Le Système d'exploitation, est chargé d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications.

- Lorsqu'un programme désire accéder à une ressource matérielle, il ne lui est pas nécessaire d'envoyer des informations spécifiques au périphérique, il lui suffit d'envoyer les informations au système d'exploitation, qui se charge de les transmettre au périphérique concerné via son pilote (programme informatique également nommé Driver, destiné à permettre à un autre programme d'interagir avec un périphérique).

- Le système d'exploitation permet ainsi de dissocier les programmes et le matériel, afin notamment de simplifier la gestion des ressources et d'offrir à l'utilisateur une interface homme-machine (IHM) simplifiée afin de lui permettre de s'affranchir de la complexité de la machine physique.



Les relations entre utilisateur, applications, système d'exploitation et matériel

Composantes du système d'exploitation

- Le système d'exploitation est composé d'un ensemble de logiciels permettant de gérer les interactions avec le matériel :
  - Le noyau (Kernel) représentant les fonctions fondamentales du système d'exploitation telles que la gestion de la mémoire, des processus, des fichiers, des entrées-sorties principales et des fonctionnalités de communication.
  - L'interpréteur de commande (en anglais « shell ») permettant la communication avec le système d'exploitation par l'intermédiaire d'un langage de commande, afin de permettre à l'utilisateur de piloter les périphériques en ignorant tout des caractéristique du matériel qu'il utilise, de la gestion des adresses physiques…
  - Le système de fichier (en anglais « file system » FS), permettant d'enregistrer les fichiers dans une arborescence.

Composantes du système d'exploitation

- Les bibliothèques servant à regrouper les opérations les plus utilisées dans les programmes informatiques. On distingue généralement deux types de bibliothèques :
  - Les bibliothèques système,
  - Les bibliothèques utilitaires : fonctions mathématiques, fonctions de tri

- Les outils système permettant de configurer le système (gérer les comptes des utilisateurs, configuration des paramètres réseau, démarrage automatique des services, ...)

- Les programmes applicatifs de base offrant des services à l'utilisateur (calculatrice, éditeur de texte, navigateur web,...). Ils sont souvent fournis en paquet promotionnel avec le système d'exploitation.

Rôles du système d'exploitation

- Un système d'exploitation doit garantir un bon niveau en matière de :
  - Sécurité : intégrité, contrôle des accès confidentialité…
  - Fiabilité : degré de satisfaction des utilisateurs même dans des conditions hostiles et imprévues,
  - Efficacité : performances et optimisation du système pour éviter tout surcoût en termes de temps et de places consommées par le système au détriment de l'application.

- La qualité de l'interface (en particulier pour les systèmes interactifs)
  - Convivialité
  - Simplicité d'utilisation
  - Documentation
  - Bonne intégration au réseau
  - Sécurité et protection
  - Répertoire étendu des fonctions

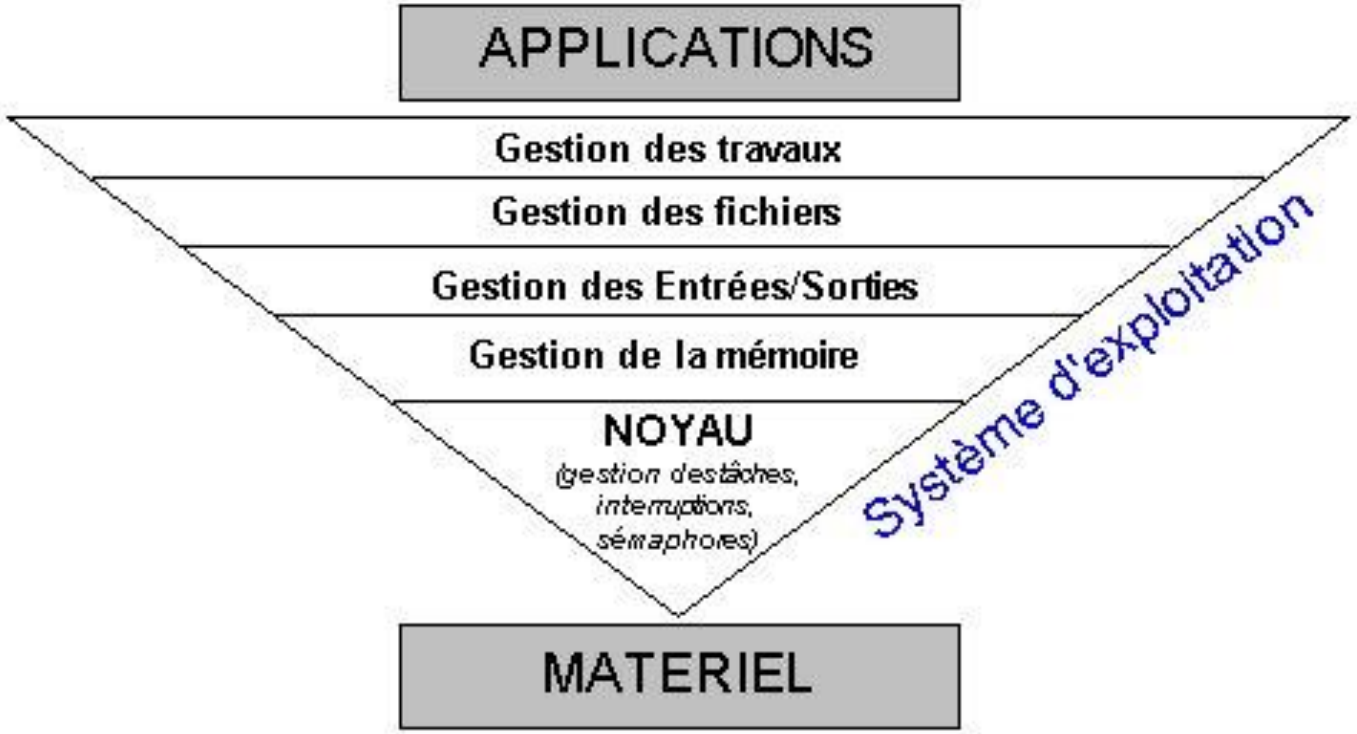
Rôles du système d'exploitation

Les rôles de l'OS sont divers et concernent notamment :

- **La gestion du processeur** : le système d'exploitation est chargé de gérer l'allocation du processeur entre les différents programmes grâce à un algorithme d'ordonnancement ; Le type d'ordonnanceur est totalement dépendant du système d'exploitation, en fonction de l'objectif visé.
- **Gestion de la mémoire vive** : le système d'exploitation est chargé de gérer l'espace mémoire alloué à chaque application et, le cas échéant, à chaque utilisateur. En cas d'insuffisance de mémoire physique, le système d'exploitation peut créer une zone mémoire sur le disque dur, appelée « mémoire virtuelle ». La mémoire virtuelle permet de faire fonctionner des applications nécessitant plus de mémoire qu'il n'y a de mémoire vive disponible sur le système. En contre partie, cette mémoire est beaucoup plus lente.
- **Gestion des entrées/sorties** : le système d'exploitation permet d'unifier et de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes.
- **Gestion de l'exécution des applications** : le système d'exploitation est chargé de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Il permet à ce titre de « tuer » une application ne répondant plus correctement.

Rôles du système d'exploitation

- **Gestion des droits** : le système d'exploitation est chargé de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont pas utilisées que par des programmes et utilisateur possédant les droits adéquats.
- **Gestion des fichiers** : le système d'exploitation gère la lecture et l'écriture dans le système de fichier et des droits d'accès aux fichiers par les utilisateurs et les applications.
- **Gestion des informations** : le système d'exploitation fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine.
- **Gestion des processus** : le système d'exploitation gère les opérations de mise à jour des processus ainsi que leur synchronisation et la communication entre eux.
- **Gestion des réseaux** : le système d'exploitation gère la communication, le partage des ressources et l'échange d'informations entre les postes de travail du réseau.
- **Gestion des commandes utilisateurs**


APPLICATIONS
Gestion des travaux
Gestion des fichiers
Gestion des Entrées/Sorties
Gestion de la mémoire
NOYAU
(gestion d'estâches,
interruptions,
sémaphores)
Système d'exploitation
MATERIEL

13

Familles de systèmes

14

Familles de systèmes

Selon les services rendus

Selon les services rendus

- **Mono/Multi-tâches** : capacité du système à pouvoir exécuter plusieurs processus simultanément. Par exemple, effectuer une compilation et consulter le fichier source du programme correspondant.
  - Mono-tâche (DOS) : A tout instant, un seul programme est exécuté; un autre programme ne démarrera, sauf conditions exceptionnelles, que lorsque le premier sera terminé.
  - Multi-tâches (Windows, Unix, Linux) : plusieurs processus (i. e. un "programme" en cours d'exécution) peuvent s'exécuter simultanément (systèmes multiprocesseurs) ou en quasi-parallélisme (systèmes à temps partagé : système monoprocesseur).
- **Mono/multi-utilisateurs** : capacité à pouvoir gérer un panel d'utilisateurs utilisant simultanément les mêmes ressources matérielles.

  Exemple : UNIX, de MVS, de Gecos…

16

Familles de systèmes

Selon leur architecture

Selon leur architecture

- **Système centralisés** : l'ensemble du système est entièrement présent sur la machine considérée. Les machines éventuellement reliées sont vues comme des entités étrangères disposant elle aussi d'un système centralisé. Le système ne gère que les ressources de la machine sur laquelle il est présent.

  Exemple: C'est le cas d'Unix, même sir les applications réseaux (X11, FTP, Mail…) se sont développées

.

- **Système répartis** (« destributed systems ») : les différentes abstractions du système sont réparties sur un ensemble (domaine) de machine (site). Le système d'exploitation réparti apparait aux yeux de ses utilisateurs comme une machine virtuelle monoprocesseur même lorsque cela n'est pas le cas. Avec un système réparti, l'utilisateur n'a pas à se soucier de la localisation des ressources. Quand il lance un programme, il n'a pas à connaitre le composant de la machine qui l'exécutera. Ils exploitent au mieux les capacités de parallélisme d'un domaine. Ils offrent des solutions aux problèmes de la résistance aux pannes.

18

Familles de systèmes

Selon leur capacité à évoluer

Selon leur capacité à évoluer

- **Systèmes fermés** (ou propriétaires) :
  - **Extensibilité réduite** : quand on veut ajouter des fonctionnalités à un système fermé, il faut remettre en cause sa conception et refaire une archive (système complet).
  - Il n'y a aucun ou peu d'échange possible avec d'autres système de type différent, voire avec des types identiques.

- **Systèmes ouverts** :
  - Extensibilité accrue : il est possible d'ajouter des fonctionnalités et des abstractions sans avoir à repenser le système et même sans avoir à l'arrêter sur une machine. Cela implique souvent une conception modulaire basée sur le modèle « client – serveur ».

20

Familles de systèmes

Selon l'architecture matérielle qui les supporte

Selon l'architecture matérielle qui les supporte

- **Architecture monoprocesseur** (temps partagé ou multiprogrammation) : Ressource processeur unique : développer un mécanisme de gestion des processus pour offrir un (pseudo) parallélisme à l'utilisateur : c'est la multiprogrammation, il s'agit en fait d'une commutation rapide entre les différents processus pour donner l'illusion d'un parallélisme.
- **Architecture multiprocesseurs** (parallélisme) :
  - On trouve une grande variété d'architectures multiprocesseurs :
  - SIMD (single instruction multiple data) : tous les processeurs exécutent les mêmes instructions mais sur des données différentes.
  - MIMD (Multiple Instruction Multiple Date) : chaque processeur est complètement indépendant des autres et exécute des instructions sur des données différentes
  - Pipeline : les différentes unités d'exécution sont mises en chaine et sont chacune partie du traitement a effectuer.
- **Architecture fortement couplée** :
  - Ce sont principalement des architectures à mémoire commune.
- **Architecture faiblement couplée** :
  - Ce sont des architectures où chaque processeur possède sa propre mémoire locale ; c'est le cas d'un réseau de stations.
- **Architecture mixte** :
  - Ce sont des architectures à différents niveau de mémoire (commune et privée).

Système temps réel

Un cas particulier, les systèmes « temps réel »

- Un système d'exploitation temps réel (RTOS) est un système d'exploitation multitâche destiné aux applications temps réel. Ces applications comprennent les systèmes embarqués (thermostats programmables, contrôleurs électroménagers, téléphones mobiles), des robots industriels, les vaisseaux spatiaux, les systèmes de contrôle commande industriel, et le matériel de recherche scientifique.
- On distingue deux types de contraintes temporelles : les contraintes strictes et les contraintes relatives.
- Pour garantir ces contraintes, le système possède des mécanismes spécifiques dont le but est de réduire les durées d'exécution à l'infini des programmes.
- RTOS utilise des ordonnanceurs spécialisées afin de fournir aux développeurs des systèmes temps réel les outils et les primitives nécessaires pour produire un comportement temps réel souhaité dans le système final.
- Deux types de conceptions existent :
  - Evènementielle (ordonnancement par priorité): l'ordonnanceur ne change de tâche que lorsqu'un évènement de plus haute priorité à besoin de service
  - Par partage de tâches : L'ordonnanceur change de tâche aux interruptions de l'horloge, et lors des évènements.

Chapitre 2

II. Système de Gestion de Fichiers

- Notions préliminaires
- Le concept de fichier
- Structure d’un système de fichiers
- La gestion de l’organisation de l’espace disque
- La gestion de l’espace libre sur le disque
- Quelques Systèmes de Gestion de Fichiers

Le système de gestion des Fichiers (SGF)

- La partie la plus visible d'un système d'exploitation qui se charge de gérer le stockage et la manipulation des fichiers sur une unité de stockage : partition, disque, CD, disquette.
- Il effectue généralement les tâches suivantes :
  - Fournir une interface conviviale pour manipuler les fichiers : généralement, l'utilisateur fournit seulement le nom et l'extension du fichier, les autres attributs sont gérés implicitement par le SGF.

    Cette interface fournit la possibilité d'effectuer plusieurs opérations sur les fichiers (ouvrir, fermer, copier, renommer des fichiers et des répertoires.
  - Gérer l'organisation des fichiers sur le disque (allocation de l'espace disque aux fichiers).
  - Gérer l'espace libre sur le disque dur.

Structure d’un disque dur

- **Plateaux** : un disque dur est usuellement constitué de plusieurs plateaux coaxiaux entraînés en rotation. Chaque plateau comporte deux faces. Une tête de lecture/écriture par face est présente.

  Chaque face est divisée en :

  Secteurs : découpage longitudinal;

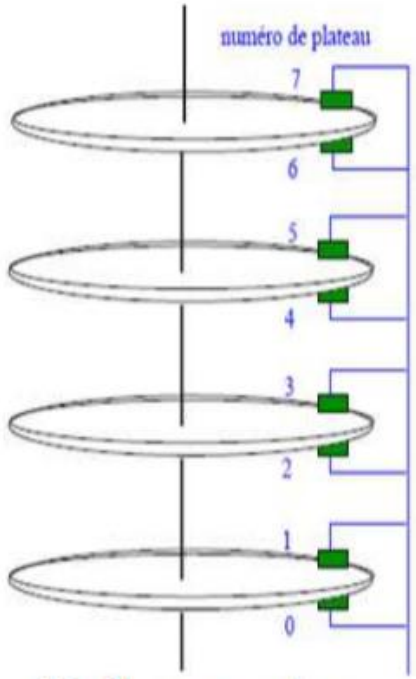
  Pistes : découpage circulaire;

- **Cylindre** : ensemble de pistes de même numéro situées sur tous les plateaux.
- **Cluster** (ou bloc) : Le cluster est la plus petite unité de disque que le système d’exploitation est capable de gérer.

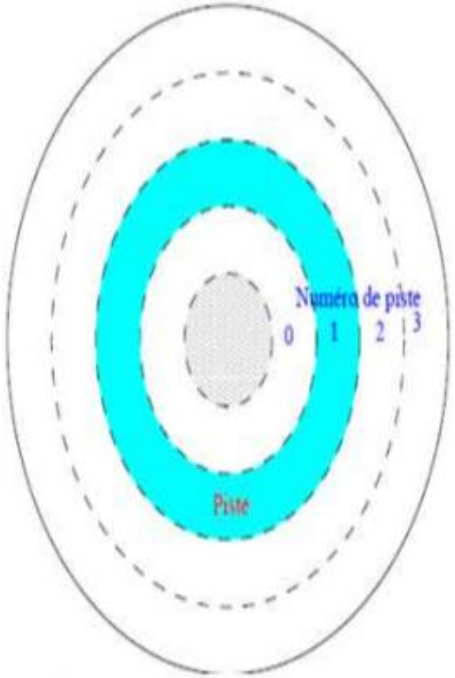
  Un cluster est constitué d’un ou plusieurs secteurs

  Ainsi, plus la taille d’un cluster est importante, moins le système d’exploitation aura d’entités à gérer.

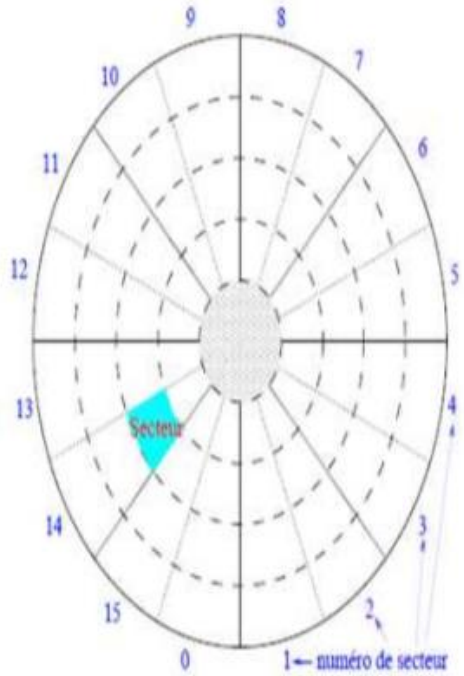
Structure d'un disque dur



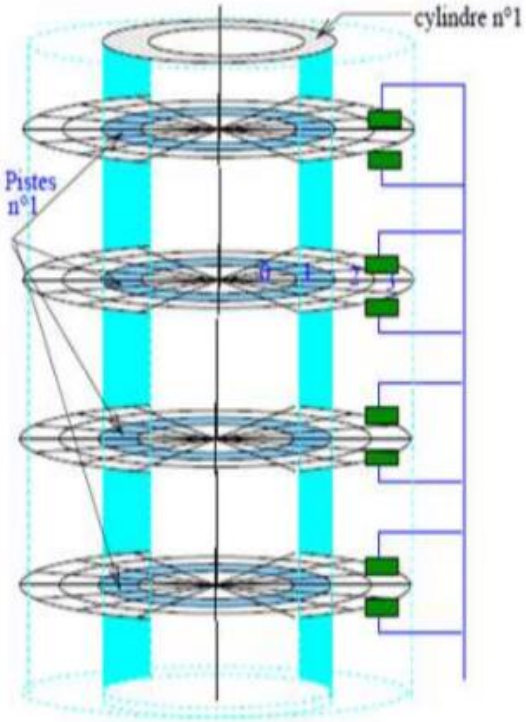
Un disque contient plusieurs plateaux



Chacun des plateaux contient plusieurs pistes



Chaque piste contient plusieurs secteurs

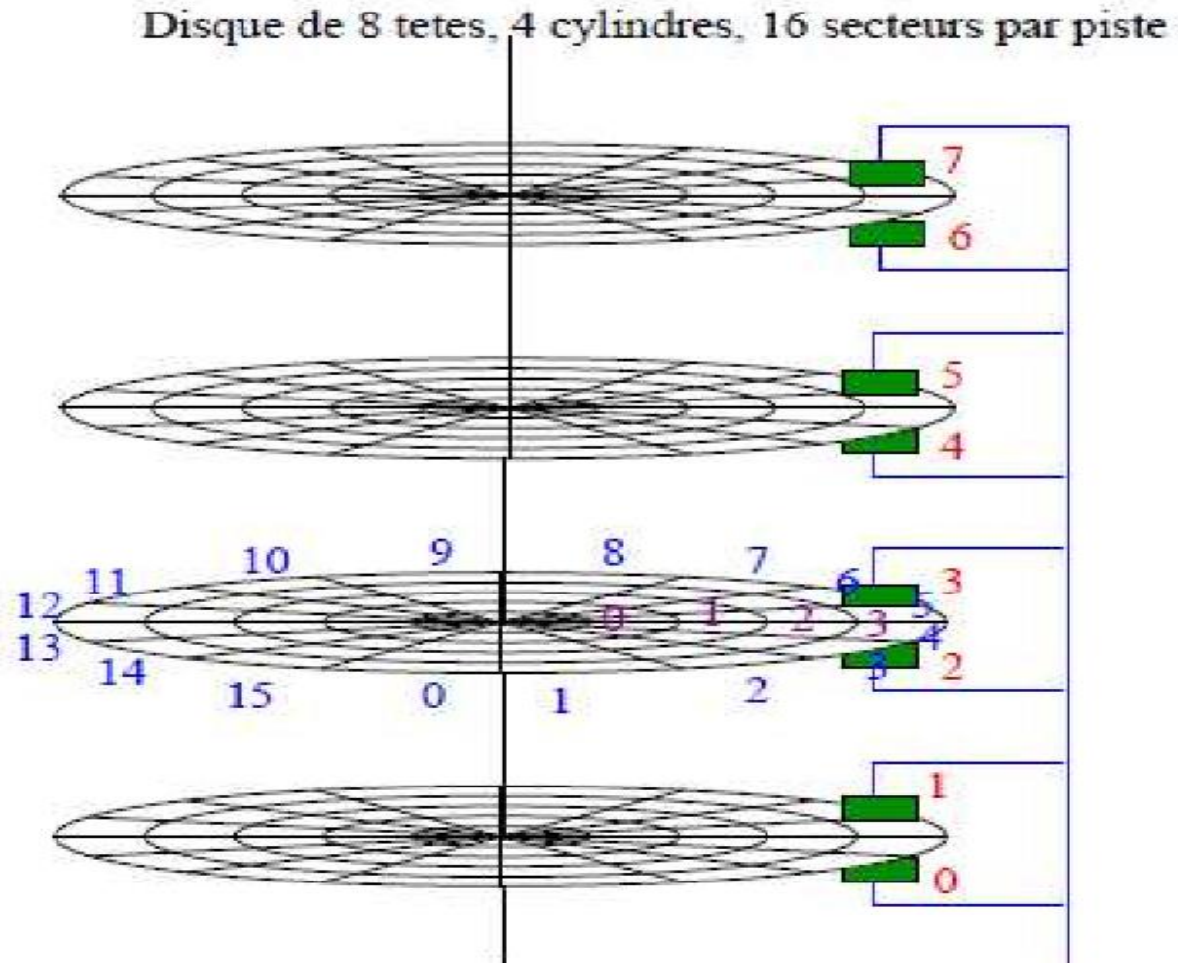


L'empilement des pistes forme un Cylindre

Géométrie du disque

- Nombre de pistes par cylindre = nombre de têtes de lecture/écriture.
- Nombre de secteurs du disque = Nombre de secteurs/piste × Nombre de pistes/cylindre × Nombre de cylindres = Nombre de secteurs/piste × Nombre de têtes de lecture/écriture × Nombre de cylindres
- Taille d'une piste = Taille d'un secteur × Nombre de secteurs/piste
- Taille d'un cylindre = Taille d'une piste × Nombre de têtes de lecture/écriture
- Taille d'un disque = Taille d'un cylindre × Nombre de cylindres par disque.

Géométrie du disque



taille secteur= 2+3=5

Le concept d'un fichier

- Un fichier est l'unité de stockage logique pour enregistrer les données de l'utilisateur : c'est l'unité d'allocation.
- Le SE établit la correspondance entre le fichier et le système binaire utilisé lors du stockage de manière transparente pour les utilisateurs.
- Un fichier peut contenir du texte, des images, des calculs, des programmes...
- 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 (le nom, l'extension, la date et l'heure de création ou de dernière modification, la taille, la protection…)
- Certains de ces attributs sont indiqués par l'utilisateur, d'autres sont complétés par le système d'exploitation.

Structure d’un fichier

- Avec la multitude des fichiers créés, le système d’exploitation a besoin d’une organisation afin de structurer ces fichiers et de pouvoir y accéder rapidement.

- Les systèmes d’exploitation modernes adoptent une structure hiérarchique des fichiers une **Arborescence** qui commence par un répertoire «racine» et regroupe des répertoires et des fichiers.

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.

Organisation de l'espace disque

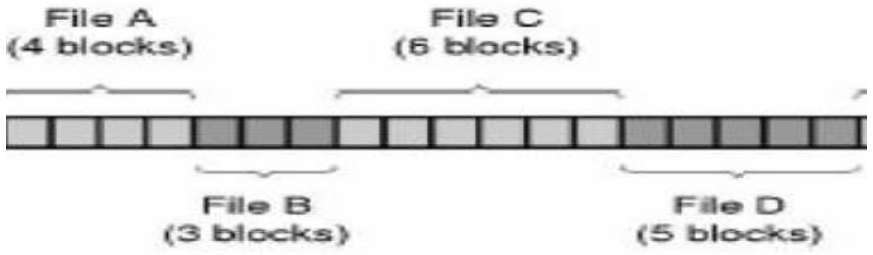
A. Allocation contiguë

- Pour chaque fichier à enregistrer, le système cherche une zone suffisamment grande pour accueillir le fichier.
- Le fichier est constitué de plusieurs blocs contigus.

☺ La rapidité d'accès (les blocs étant contigus, on limite les déplacements lecture/écriture, coûteux en temps).

☹ 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.



Organisation de l'espace disque

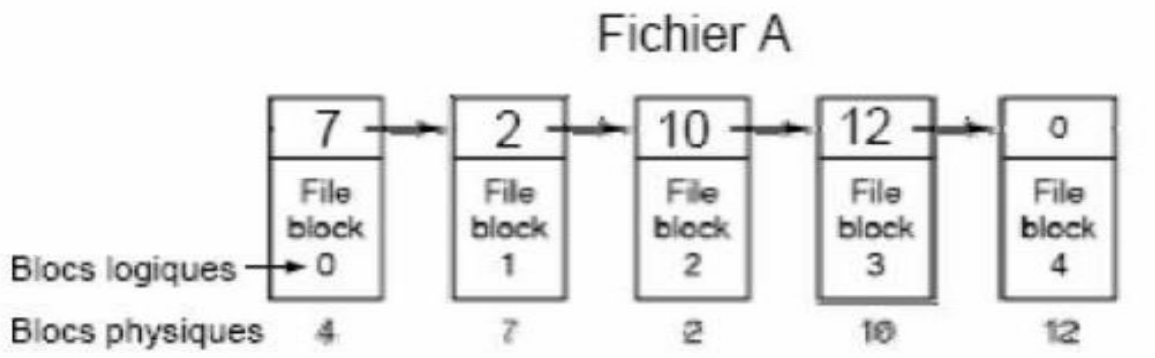
B. Allocation chainée (non contiguë)

- 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.

☹ 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.



Organisation de l'espace disque

C. Allocation chainée indexée

- Tous les inconvénients de l'allocation chaînée peuvent être résolus d'une manière simple : 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.
- Donc l'allocation chaînée indexée consiste à regrouper toutes les adresses dans une table FAT (File Allocation Table).

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).
- Cette solution est utilisée pour trouver *n* blocs contigus, elle est utilisée dans les systèmes : NTFS, ext2fs.

❖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.

Systèmes de Gestion de Fichiers

Chaque SE supporte différents systèmes de gestion de fichiers. Certains SGF sont supportés par plusieurs SE.

Système d'exploitation	Types de système de fichiers supportés
Dos	FAT16
Windows 95	FAT16
Windows 95 OSR2	FAT16, FAT32
Windows 98	FAT16, FAT32
Windows NT4	FAT, NTFS (version 4)
Windows 2000/XP	FAT, FAT16, FAT32, NTFS (versions 4 et 5)
Linux	Ext2, Ext3, ReiserFS, Linux Swap, FAT16, FAT
MacOS	HFS (Hierarchical File System), MFS (Macinto
OS/2	HPFS (High Performance File System)
SGI IRIX	XFS
FreeBSD, OpenBSD	UFS (Unix File System)
Sun Solaris	UFS (Unix File System)
IBM AIX	JFS (Journaled File System)

Systèmes de Gestion de Fichiers

Le Système de Gestion de Fichiers FAT

- Le système FAT (File Allocation Table) est considéré comme le premier système de gestion de fichiers.
- Ce système utilise une table d'allocation de fichier sous la forme d'un index ayant pour rôle de lister le contenu du disque pour enregistrer l'emplacement des fichiers.
- La table d'allocation est un tableau dont chaque cellule correspond à un bloc. Chaque cellule contient un chiffre qui permet de savoir si le bloc est utilisé par un fichier, et, le cas échéant, indique l'emplacement du prochain bloc que le fichier occupe.
- On obtient une chaîne FAT : une liste chainée de références pointant vers les différents blocs successifs, jusqu'au cluster de fin de fichier. Chaque entrée de la FAT a une longueur de 16 ou 32 bits
- Les deux premières entrées permettent de stocker des informations sur la table elle-même, tandis que les entrées suivantes permettent de référencer les blocs.
- La valeur 0000 indique que le bloc n'est pas utilisé, FFF7 permet de marquer le bloc comme défectueux pour éviter de l'utiliser, et FFFF indique la fin d'un fichier.

Systèmes de Gestion de Fichiers

❖ FAT16

Le système de fichiers FAT16 (utilisé par MS-Dos et windows 95) est un système 16 bits : il ne peut pas adresser les blocs (clusters) sur plus de16bits.

- Le nombre maximum de blocs est de 2^{16}, soit 65536 blocs.
- La taille maximale d'une partition FAT16 = le nombre maximal de clusters * la taille d'un cluster.
  - Avec des clusters d'une taille 32Ko, la taille maximale d'une partition FAT est donc de 2Go ($32\times1024\times2^{16}$).

Un fichier ne peut occuper qu'un nombre entier de clusters : si un fichier occupe plusieurs clusters, le dernier sera occupé en partie, et la place inoccupée restante sera perdue. Par conséquent plus la taille d'un cluster est réduite, moins il y a de gaspillage de place.

❖ FAT32

Le FAT 32 (Windows 98, 2000), utilise 32 bits pour les entrées de la FAT. : seuls 28 bits sont utilisés, 4 bits sont réservés.

- Le nombre maximal de clusters par partition est passé de 65 535 à 268 435 456 (2^{28}).
- La taille maximale d'une partition FAT32 est de 8To,

Une partition FAT32 peut contenir beaucoup plus de clusters qu'une partition FAT16 ➔ réduire la taille des clusters ➔ limiter le gaspillage d'espace disque.

Systèmes de Gestion de Fichiers

Le SGF NTFS (New Technology File System)

- Le système de fichiers NTFS est utilisé par Windows2000, WindowsNT, Windows XP et Windows Vista.
- Il utilise un système basé sur une structure appelée MFT (Master File Table), permettant de contenir des métadonnées (informations détaillées) sur les fichiers.
- NTFS permet l'utilisation de noms longs et il *est sensible à la casse* : il est capable de différencier des noms en majuscules des noms en minuscules.
- Il est plus *performant* : l'accès aux fichiers sur une partition NTFS est plus rapide que sur une partition de type FAT car il utilise un arbre binaire performant pour localiser les fichiers.
- Il est plus *sécurisant* : il permet de définir des attributs pour chaque fichier : droits de lecture, écriture, exécution, etc...

Systèmes de Gestion de Fichiers

Le SGF Ext3FS (Third Extented File System)

- Le SGF Ext3FS utilise la structure inode et il est utilisé sous Unix et Linux.
- Un fichier est identifié par un numéro appelé numéro d'inode ("i-noeud«)
- Un répertoire est considéré comme un fichier, identifié par un inode, contenant une liste d'inode représentant chacun un fichier.
- Les inodes sont créés lors de la création du système de fichiers.
- La quantité d'inodes (généralement déterminée lors du formatage et dépendant de la taille de la partition) indique le nombre maximum de fichiers que le système de fichiers peut contenir.
- L'inode contient la totalité des informations sur le fichier, sauf le nom.
- Les informations des inodes : • Type (fichier, répertoire...) • Droits d'accès • Identité du propriétaire et du groupe • Dates de modification (création, lecture, modification ...) • Autres (taille, liens, adresses...)

Chapitre 3

III. Gestion de la mémoire
- Gestion sans recouvrement ni pagination
- Gestion avec recouvrement ("va et vient" ou "swapping") sans pagination
- Gestion avec recouvrement, avec pagination ou segmentation

Introduction

- Pour s'exécuter, les processus ont besoin d'être présents en mémoire centrale.

☹ Conflit d'accès sur des zones communes, violation d'accès à des zones protégées, impossibilité de trouver un espace contigu pour charger tout le processus, …

- Les fonctionnalités d'un gestionnaire efficace de la mémoire sont les suivantes :
  - Connaître les parties libres de la mémoire physique.
  - Délimiter l'espace utilisateur de celui du système
  - Partager l'espace utilisateur entre les processus.
  - Allouer de la mémoire aux processus, en évitant au tant que possible le gaspillage
  - Récupérer la mémoire libérée par la terminaison d'un processus.
  - Offrir aux processus des services de la mémoire virtuelle, moyennant le ***swapping*.**

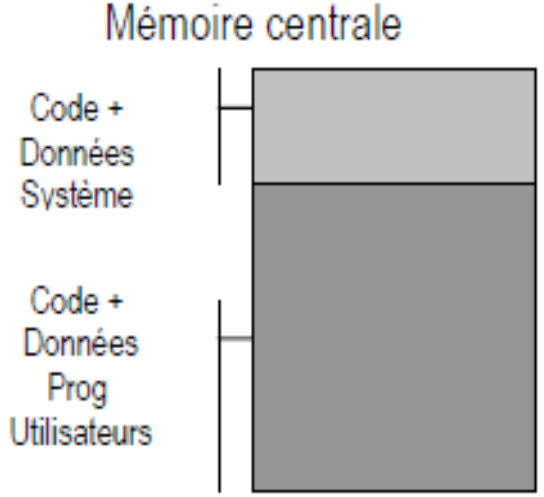
Introduction

- Dans les systèmes **mono-programmés**, un seul programme utilisateur est présent en mémoire centrale.
  - Lorsque sa taille est importante, il est possible de recourir aux méthodes de **recouvrement** à condition de découper au préalable le programme en plusieurs entités logiques (e.g. des modules compilés séparément).

- Dans les systèmes **multi-programmés**, plusieurs programmes résident simultanément en mémoire centrale.
  - La mémoire est donc divisée en zones qui peuvent être de taille fixes ou variables, et forment des **partitions**.

Allocation de la mémoire physique

- Lorsqu'un utilisateur demande l'exécution d'un programme = SE cherche à trouver une place libre suffisamment grande



- Lorsque le processus est terminé = la place mémoire doit être libérée.

- Deux stratégies d'allocation de la mémoire centrale :
  - Partition variable : un programme est un ensemble de mots continus insécables = espace d'adressage linéaire
  - Pagination / fragmentation : un programme est un ensemble de mots contigus sécables = chaque morceau peut être alloué de manière indépendante.

Multiprogrammation avec partitions Fixes

- Diviser la mémoire en *n* partitions qui peuvent être de tailles égales ou inégales.
- On appelle ceci *Mutiprogramming with a Fixed Number of Tasks* (***MFT***).
- Cette division est réalisée par le SE lors du démarrage de la machine.
- Avec MFT, le nombre et la taille des partitions sont fixés à l'avance.

Partitions fixes de tailles égales

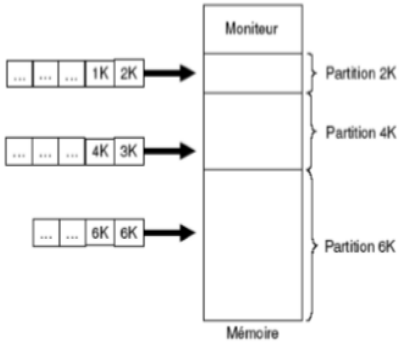
- Subdiviser la mémoire en partitions fixes de tailles égales.
- Chaque nouveau processus est placé dans une partition vide.
- Le sous système chargé de la gestion de mémoire met en place une structure des données appelée *Table des partitions* ayant pour rôle l'indication de l'état (libre ou occupée) de chaque partition en identifiant chacune soit par son adresse soit par son numéro.

Partitions fixes de tailles inégales

- La mémoire est subdivisée en partitions de tailles inégales :

1. On crée une file d'attente par partition. Chaque nouveau processus est placé dans la file d'attente de la plus petite partition qui peut le contenir.

   ☹ Un processus peut rester en attente dans une file, alors qu'une autre partition pouvant le contenir est libre.



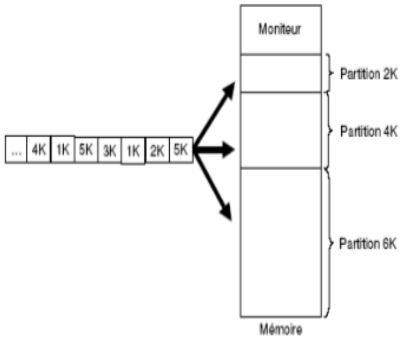
2. On crée une seule file d'attente globale :

   2.1. Dès qu'une partition se libère, on lui affecte la première tâche de la file qui peut y tenir.

   ☹ On peut affecter une partition de grande taille à une petite tâche et perdre beaucoup de place.

   2.2. Dès qu'une partition se libère, on lui affecte la plus grande tâche qui peut y tenir.

   ☹ On pénalise les processus de petite taille.



Partitions fixes de tailles inégales

- Avec le schéma MFT, on fait face au problème de fragmentation, puisqu'on génère deux types d'espaces de mémoire non utilisés :
  - **Fragmentation interne**: c'est la partie d'une partition non utilisée par un processus.

    Mémoire allouée peut-être légèrement plus large que la mémoire demandée; cette différence de taille est une mémoire interne à un bloc alloué, mais non utilisée

  - **Fragmentation externe** : engendrée par les partitions qui ne sont pas utilisées.

    somme des blocs peut satisfaire une requête, mais non contigus

Multiprogrammation avec partitions Variables

- Le problème principal avec MFT est la détermination optimale des tailles des partitions pour minimiser la fragmentation externe et interne

Exemple : Une mémoire de 120Ko, des processus de taille moins de 20Ko, un processus de 80Ko (une fois par jour) ➔ Fragmentation Externe ou Fragmentation Interne de 60 Ko

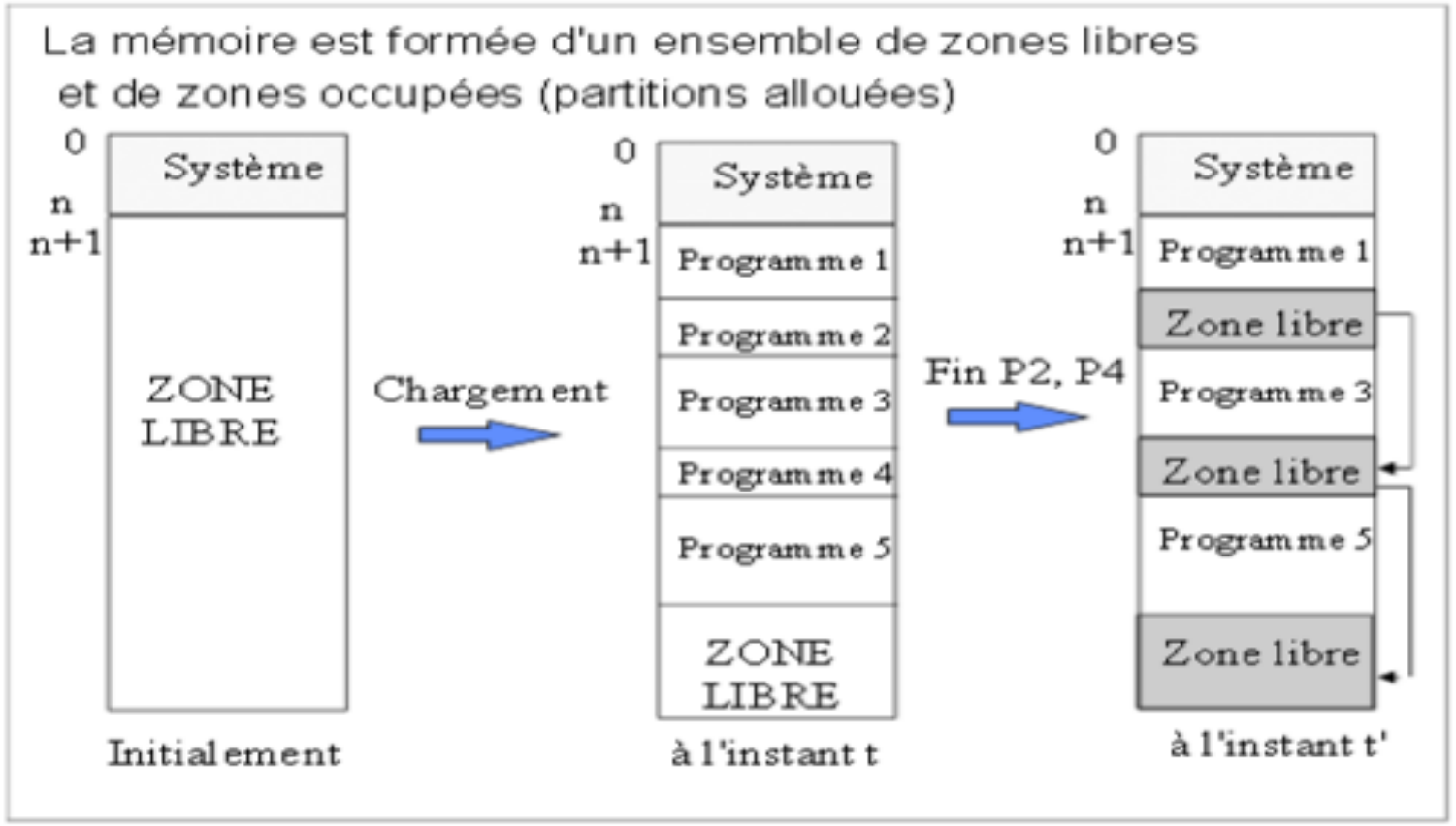
➢ La solution consiste à faire que les zones de mémoire varient dynamiquement.

- Cette technique s'appelle *Mutiprogramming with Variable Tasks* (***MVT***).

Multiprogrammation avec partitions Variables

- Au lancement du système, on crée une seule zone libre de taille maximale.
- Lorsqu'on charge un programme, on le place dans une zone libre suffisante, et on lui alloue exactement la mémoire nécessaire. Le reste devient une nouvelle zone libre.
- Lorsqu'un programme s'achève, sa partition redevient libre, et peut éventuellement grossir une zone libre voisine.
- Finalement, la mémoire centrale se retrouve constituée d'un ensemble de zones allouées et de zones libres réparties dans toute la mémoire

Multiprogrammation avec partitions Variables



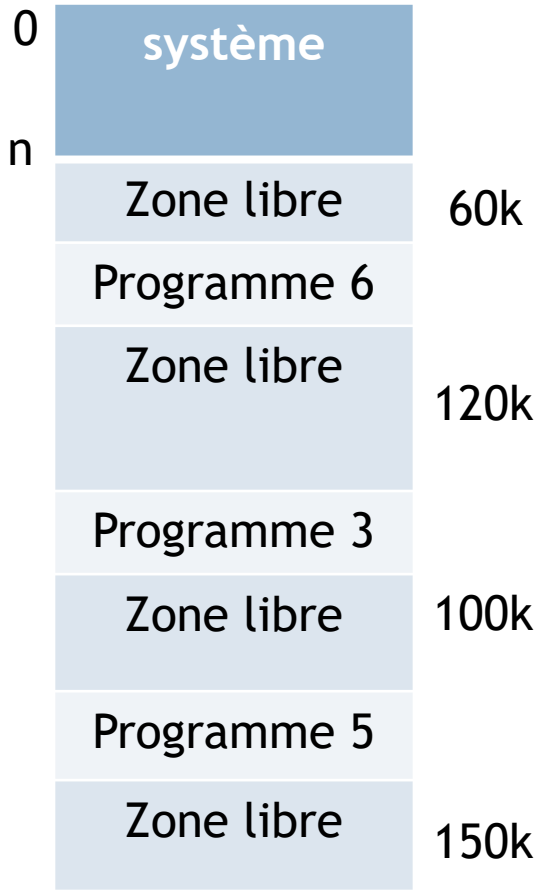
Algorithmes d'allocation de la mémoire

- Les algorithmes d'allocation de la mémoire pour un processus en attente de son chargement se déclinent en :
  - **First Fit** : on alloue la première zone libre qui peut loger le processus. Il se forme alors un trou qu'il faudra ajouter à la liste des zones libres.
  - **Best Fit** : on alloue la zone libre la plus petite qui soit à condition de pouvoir y loger le processus. Le but ici est d'éviter de couper inutilement une grande zone.

    Le trou résiduel est de taille inférieure à celle d'une miette =>

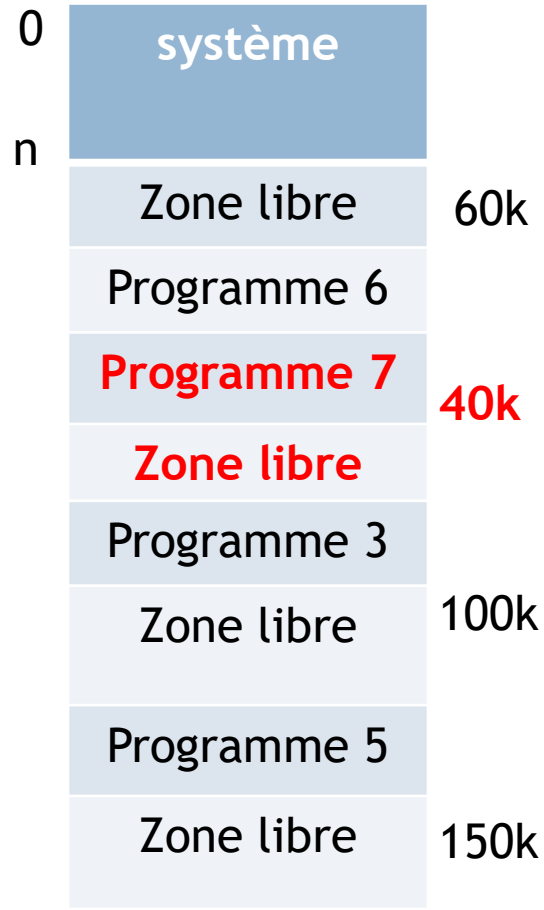
    ☹ Fragmentation interne => nécessité de **compactage**
  - **Worst fit** : on alloue la zone libre la plus grande qui existe et qui peut loger le processus. Le but ici est de minimiser la fragmentation interne car on espère que le trou restant après allocation est suffisamment large pour accueillir d'autres processus.

Allocation First Fit



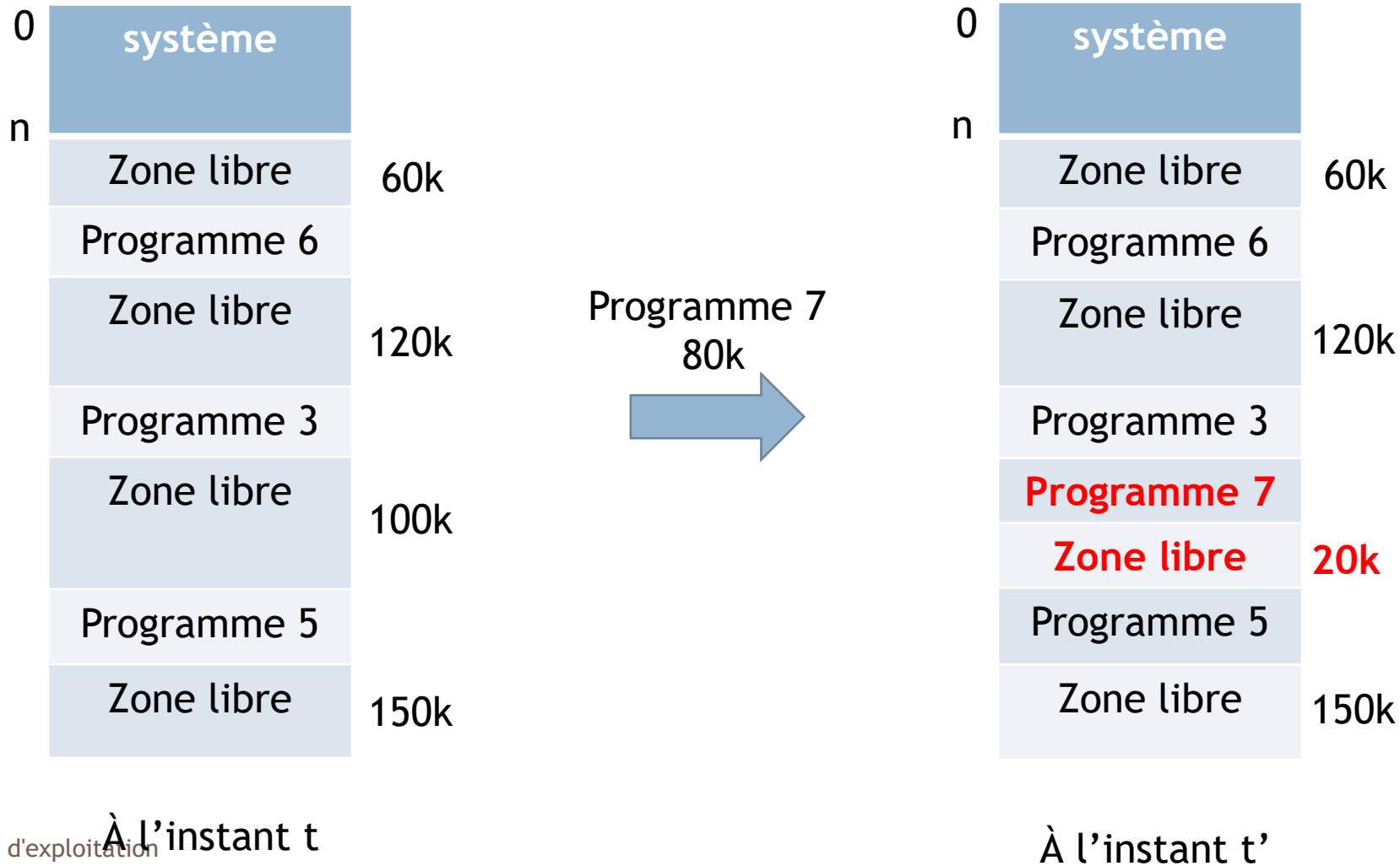
À l'instant t

Programme 7
80k



À l'instant t'

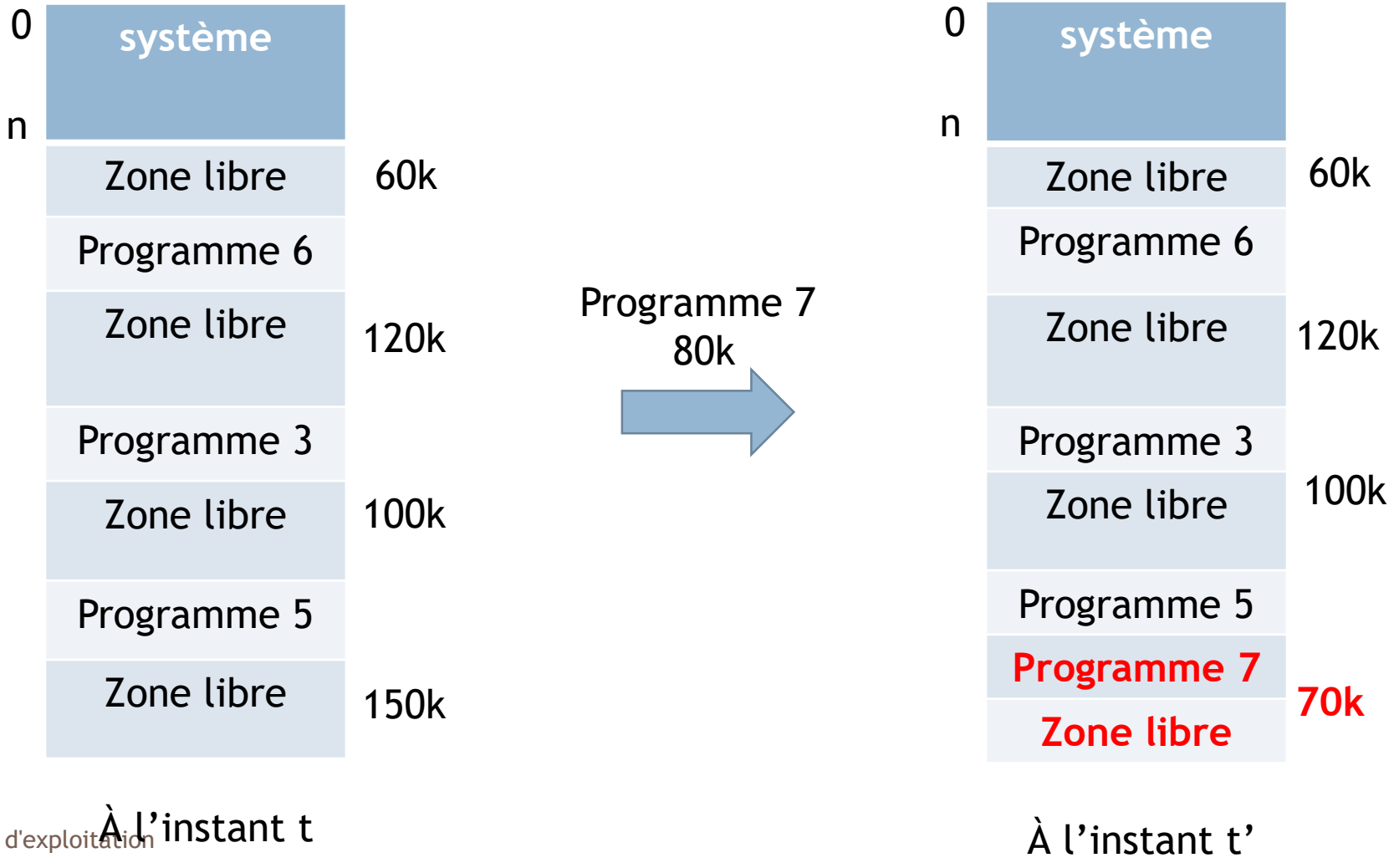
Allocation Best Fit



À l'instant t

À l'instant t'

Allocation Worst Fit



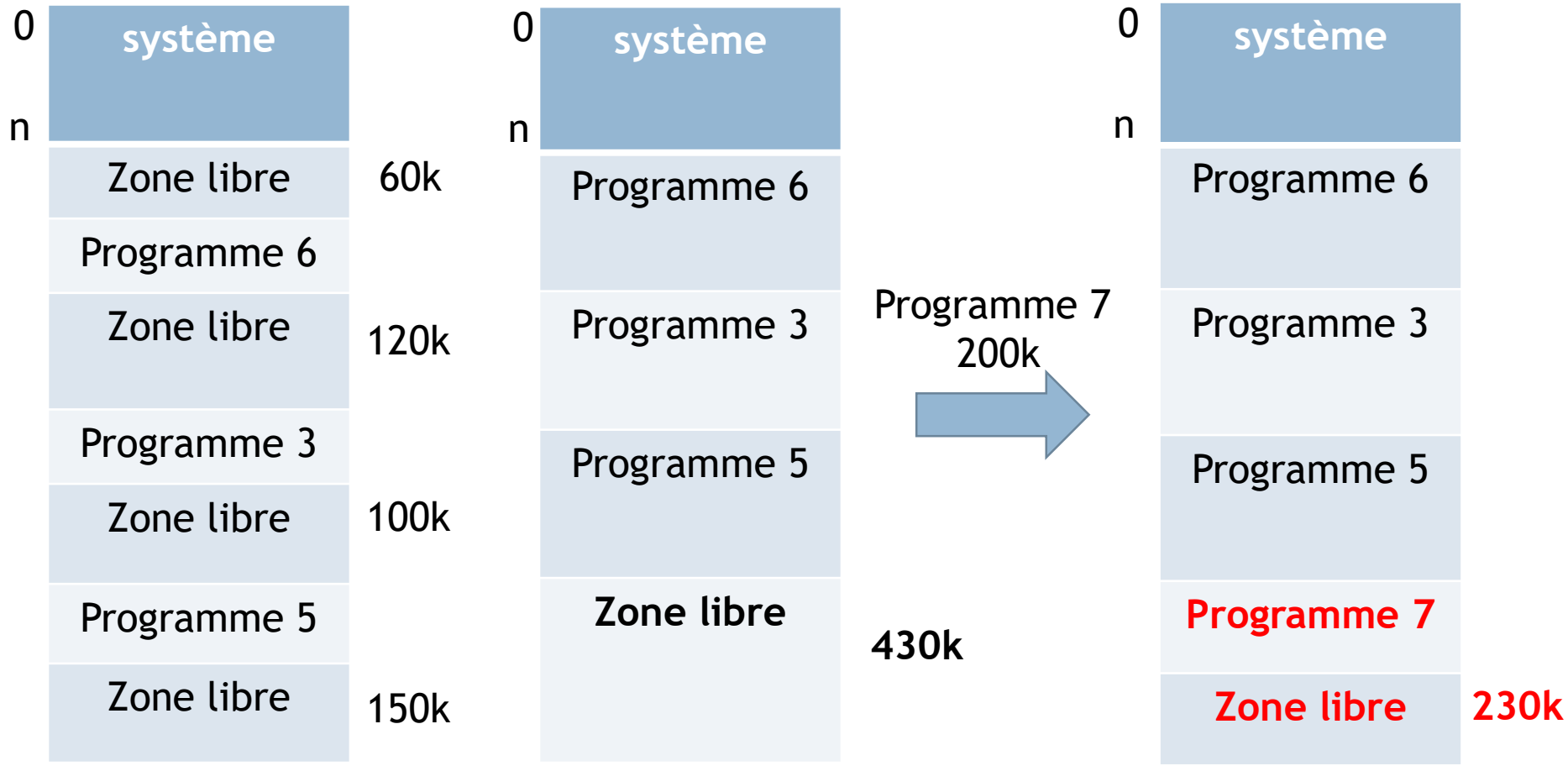
À l'instant t

À l'instant t'

Compactage de la mémoire

- Aucune de ces stratégies ne résout correctement le problème de l'éventuel émiettement de la mémoire(fragmentation). On préconise alors deux solutions ad hoc :
  - ✓ la **collision** des trous dès leur formation,
  - ✓ le ramassage des miettes (**compactage** des trous) en un seul bloc libre lorsque cela s'impose.

Compactage de la mémoire



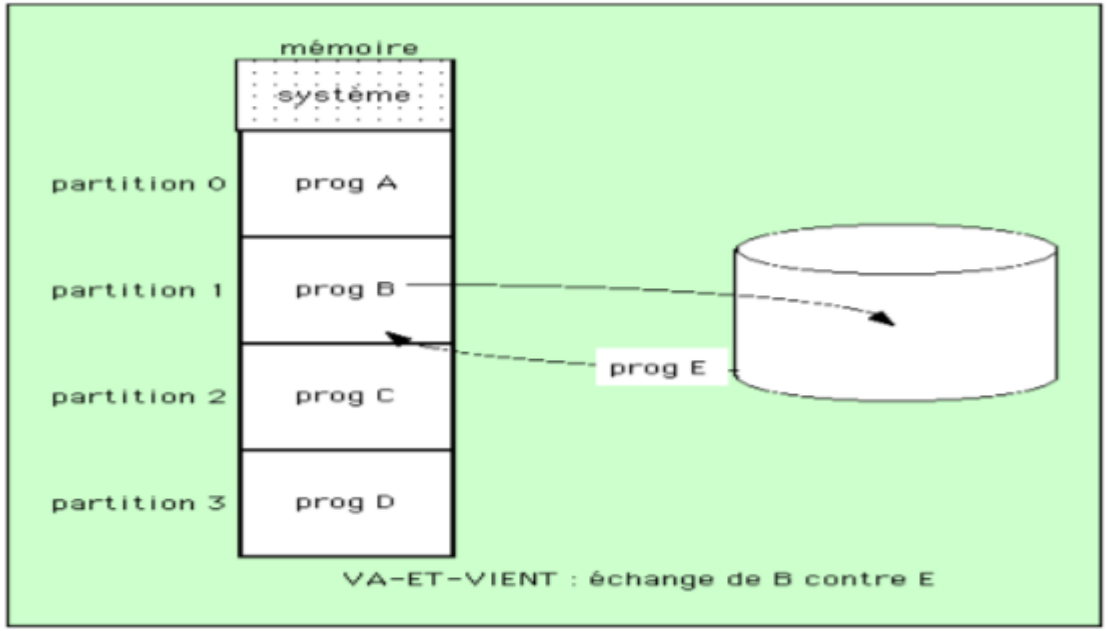
Gestion avec recouvrement ("va et vient" ou "swapping") sans pagination

Gestion avec recouvrement, sans pagination

- Dans le cas de la multiprogrammation sans va-et-vient (entre la mémoire et le disque), un processus chargé en mémoire y séjournera jusqu'à ce qu'il se termine
- Puisque la mémoire ne peut pas contenir tous les processus, il est préférable d'utiliser la multiprogrammation avec va et vient qui consiste à placer quelques processus inactifs sur le disque (mémoire de réserve: swap area ou backing store), en utilisant le va-et-vient.
- Il faut, ultérieurement, ramener ces processus en mémoire principale pour leur permettre de poursuivre leurs exécutions.
- **Le mouvement des processus entre la mémoire principale et le disque est appelé va et vient (swapping).**
- Remarque : Le temps de swapping est trop grand comparé au temps de l'UCT c'est pourquoi il faut que chaque processus qui obtient le contrôle de l'UCT s'exécute pour un temps supérieur au temps de Swap.

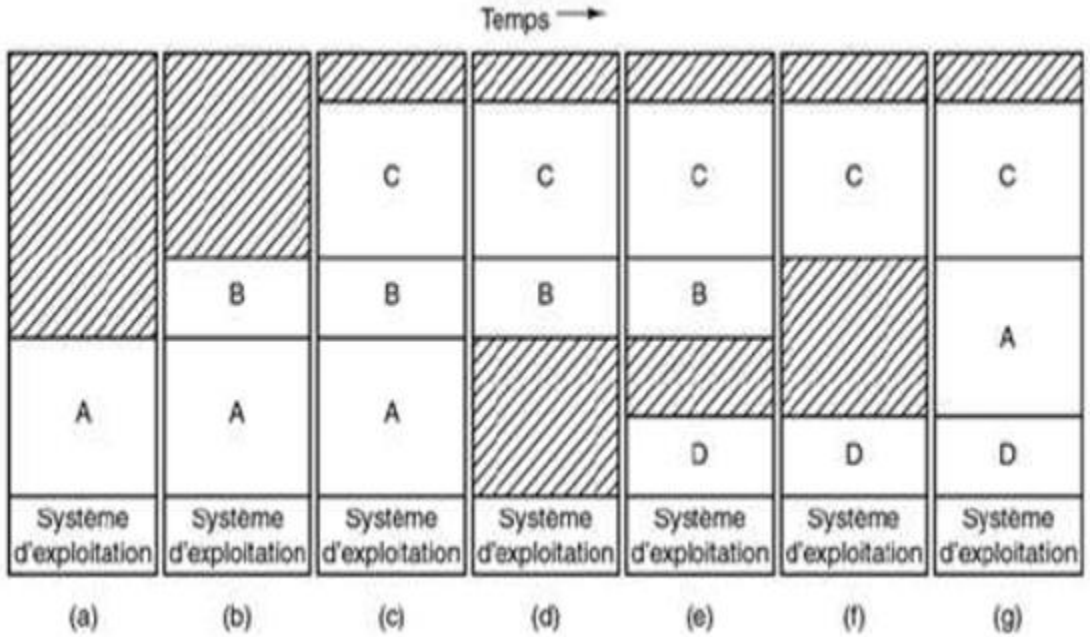
Gestion avec recouvrement, sans pagination

- Par exemple, supposons toutes les partitions de même taille. On peut exécuter 5 programmes simultanément dans 4 partitions. à chaque instant, l'un des 5 programmes se trouve sur disque. Pour exécuter le programme E, il faut choisir parmi A, B, C et D la victime, soit B; recopier B sur disque dans une zone prévue pour cela, puis mettre E à l'emplacement mémoire qu'occupait B.



Gestion avec recouvrement, sans pagination

- A un retour de recouvrement (swap), le gestionnaire doit permettre de loger le nouveau processus à implanter en MC, avec le choix entre les trois algorithmes d'allocation



Gestion avec recouvrement, sans pagination

- ☺ Le système de va-et-vient, permet de pallier le manque de mémoire nécessaire à plusieurs processus
- ☹ Il n'autorise pas l'exécution de programmes de taille supérieure à celle de la mémoire centrale.
- => Le programme devrait être divisé en segments, qui se charge un à un en mémoire.
- Le segment 0 s'exécute en premier. Lorsqu'il se termine, il appelle un autre segment de recouvrement.
- ☹ Cette solution n'est pas intéressante, car c'est le programmeur qui doit se charger du découpage de son programme en segments de recouvrement.
- Une autre solution consiste à décharger le programmeur de cette tâche. Le système d'exploitation charge en mémoire les parties du programme qui sont utilisées par le processeur.
- Le reste du programme est stocké sur le disque (mémoire virtuelle).

=> On peut ainsi exécuter des programmes dont la taille dépasse celle de la mémoire.

Gestion avec recouvrement, avec pagination ou segmentation

Gestion avec recouvrement, avec pagination ou segmentation

La pagination

La mémoire virtuelle

- En cas d'insuffisance de mémoire physique, le système d'exploitation peut créer une zone mémoire sur le disque dur, appelée «*mémoire virtuelle*».
- La **mémoire virtuelle** permet de faire fonctionner des applications nécessitant plus de mémoire qu'il n'y a de mémoire vive disponible sur le système.
- Un fichier est créé sur le disque dur (ou en mémoire secondaire) et devient un fichier d'échange nommé **swap**.
- Lorsqu'un processus a besoin de laisser des informations en mémoire et qu'il n'y a plus de place sur la mémoire centrale, les données concernant un autre processus moins prioritaire vont être transférées dans le fichier swap.
- Ensuite, le programme prioritaire utilise la place libérée sur la mémoire vive. En contrepartie la mémoire secondaire est beaucoup plus lente puisque il y aurait des va-et-vient

Mémoire centrale non contiguë

- La mémoire est décomposée en blocs :
  - Si la taille de tous les blocs est constante, le bloc est appelé page et la technique est appelée **Pagination**.
  - Si la taille des blocs n'est pas constante, le bloc est appelé segment et la technique est dite **Segmentation**.

- Les deux problèmes de partitions variables :
  - Une zone libre contiguë
  - Fragmentation externe

Solution :
- programme = ensemble sécable
- Le programme peut être divisé en parties. Chacune a être allouée d'une manière aléatoire en mémoire.

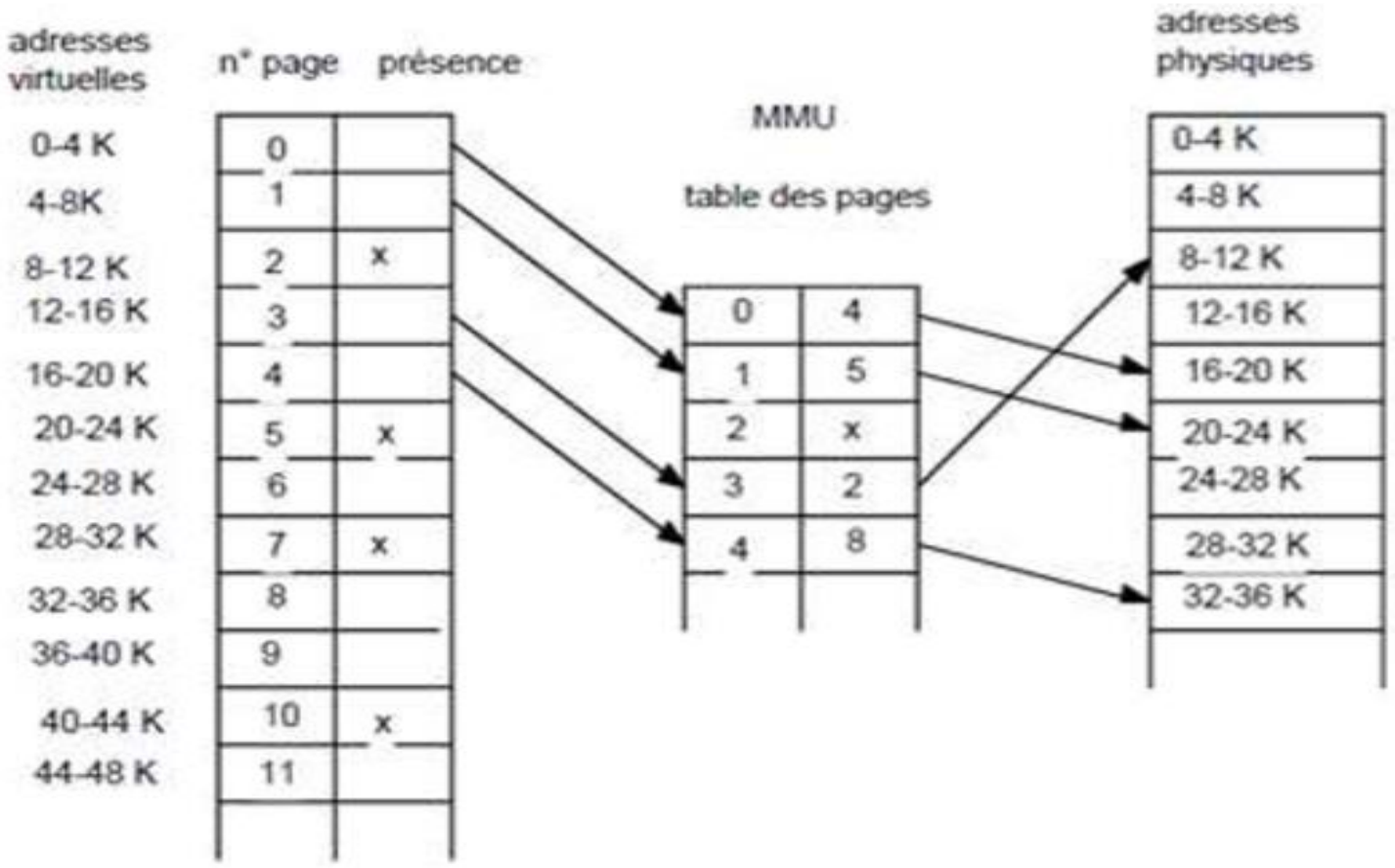
Pagination

- Découper le programme en morceaux tous de même taille (Pages). Ce découpage est sans rapport avec le découpage logique du programme. Il est donc fait de manière totalement transparente pour le programmeur, le compilateur et l'utilisateur.
- La mémoire centrale est découpée en unités physiques de même taille appelées cases (taille d'une case = taille d'une page).

- Son principe : chargement du programme en mémoire
  - Placer les pages dans les cases disponibles
  - Connaitre les cases disponibles : **table des cases**
  - Table des cases : tableau indiquant l'état (libre, occupé) des cases de la mémoire :

  Le mécanisme DAT de la MMU (Memory Management Unit ou unité de gestion de la mémoire) assure la conversion des adresses virtuelles en adresses physiques, en consultant une table des pages pour connaître le numéro du cadre qui contient la page recherchée. L'adresse physique obtenue est le couple (numéro de cadre, déplacement).

- Une page est mappée (chargée) si elle est physiquement présente en mémoire.



Pagination

- L'adresse virtuelle 12292 correspond à un déplacement de 4 octets dans la page virtuelle 3 (car 12292=12288 + 4 et 12288 = 12∗1024).
- La page virtuelle 3 correspond à la page physique 2.
- L'adresse physique correspond donc à un déplacement de 4 octets dans la page physique 2, soit : (8∗1024) + 4 = 8196.
- Par contre, la page virtuelle 2 n'est pas mappée.
- Une adresse virtuelle comprise entre 8192 et 12287 donnera lieu à un défaut de page.

Pagination

- En transformant une adresse virtuelle en une adresse physique. Trois situations sont possibles :
  - la page adressée est présente en mémoire centrale et le déplacement explicité est dans les limites de la page : accès au contenu de mot mémoire désiré
  - la page adressée est présente en mémoire centrale mais le déplacement explicité est en dehors des limites de la page : erreur fatale de violation mémoire
  - la page adressée n'est pas présente en mémoire centrale : il y a un déroutement pour **défaut de page.**

Pagination

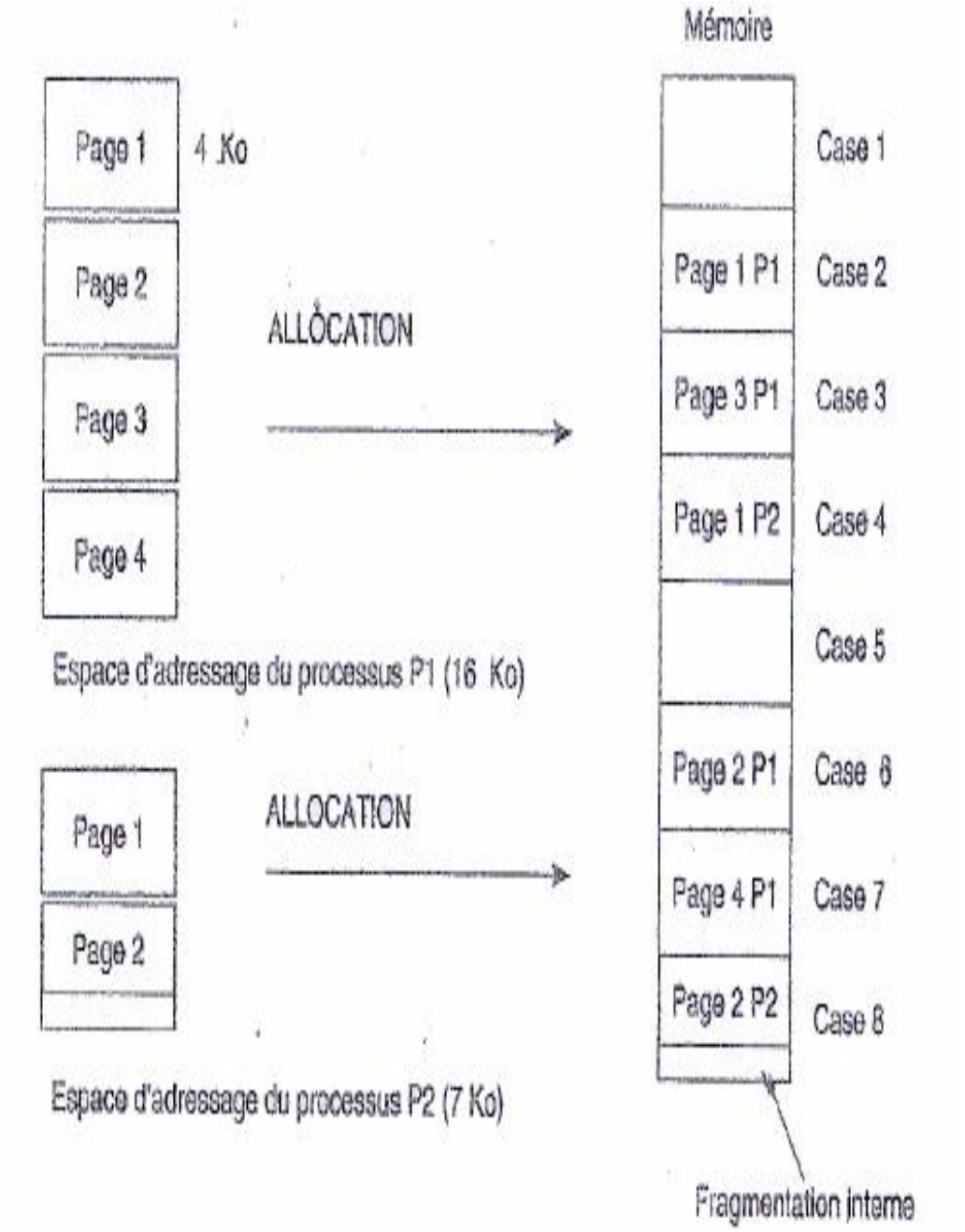
- Il y a défaut de page quand il y a un accès à une adresse virtuelle correspondant à une page non mappée. En cas de défaut de page, un déroutement se produit et le processeur est rendu au SE.
- Le système doit alors effectuer les opérations suivantes :
  - Trouver une page réelle libre
  - Charger la page manquante
  - Modifier la table de présence et la table de pages
  - Relancer l'instruction
  - S'il n'y a pas de page réelle libre, en libérer une : en déplaçant une page de la mémoire réelle vers la mémoire virtuelle.
- Le choix est très important. Il influe sur le rendement du système.

Pagination

1. **Algorithme optimal** : Déplacer la page dont l'occurrence de la prochaine utilisation est la plus éloignée dans le futur. => Implémentation difficile voire impossible.
2. **FIFO (first in, first out)** : L'idée est de déplacer la page la plus vieille, en espérant qu'elle ne sera pas demandée rapidement après son déplacement. Pour son implémentation, il faudra inscrire pour chaque, l'heure de son entrée dans la mémoire réelle.
3. **LRU (least recently used)** : Retirer la page la moins récemment référencée. Indexer chaque page par le temps écoulé depuis sa dernière référence et constituer une liste chaînée des pages par ordre décroissant de temps depuis la dernière référence. L'algorithme est stable. Mais il nécessite une gestion coûteuse de la liste qui est modifiée à chaque accès à une page.
4. **NRU (not recently used)** : Retirer la page la moins fréquemment utilisée. Il nécessite d'enregistrer le nombre d'accès à la page par unité de temps.

Conséquence :

Fragmentation interne



	Page	Processus
Case 1	-1 (libre)	
Case 2	1	P1
Case 3	3	P1
Case 4	1	P2
Case 5	-1 (libre)	
Case 6	2	P1
Case 7	4	P1
Case 8	2	P2

Table des cases

Pagination

- La correspondance entre les pages logiques et les pages physiques : **table des pages**

- **Caractéristiques :**
  - Une table par processus
  - Tableau contenant autant d'entrées (n° de la page, case physique) que le processus a de pages

Exemple

Table de pages « processus 2 »	
Page 1	Case 4
Page 2	Case 8

Table de pages « processus 1 »	
Page 1	Case 2
Page 2	Case 6
Page 3	Case 3
Page 4	Case 7

Pagination

Conversion de l'adresse paginée en adresse physique

- @logique = @paginée
- @paginée = n°de la page (p) + déplacement relatif p/r au début de la page (d)
- page = case

Gestion avec recouvrement, avec pagination ou segmentation

La segmentation

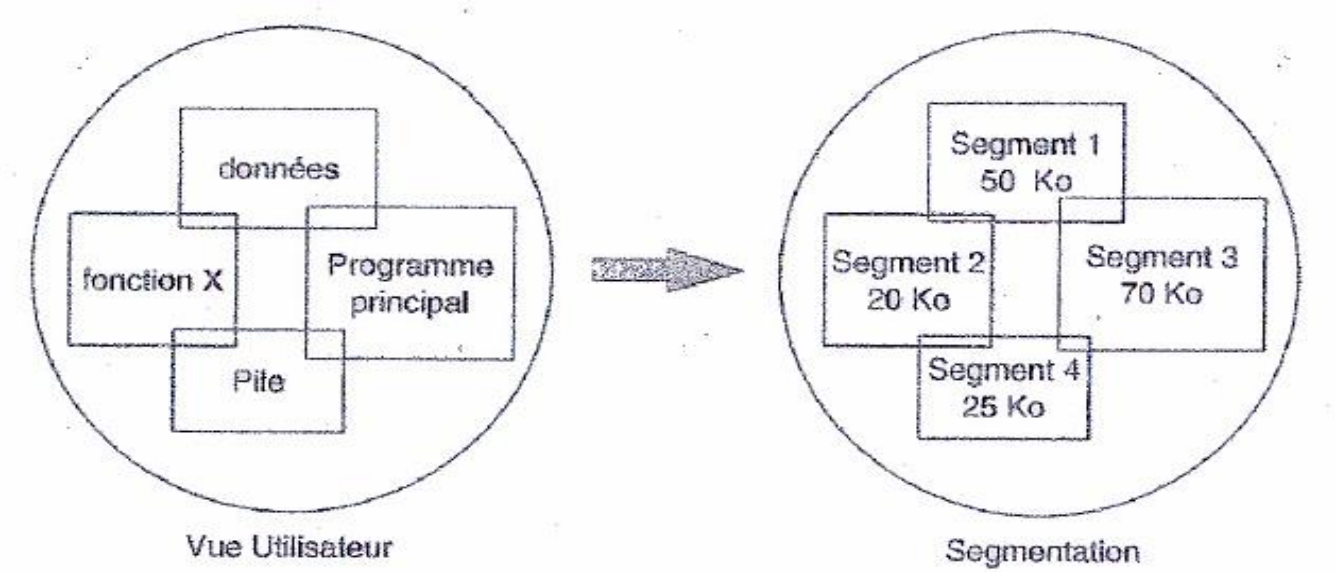
La segmentation

- La segmentation est basée sur le fait que les programmes à exécuter sont préalablement découpés en entités logiques (des modules) qu'on pourrait charger dans des segments de mémoire égales à leur taille respective.

- Les systèmes segmentés fonctionnent suivant un mode très voisin à celui des systèmes paginés, la **table des segments** remplaçant la table des pages.

La segmentation

- Ensemble d'emplacements en mémoire non sécables
- A la différence des pages, les segments d'un même processus peuvent être de tailles différentes.



La segmentation

Conversion de l'adresse segmentée en adresse physique

- @logique = @segmentée
- @segmentée = n° du segment (s) + déplacement relatif p/r au début du segment(d)

Exercice : segmentation