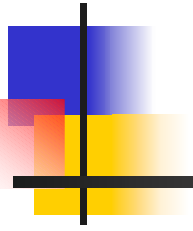
Chapitre V : La gestion de la mémoire



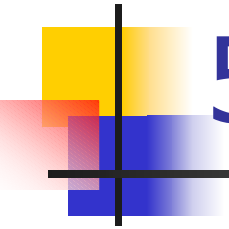
- **Hiérarchie de mémoires**
- **Objectifs**
- **Méthodes d'allocation**
- **Simulation de mémoire virtuelle**
- **Le mapping**

Introduction

- Plusieurs dizaines de processus doivent se partager une mémoire commune la RAM (mémoire centrale)
- La capacité de la mémoire centrale est restreinte (64Mo pour un ordinateur de bureau)
- Si les processus sont nombreux, ils vont occuper plus de place que ne peut leur offrir la mémoire centrale
- La mémoire secondaire (disque dur) est beaucoup plus importante (20Go) – 200 fois la mémoire centrale
- Le SE propose un mécanisme de **mémoire virtuelle** qui consiste à utiliser le disque dur pour simuler de la mémoire réelle

5.1 Hiérarchie de mémoires

- Les types de mémoires susceptibles de stocker des programmes ou des données sont multiples
- Du point de vue du SE, trois caractéristiques sont importantes :
  - Vitesse d'accès
  - Coût
  - Capacité de stockage
- La capacité de stockage est d'autant plus réduite que la vitesse d'accès et les coûts sont élevés



5.1 Hiérarchie de mémoires

5.1.1 Mémoire volatile

- Construite à partir de circuits intégrés
- Très rapide, mais coût élevé
- Au même titre que le processeur elle est considéré comme une ressource essentielle (critique, rare)
- Les deux types de mémoires volatiles sont :
  - Les registres du processeur
  - La mémoire central (RAM)

5.1 Hiérarchie de mémoires

- Pour qu'un programme fonctionne il doit être placé en mémoire centrale
- Le CPU
  - Transfert des informations de la mémoire centrale vers ses registres,
  - Réalise des opérations (additions, etc.)
  - Range les résultats dans la mémoire centrale
- Les registres du CPU peuvent contenir des adresses ou des données (cf chapitre 1)
- Leur nombre est faible mais leur accès est très rapide (c'est le type de mémoire le plus rapide)

5.1 Hiérarchie de mémoires

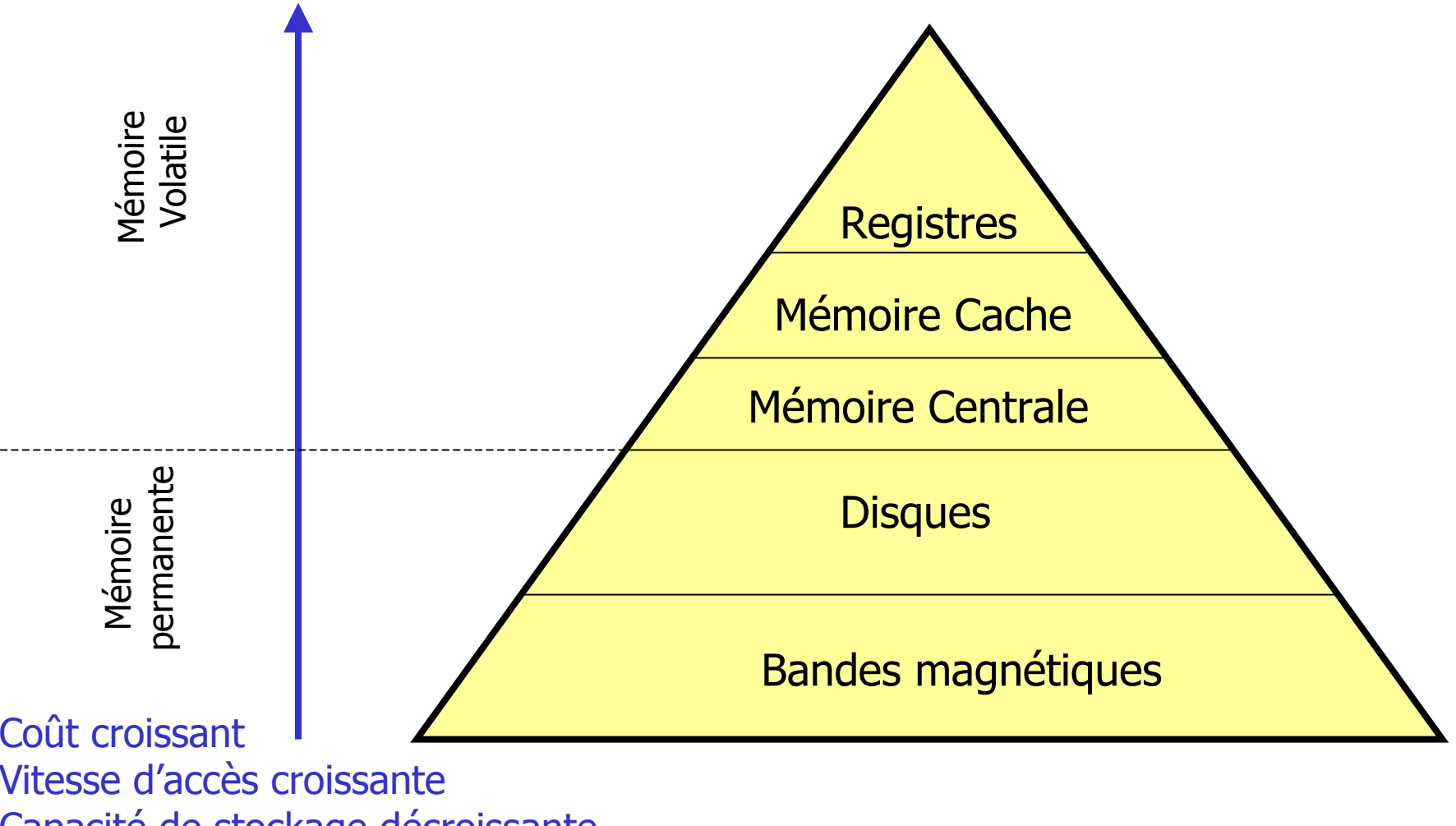
- La mémoire centrale et les registres du CPU ne travaillent pas à la même vitesse
- Pour servir d'intermédiaire entre les registres et la mémoire centrale on ajoute une **mémoire cache**
- Elle fonctionne comme un tampon directement accessible par le processeur avec un temps d'accès très faible
- Une mémoire cache est constituée de registres associatifs = une clé, une valeur
- La clé correspond à l'adresse mémoire de l'emplacement qui est caché

5.1 Hiérarchie de mémoires

5.1.2 Mémoire permanente

- Elle utilise des supports magnétiques (disques, bandes, disques optiques)
- Les temps d'accès sont beaucoup plus long mais le coût est moins élevé
- On parle également de mémoire de masse, mémoire secondaire ou auxiliaire

5.1 Hiérarchie de mémoires



5.1 Hiérarchie de mémoires

Type de mémoire	*Taille (octets)*	*Temps d'accès en secondes*	*Coût relatif par bit*
Cache	256 – 4096 Ko	3-20ns	1000
Mémoire Centrale	64Mo – 4Go	60ns	100
Disque	1Go – 75Go	4ms	1
Bandes	250Mo – 300Go	-	0,5

5.2 Les objectifs

- Le mécanisme de gestion de la mémoire par le SE a pour objectifs :
  - L'**organisation** de la mémoire en zone
  - La gestion de la mémoire (**allocation, libération**)
  - La **protection** des processus et des segments de processus
  - La **simulation** d'une mémoire de grande taille (supérieure à la mémoire physique)
  - La **transparence** des accès (quelque soit le type de mémoire physique ou virtuelle)

5.2 Les objectifs

5.2.1 Organisation de la mémoire

- Description indépendante du support
- Un espace mémoire est considéré comme un type abstrait
  - Constitué d'un ensemble de mots
  - Chacun ayant sa propre adresse (localisation)
  - Pouvant contenir des informations d'une taille fixée

5.2 Les objectifs

- La première fonction assurée par le SE est la structuration de la mémoire
- La mémoire est divisée en **zones** de taille fixe ou variable et pouvant contenir des programmes
- Une zone de taille N de la mémoire est un sous ensemble des N mots ayant des adresses **consécutives**
- Le SE doit garantir l'intégrité de ces zones en interdisant les accès non autorisés
- Le contenu de certaines zones doit pouvoir être partagé

5.2 Les objectifs

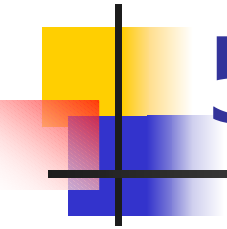
5.2.2 Gestion de la mémoire

- Le SE attribut certaines zones à plusieurs utilisateurs
- Il utilise des stratégies d'**allocation** et de **libération**
- Les opérations associées à la mémoire sont :
  - **Opération sur les mots :** lecture ou écriture
  - **Opérations sur les zones :** allocations ou libération
- Dans certains cas on ajoute les opérations :
  - Extension ou diminution de zones
  - Division d'une zone (partition)

5.3 Méthodes d'allocation

- Pour un espace donné on peut choisir deux modes d'allocation :
  - **Allocation contiguë** consiste à placer la totalité d'un programme à des adresses consécutives
  - **Allocation non contiguë** consiste à fractionner le programme et à placer les différents fragments à des adresses dispersées

5.3 Méthodes d'allocation



- On appelle **bloc** un groupe d'un nombre fixe de mots (16,256,1024, 4096,…)
- Dans la pratique, la taille des blocs est fixée par les caractéristiques du matériel
- Dans notre modèle de gestion de la mémoire, les zones allouées comportent toujours un nombre entier de blocs

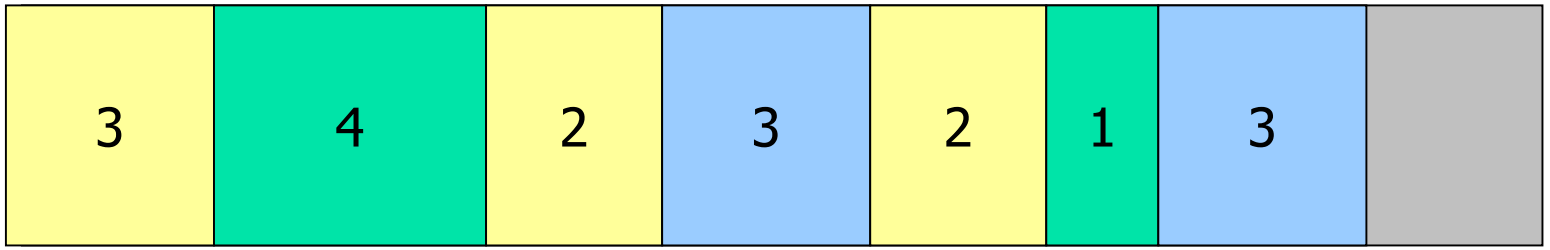
5.3 Méthodes d'allocation

5.3.1 Allocation contiguë

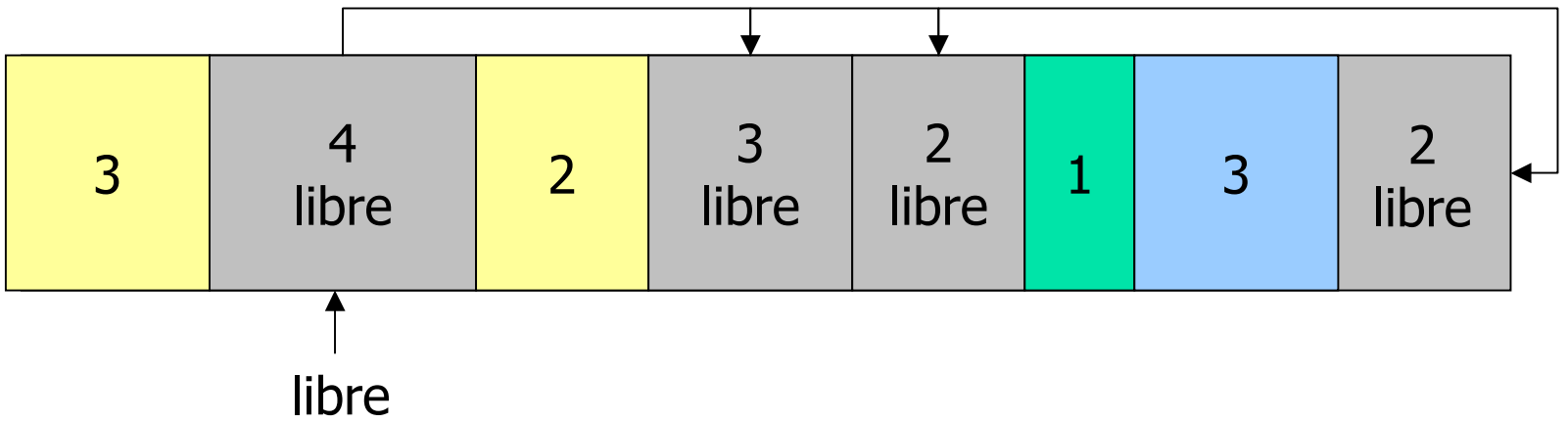
- L'espace mémoire est divisé en plusieurs zones de tailles variables qui sont soit :
  - **Allouées** pour un programme
  - **Vides (libre)**
- Le SE maintient en permanence la liste des zones vides (libres)
- Quand une zone est allouée, elle est prélevée à l'extrémité d'une zone libre

5.3 Méthodes d'allocation

Mémoire de 20 blocs après allocations successives



Libération de trois zones qui étaient allouées

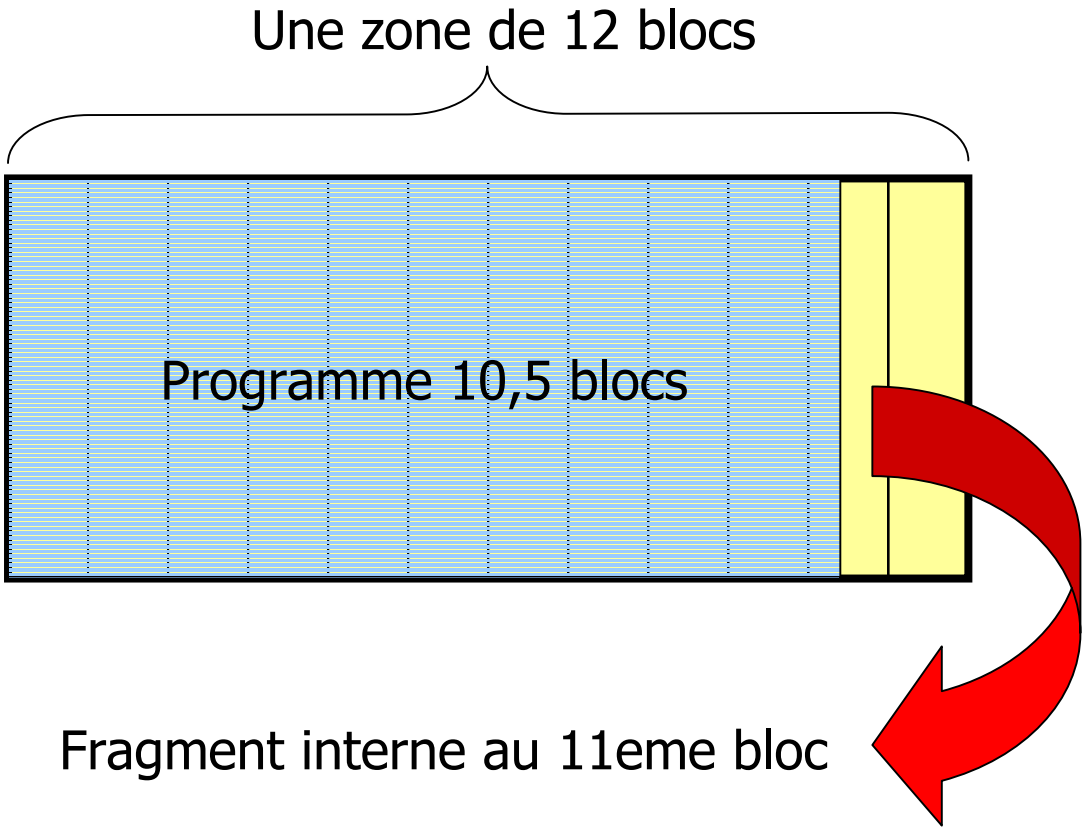


5.3 Méthodes d'allocation

- **Problème :** cette méthode laisse inutilisées certaines zones de la mémoire (fragmentation)
- **Fragmentation interne :**
  - Lorsque la taille d'un bloc est supérieure à un mot, l'ensemble des informations utilisées n'est pas nécessairement égale à la taille de la zone (cet ensemble n'est pas utilisé dans sont intégralité) ce qui produit un fragment interne au bloc

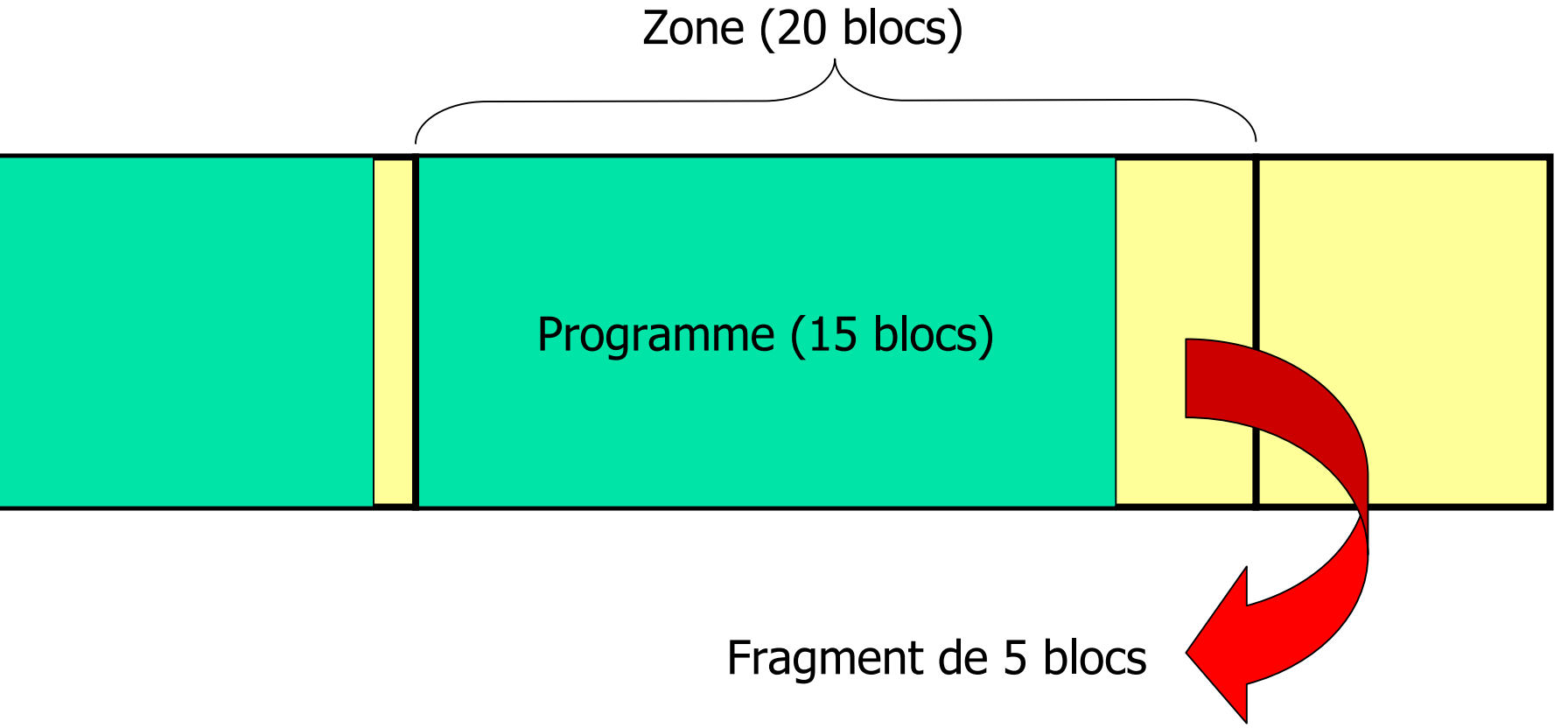
5.3 Méthodes d'allocation

- Fragmentation interne



5.3 Méthodes d'allocation

- Exemple de fragmentation externe :

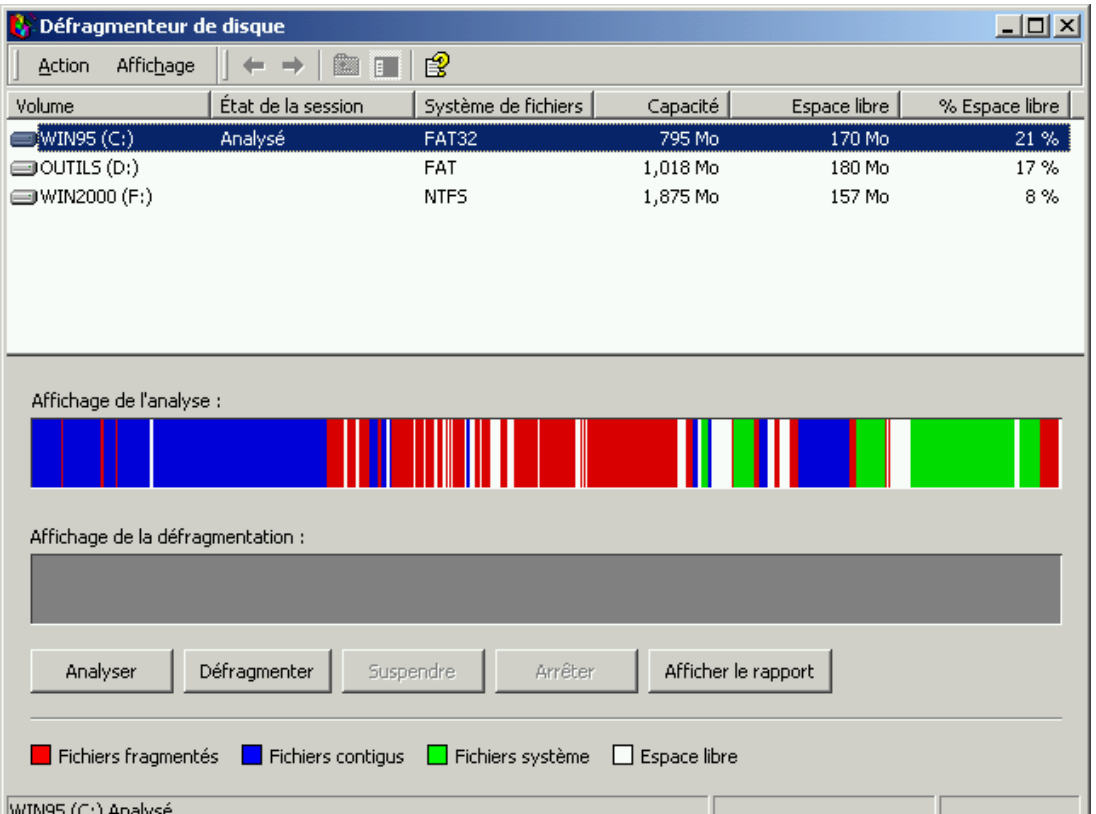


5.3 Méthodes d'allocation

- **Fragmentation externe :** lorsqu'une demande d'allocation de blocs intervient alors que chacune des zones libres est de taille strictement supérieure au nombre de blocs demandés,
  - il faut choisir une zone libre et lui prélever le nombre de blocs nécessaires,
  - une zone libre plus petite en résulte
  - **Après un certain temps de fonctionnement la mémoire contient un grand nombre de petites zones**

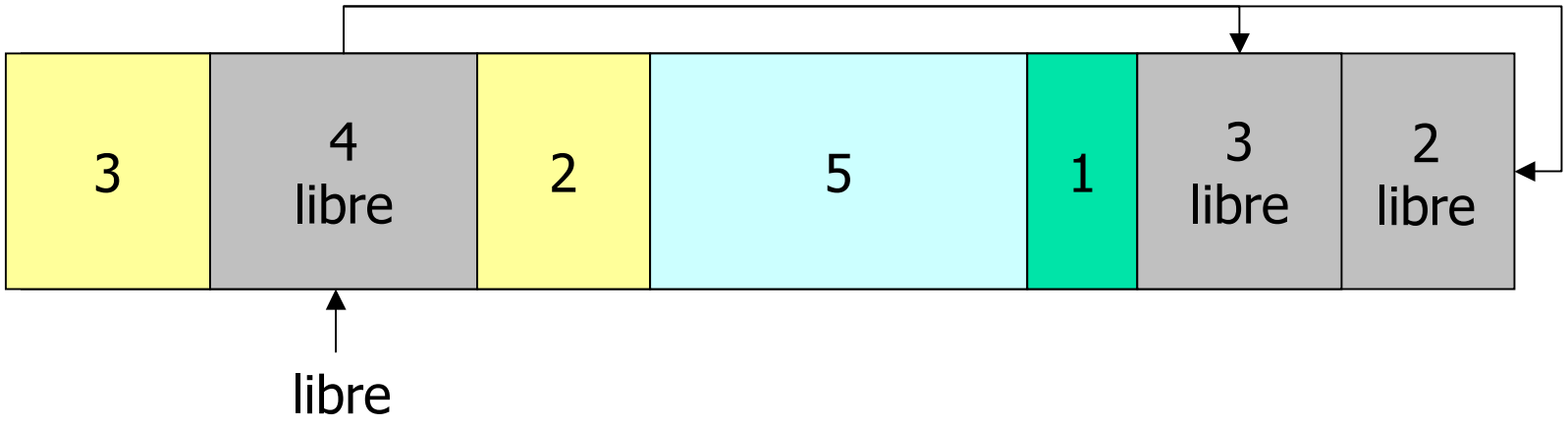
5.3 Méthodes d'allocation

- Ces types de fragmentations se retrouvent également pour les mémoires secondaires (disques)



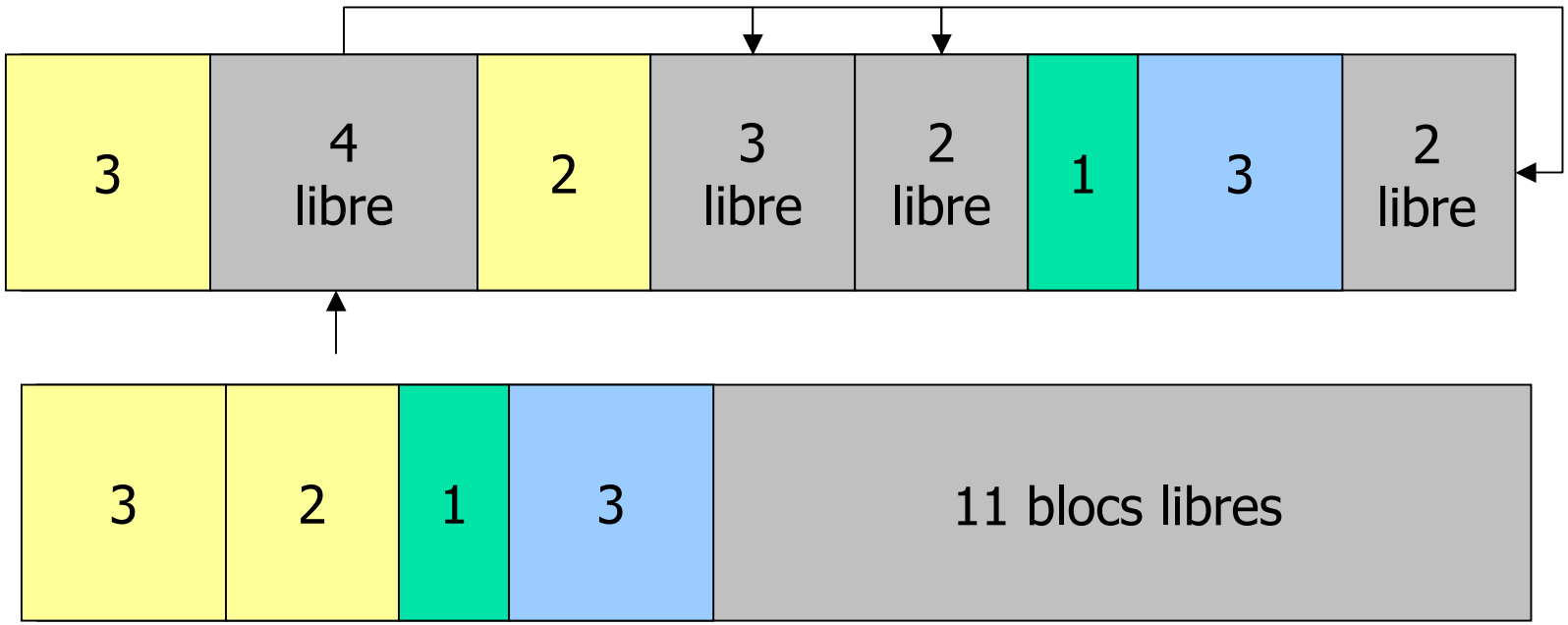
5.3 Méthodes d'allocation

- **Exemples :** demande de 5 blocs satisfaite en fusionnant 3+2



5.3 Méthodes d'allocation

- **<u>Exemples :</u>** demande de 11 blocs non satisfaite car il n'existe pas 11 blocs consécutifs => **compactage**



5.3 Méthodes d'allocation

- Le compactage consiste à déplacer les zones allouées pour obtenir des zones libres contiguës
- Diverses stratégies :
  - Conserver l'ordre
  - Minimiser les déplacements
- Le compactage a plusieurs inconvénients :
  - Occupe l'unité centrale et diminue les performances (*overhead* = surcharge)
  - Comment retrouver les zones déplacées ?

5.3 Méthodes d'allocation

- **Stratégies de placement :** permet de décider de l'endroit de la mémoire à allouer.
- L'efficacité de différentes stratégies peut être mesurée au moyen des critères suivants :
  - La **rapidité** du choix de l'emplacement
  - La **fragmentation externe** qui en résulte (elle doit être minimale)
  - **L'utilisation globale de l'espace** mémoire de la zone (elle doit être maximale)

5.3 Méthodes d'allocation

- Les trois stratégies les plus utilisées sont :
  - **Première zone libre** (*first-fit*) : exploration des zones, choix de la première de taille suffisante
  - **Le meilleur ajustement** (*best-fit*) : la plus petit zone de taille suffisante (fragment minimal)
  - **Le plus grand résidu** (*worst-fit*) : zone la plus grande choisie

5.3 Méthodes d'allocation

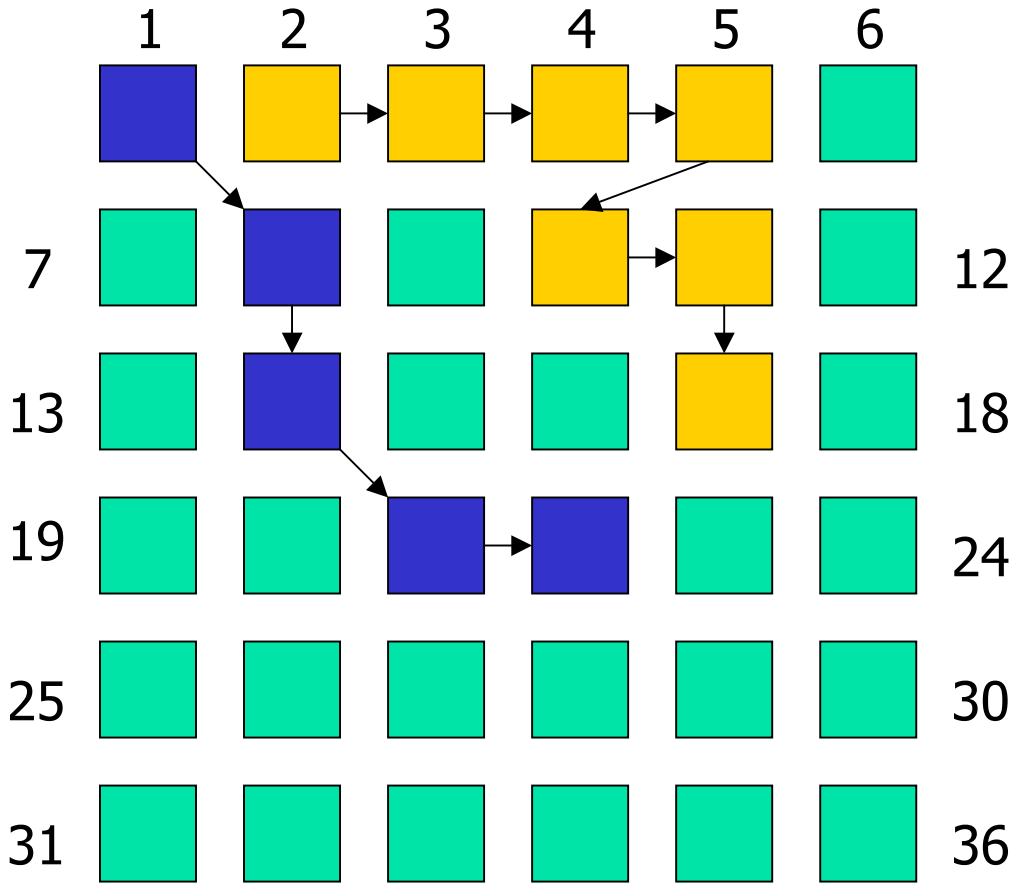
5.3.2 Allocation non contiguë

- Fractionner un ensemble d'informations pour placer les différentes parties obtenues dans des blocs libres
- Permet de satisfaire rapidement les requêtes en supprimant la fragmentation externe
- Évite une réorganisation de l'espace au moyen d'opération coûteuses (fusion, compactage)
- Mais pour retrouver l'ensemble des données stockées le SE doit conserver les informations d'allocation : chaînage et indexation

5.3 Méthodes d'allocation

- **Chaînage :** chaque ensemble d'informations est composé d'une liste chaînées de blocs (liés les uns aux autres)
- La taille de l'ensemble peut varier par adjonction de blocs prélevés dans la liste des blocs libres
- **Inconvénients :**
  - Nécessite de parcourir la liste depuis le premier bloc
  - Perte de lien dangereuse pour la stabilité du SE
  - Perte d'espace dans chaque bloc du à la présence du lien

5.3 Méthodes d'allocation



Processus 1

Processus 2

Blocs libres

Liste de blocs libres

Index des processus

1

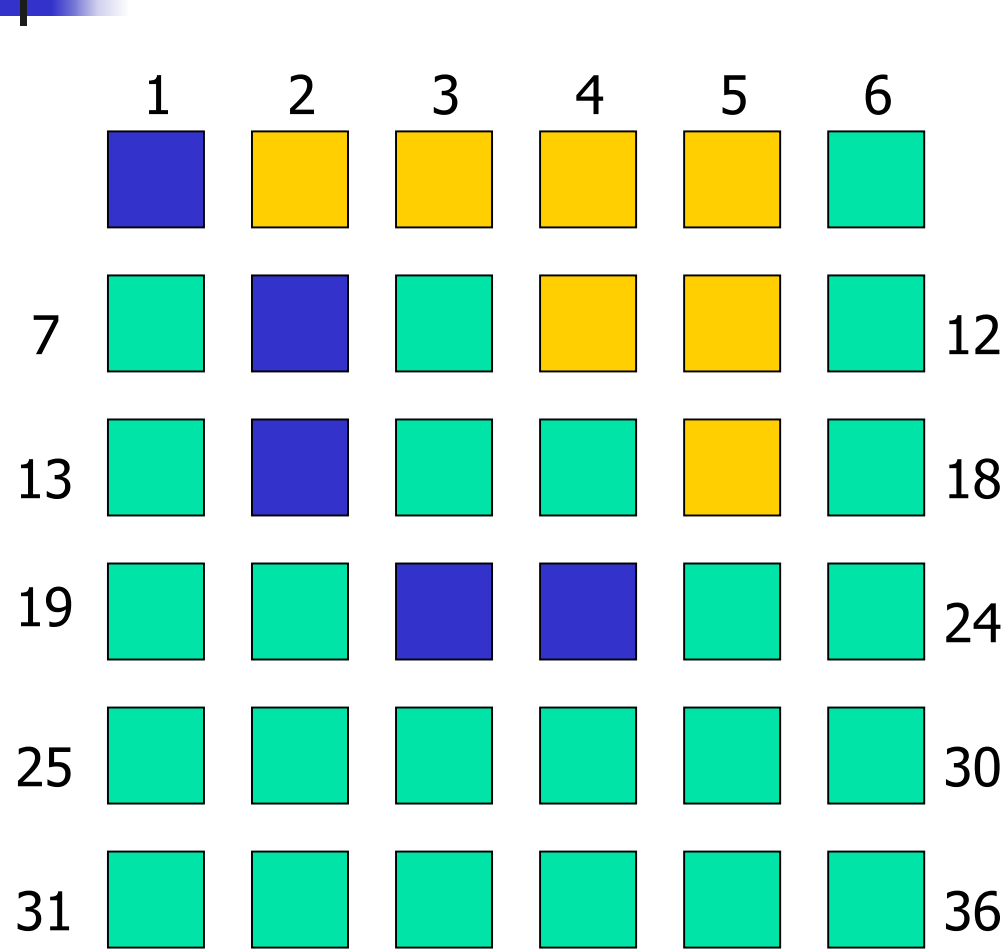
2

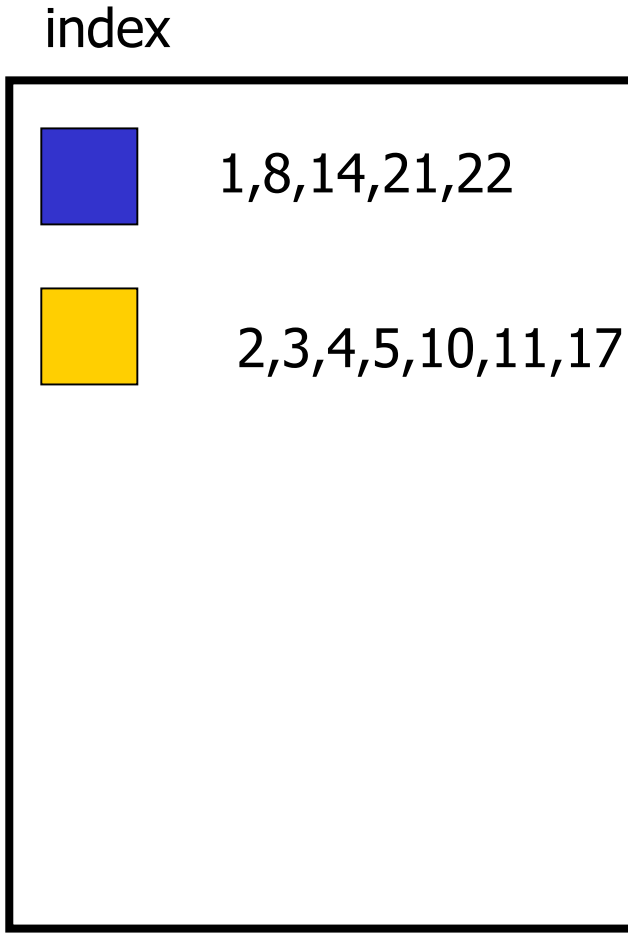
Allocation non contiguë par chaînage

5.3 Méthodes d'allocation

- **Indexation :** consiste à conserver dans une table tous les numéros de blocs utilisés pour un ensemble
- On évite ainsi les liens
- Il est possible d'accéder directement à un bloc recherché en consultant la table d'index
- **Inconvénient :**
  - La table d'index peut être très grande
  - Il est stocké lui aussi en mémoire (taille fixée)

5.3 Méthodes d'allocation





Allocation non contiguë par indexation

5.4 Simulation de mémoire

- Pour que la mémoire de l'ordinateur paraisse infinie (presque) on utilise la mémoire de masse (disque) beaucoup moins coûteuse
- La simulation de mémoire repose sur le mécanisme de transfert
- **Mécanisme de transfert :** si les données en mémoire sont repérées par des adresses virtuelles (ou logiques), le SE pourra utiliser **un mécanisme unique pour stocker les données**
- Si l'adresse se trouve au delà de la mémoire c'est que le processus concerné se trouve sur la mémoire de masse. Il faut aller le chercher : cette opération porte le nom de **swapping** *(échange)*

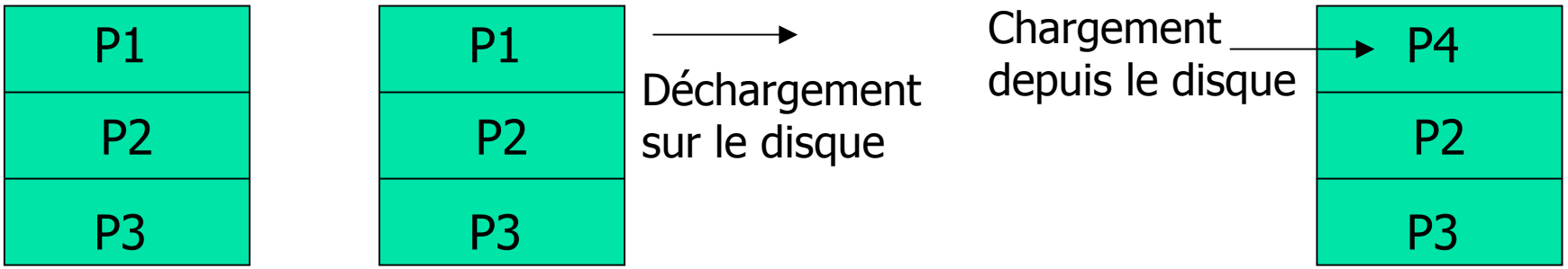
5.4 Simulation de mémoire

5.4.1 Le swapping

- Il s'effectue en deux temps :
  1. Pour faire place au processus, le SE commande le déchargement d'un processus de la mémoire centrale vers la mémoire de masse
  2. Le SE charge le nouveau processus depuis le mémoire de masse dans la mémoire centrale

5.4 Simulation de mémoire

- **Exemple :**
  - La mémoire contient à l'instant t, 3 processus (P_1,P_2,P_3) un processus P_4 est appelé par une instruction de P_2 le SE va décharger P_1 sur le disque au temps t+1, puis charger P_4 au temps t+2
  - Tout se passe comme si la mémoire centrale suffisait pour contenir P_1,P_2,P_3 et P_4



5.4 Simulation de mémoire

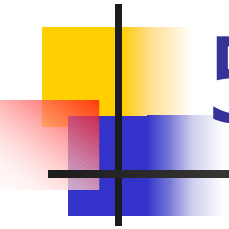
- La mise en marche du swapping
  - Le swapping s'effectue à la réception d'un signal de défaut de page quand le processus demandé par l'ordonnanceur ne se trouve pas en mémoire centrale
    - La couche de SE chargée de gérer la mémoire centrale suspend l'exécution du programme
    - Un signal est envoyé aux procédures d'interruption concernées
    - Les procédures suspendent le processus, le font charger puis le débloquent

5.4 Simulation de mémoire

5.4.2 La transparence

- Les opérations mises en œuvre pour la gestion de la mémoire centrale doivent être **transparentes** et **immédiates**
- Les activités de gestion de la mémoire doivent rester invisibles aux processus et au système informatique
- L'invisibilité pour un processus signifie que l'emplacement du processus et son déplacement à l'intérieur ou à l'extérieur de la mémoire centrale ne provoquent pas d'actions au sein du processus

5.4 Simulation de mémoire



- La gestion de la transparence des accès à la mémoire s'effectue la couche du système d’exploitation chargée de gérer la mémoire
- Notion d'adresse virtuelle (MMU)

5.4 Simulation de mémoire

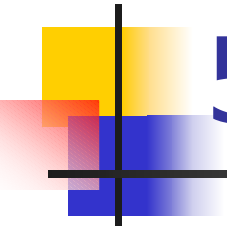
5.4.3 La protection réciproque des processus

- Dans un ordinateur en fonctionnement, au moins deux processus co-existent l'un pour le SE, l'autre pour le programme en cours d'exécution
- Pour éviter que le mauvais fonctionnement de l'un des processus ne provoque la destruction d'un autre, il faut placer un garde fou qui l'empêche d'écrire en dehors des zones mémoire qui lui sont allouées : la mécanisme de **mapping**

5.4 Simulation de mémoire

5.4.4 La notion de segment de processus

- Un processus se décompose en quatre parties principales :
  - Les données
  - Le programme (instructions)
  - La pile (récursivité, interruptions, etc.)
  - Les données du système (fichier ouverts, etc.)
- On appelle ces quatre parties des segments
- Le rôle du mapping consiste à gérer les emplacement des segments de chaque processus de telle façon qu'ils forment un ensemble logique quelque soit leur adresse réelle



5.5 Le mapping

- Le mapping est un système de gestion de la correspondance entre la mémoire virtuelle et la mémoire réelle
- Le mapping joue un rôle essentiel dans la gestion de la mémoire
- Si le mapping était assuré par un fonction logicielle son *overhead* rendrait la machine inutilisable
- Opération effectuée par la MMU

5.5 Le mapping

- Il existe différentes techniques de mapping :
  - **5.4.1 Le système de bancs :** séparation de la mémoire central en bloc de taille fixe, contrôles par un registre de sélection de banc. Chaque banc est vide ou abrite un processus. Le nombre de processus ne peut pas excéder le nombre de bancs
  - **5.4.2 Les partitions fixes :** comme le banc une partition à une taille fixe dans le temps mais variable dans l'espace
  - **5.4.3 La pagination :** la mémoire est divisée en pages (plusieurs milliers)

5.5 Le mapping

- Chaque processus occupe un certain nombre de pages dans le mémoire (physique ou virtuelle)
- Le lieu de résidence d'un processus peut être le disque aussi bien que le mémoire centrale
- Problème du va et vient ?
- Le chargement d'un processus depuis le disque ne s'avère nécessaire que dans une petite minorité de cas
- **L'adressage est relatif :** une adresse se compose d'une page A et d'un déplacement d dans la page A

5.5 Le mapping

- Une table des pages fait correspondre des pages virtuelles avec des pages réelles
- Les algorithmes de changement de page :
  - Plusieurs stratégies sont possibles pour désigner la page à décharger de la mémoire central
  - L'algorithme concerné porte la nom d'algorithme de changement de page ou d'algorithme de remplacement
  - L'efficacité de la pagination dépend de l'algorithme
  - Mauvais fonctionnement = trashing

5.5 Le mapping

- Stratégie pour décharger une page :
  - FIFO : on décharge la page la plus ancienne
    - Pb si on est dans une itération
  - LRU (least recently used)
  - LFU (least frequently used)
  - NRU (not recently used)