Université Sultan Moulay Slimane

Ecole Supérieure de Technologie de Béni Mellal

Département : ITG
Filière : ARI - *2ème année - S3*
Année universitaire: 2023-2024

Module: Sys. & Réseaux Informatiques Avancés (M11)
Elément: Systèmes d'exploitation avancés (E2)

Support de cours

Systèmes d'exploitation Avancés

Chapitre III: Gestion de la mémoire

Pr. Siham BAKKOURI

Email: s.bakkouri@usms.ma

Plan

- INTRODUCTION
- GESTION ÉLÉMENTAIRE DE LA MÉMOIRE
- CONCEPTS FONDAMENTAUX
  - → Adresses logiques et physiques
  - → Allocation de la mémoire
- SWAPPING
- PAGINATION
- SEGMENTATION
- SEGMENTATION AVEC PAGINATION
- LA MÉMOIRE VIRTUELLE

Introduction

→ Mémoire centrale

- La **mémoire centrale** (MC ou mémoire vive) est l'endroit où les programmes et les données se trouvent lorsqu'ils sont exécutés par le processeur.
- Elle est distinguée du concept de **mémoire secondaire** (MS), représenté par des disques de plus grande capacité où les processus peuvent résider avant d'être exécuté (les disques durs, les disques SSD...).
- La mémoire centrale est de deux types : **RAM** (*Random Access Memory*) ou **ROM** (*Read Only Memory*) utilisée pour le BIOS.
- Elle favorise un accès rapide mais elle fournit une faible capacité de stockage et est d'un coût élevé en raison de la taille croissante des programmes. On aura toujours besoin de gestionnaires de mémoires performants.

Introduction

→ Gestionnaire de la mémoire

- Le **Gestionnaire de la Mémoire** est nécessaire pour gérer l'hiérarchie de mémoire
  - **Mémoire du cache**: volatile, rapide, chère.
  - **Mémoire centrale**: volatile, moins rapide, moins chère.
  - **Mémoire de masse** –**disque**: non volatile, lente, pas chère.
- **Fonctions**
  - Connaître les zones libres et utilisées
  - Allouer et récupérer la mémoire
  - Offrir de la mémoire virtuelle : Utiliser la MS
- **Stratégies**
  - Gestion de la mémoire élémentaire : Monoprogrammation, multiprogrammation..
  - Va et vient (Swapping)
  - Pagination
  - Segmentation
  - …

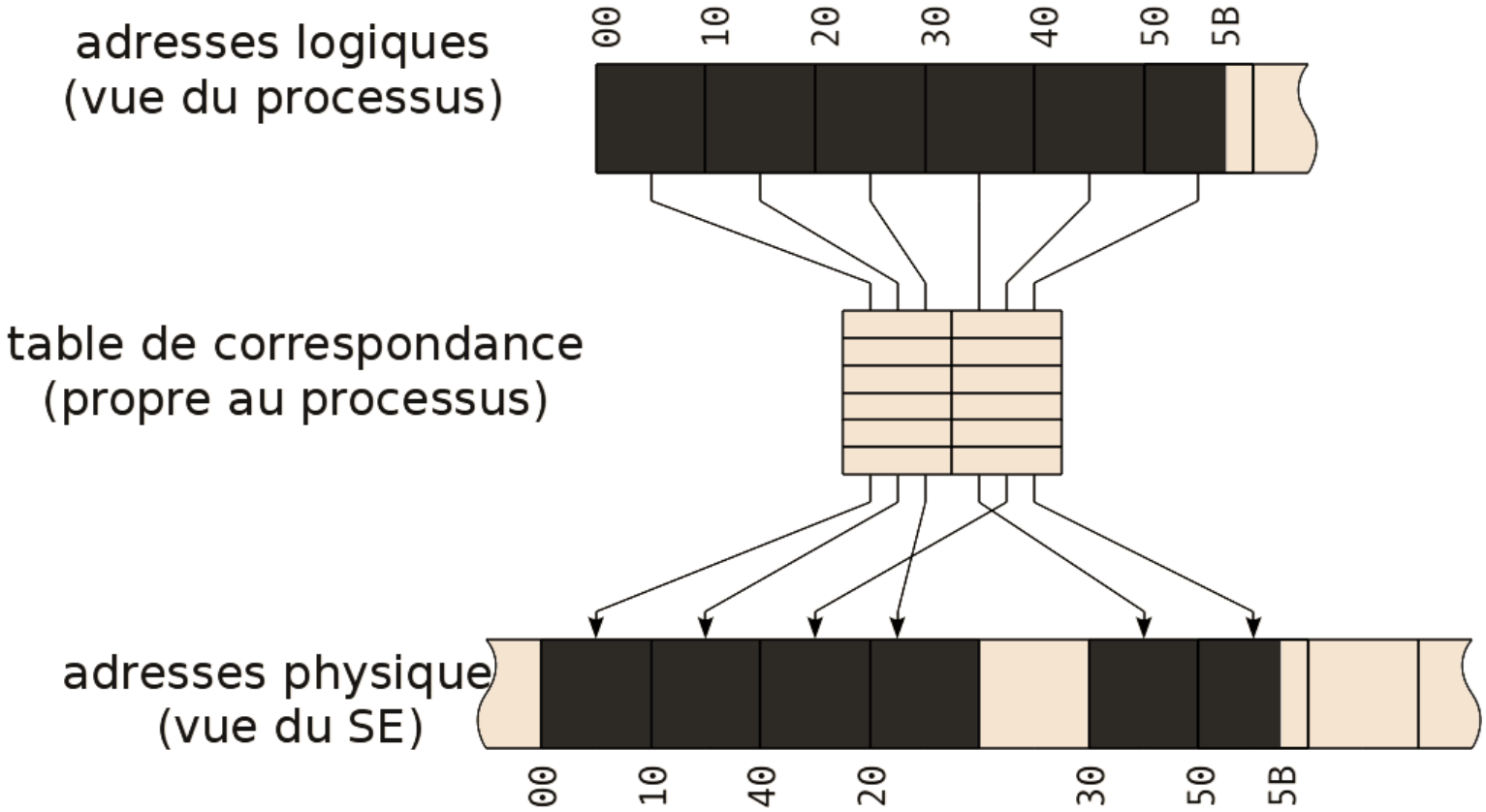
Concepts fondamentaux

→ Adresses physiques et logiques

- **L'adresse logique**, également appelée adresse virtuelle, est l'adresse utilisée par un programme ou un processus pour accéder à un emplacement mémoire. Cette adresse est générée par le processeur
- **L'adresse physique**, fait référence à l'emplacement réel en mémoire où se trouve l'information.
- La **traduction des adresses logiques en adresses physiques** est gérée par l'unité de gestion de mémoire du processeur (**MMU**). Cette traduction est généralement effectuée en utilisant une table de pagination, qui associe les adresses logiques aux adresses physiques correspondantes.

- **Mémoire physique**: : la mémoire principale RAM de la machine (mémoire vive)
  - **Adresses physiques**: les adresses de cette mémoire
- **Mémoire logique**: l'espace d'adressage d'un programme (mémoire virtuelle)
  - **Adresses logiques**: les adresses dans cet espace

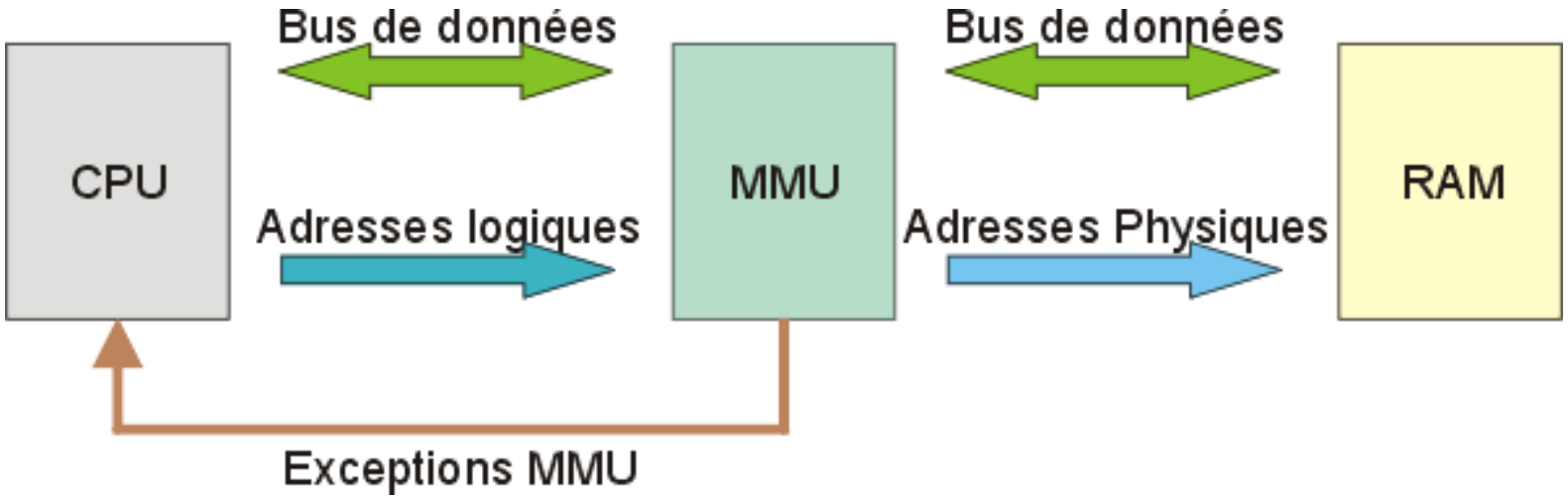
→ **Adresses physiques et logiques**



Concepts fondamentaux

→ Unité de gestion mémoire MMU

- En anglais : Memory Management Unit
- Matériel de mapping des adresses virtuelles en adresses physiques
- La valeur du registre d'emplacement est ajoutée à chaque adresse générée par un processus au moment de l'envoi à la mémoire
- Le processus traite des adresses *logiques*; il ne voit jamais les adresses physiques



→ Allocation de la mémoire

- Avant d'implanter une technique de gestion de la mémoire centrale par va-et-vient (swapping), il est nécessaire de connaître son état : les zones **libres** et **occupées**; de disposer d'une stratégie d'allocation et enfin de procédures de libération.

- Les techniques que nous allons décrire servent de base au va-et-vient; on les met aussi en œuvre dans le cas de la multiprogrammation simple où plusieurs processus sont chargés en mémoire et conservés jusqu'à la fin de leur exécution.

→ Allocation de la mémoire

❑ État de la mémoire

Le système garde la trace des emplacements occupés de la mémoire par l'intermédiaire:

- **D'une table de bits**
- **D'une liste chaînée**

La mémoire étant découpée en unités, en blocs, d'allocation.

→ **Allocation de la mémoire**

❑ **État de la mémoire**

1. Tables de bits

On peut conserver l'état des blocs de mémoire grâce à une table de bits. Les unités libres étant notées par 0 et ceux occupées par un 1. (ou l'inverse)

0	0	1	1	0	0	

- La technique des tables de bits est facile à mettre en place, mais elle n'est pas largement utilisée.
- On peut noter que plus l'unité d'allocation est petite, moins il y a de gaspillage de mémoire lors de l'allocation, mais en revanche , la table qui stocke ces informations prend plus de place en mémoire.

→ **Allocation de la mémoire**

❑ **État de la mémoire**

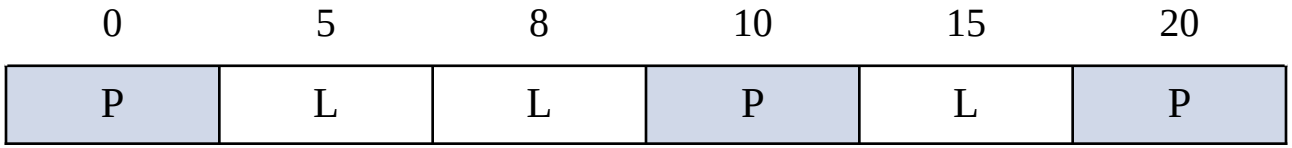
2. Listes chainées

On peut représenter la mémoire par une liste chaînée de structures dont les membres sont :

- Le type (libre ou occupé)
- L'adresse de début
- La longueur
- Un pointeur sur l'élément suivant

Exemple

Pour une mémoire ayant l'état suivant :

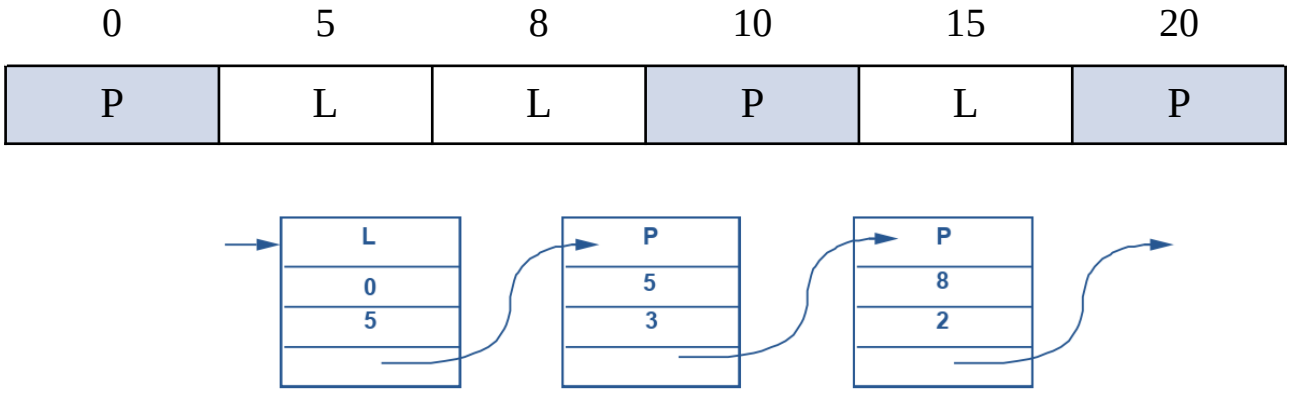


→ **Allocation de la mémoire**

❑ **État de la mémoire**

2. Listes chainées

Exemple Pour une mémoire ayant l'état suivant :



- ❑ On peut légèrement modifier ce schéma en prenant **deux listes** : **l'une** pour **les processus** et **l'autre** pour **les zones libres**.
- ❑ La liste des blocs libres peut elle-même se représenter en réservant quelques octets de chaque bloc libre pour contenir un pointeur sur le bloc libre suivant.

→ Allocation de la mémoire
❑ Politiques d'allocation

L'allocation d'un espace libre pour un processus peut se faire suivant trois stratégies principales (trois algorithmes de placement) :

- Le « **premier ajustement** » (*first fit*) : on prend le premier bloc libre de la liste qui peut contenir le processus qu'on désire charger
- Le « **meilleur ajustement** » (*best fit*): tente d'allouer au processus l'espace mémoire le plus petit qui puisse le contenir
- Le « **pire ajustement** » (*worst fit*): prend le plus grand bloc disponible et le fragmente en deux.

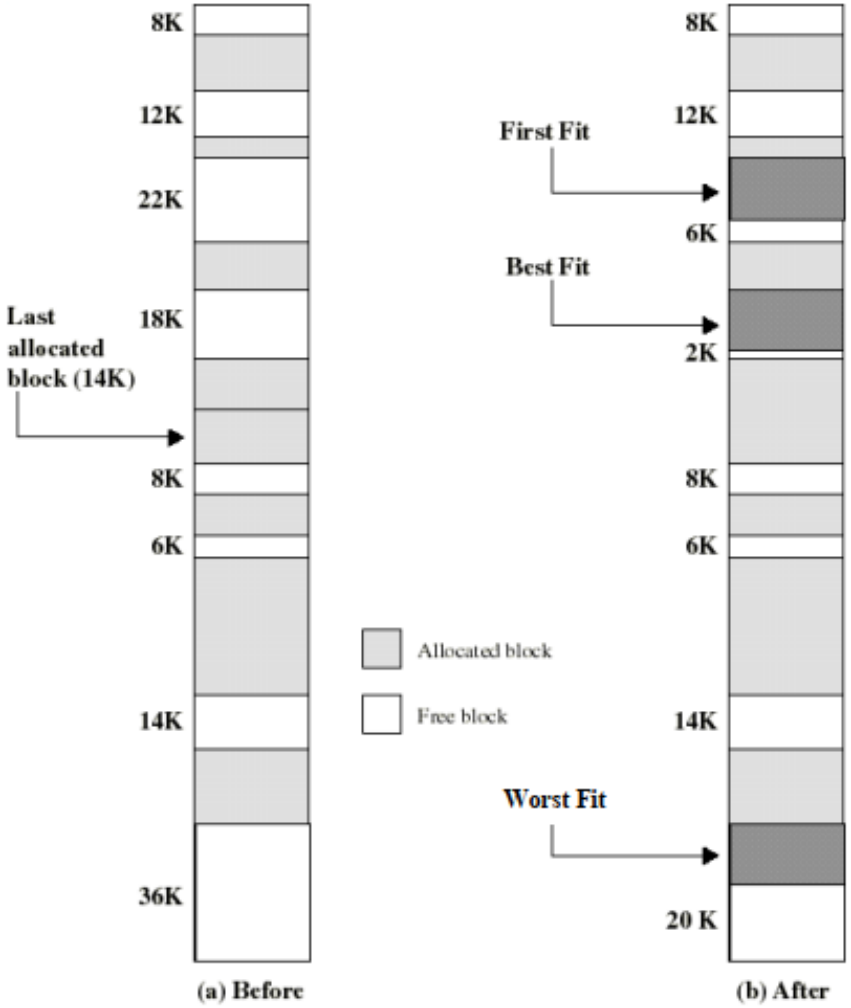
- Des simulations ont montré que le « **premier ajustement** » était meilleur que les autres.
- « **meilleur ajustement** », n'est pas optimal car il produit une fragmentation importante, donc, la mémoire se remplit de trous trop petits pour contenir un programme.
- « **pire ajustement** »: les allocations se feront souvent à la fin de la mémoire

→ Allocation de la mémoire

❑ Politiques d'allocation

Exemple

Configuration de la mémoire avant et après l'allocation d'un bloc de 16 Ko

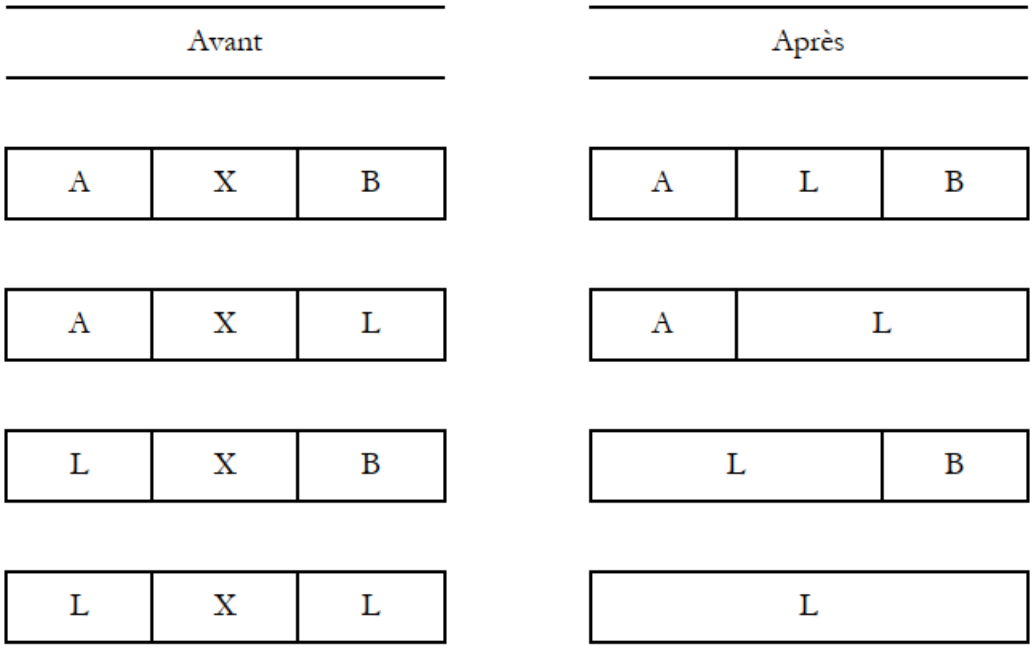


→ Allocation de la mémoire

❑ Libération

La libération se produit quand un processus est évacué de la mémoire. On marque alors le bloc à libre et on le fusionne éventuellement avec des blocs adjacents.

❑ Supposons que **X** soit le bloc qui se libère, on a les schémas de fusion suivants :



→ Allocation de la mémoire

❑ Fragmentation

- La **fragmentation** se produit lorsque les espaces libres de mémoire ne sont pas répartis de manière homogène, ce qui entraîne une utilisation inefficace de l'espace disponible.
- La fragmentation peut se produire de deux manières : la fragmentation **externe** et la fragmentation **interne**.
  - ❑ **Fragmentation externe** : se produit lorsqu'il y a suffisamment d'espace libre dans la mémoire pour allouer une nouvelle demande de mémoire, mais que l'espace libre n'est pas contigu. Cela peut rendre impossible l'allocation d'un bloc de mémoire suffisamment grand pour satisfaire une demande de taille spécifique.
  - ❑ **Fragmentation interne** :se produit lorsqu'un bloc de mémoire est alloué à un processus, mais que la taille du bloc est supérieure à la taille de la demande réelle. Cela peut entraîner un gaspillage d'espace de mémoire précieux.

→ Allocation de la mémoire
❑ La récupération de la mémoire

- La **fragmentation** de la mémoire est particulièrement dommageable car elle peut saturer l'espace disponible rapidement. Ceci est particulièrement vrai pour les gestionnaires de fenêtrage.

- Pour la diminuer, on peut **compacter** régulièrement la mémoire. Pour cela, on déplace les processus, par exemple, vers la bas de la mémoire et on les range l'un après l'autre de manière **contiguë**. On construit alors un seul bloc libre dans le haut de la mémoire. Cette opération est coûteuse et nécessite parfois des circuits spéciaux

  → Une fois qu'un objet ou une zone a été utilisé, le programmeur système doit récupère la mémoire « à la main » par une libération du pointeur sur cette zone – free(). => Ceci est une source d'erreurs car on oublie parfois cette opération.

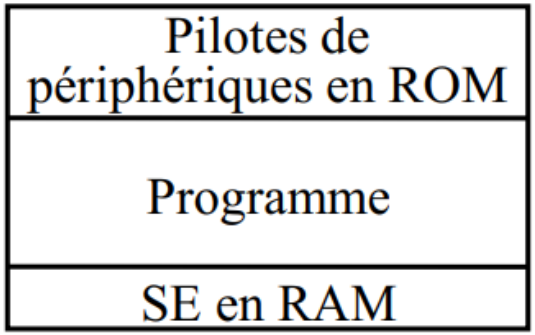
  → Si on alloue dans une fonction, il n'y a plus moyen d'accéder au pointeur après le retour de la fonction. Le bloc de mémoire est alors perdu et inutilisable. On a une « fuite de mémoire » – *memory leak*

Gestion de la mémoire élémentaire

→ Monoprogrammation

- La **monoprogrammation**, également appelée la programmation à un seul processus, est un modèle d'exécution de programme dans lequel un seul programme peut être exécuté à la fois.

- Dans ce modèle, le système d'exploitation alloue l'ensemble des ressources matérielles, telles que le processeur, la mémoire et les périphériques d'E/S, à un seul processus à la fois.



- **Avantages** *:* Simplicité, SE très réduit
- **Inconvénients** *:*
  - Mauvaise utilisation de la mémoire (un seul processus) et du processeur (attente des E/S).
  - Manque de flexibilité : les programmes sont limités à la mémoire existante

Gestion de la mémoire élémentaire

→ Multiprogrammation avec partitions fixes

- La **multiprogrammation** est une technique qui permet une utilisation efficace de la mémoire en permettant à plusieurs programmes d'utiliser en alternance la même zone de mémoire.
- La multiprogrammation améliore le taux d'utilisation du CPU mais requiert un bon partitionnement de la mémoire.
- A l'initialisation du système, la mémoire est divisée en *n* partitions de taille fixe.
- Un processus est mis dans une file en attente d'une partition libre
- Deux méthodes de gestion

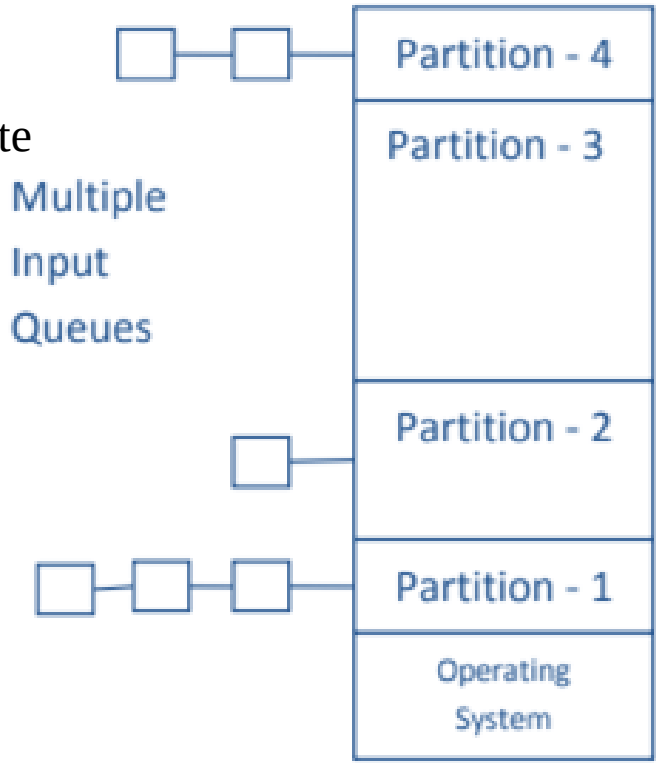
  → **Files multiples**

  → **File unique**

→ Multiprogrammation avec partitions fixes

❑ Files multiples

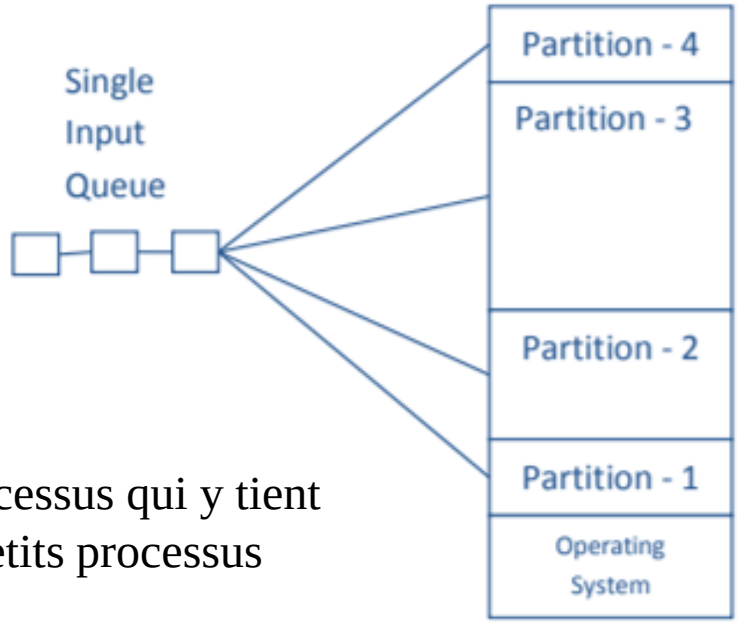
- **Une file par partition**
- Chaque processus est mis dans la file de la plus petite partition pouvant le contenir
- Des petits processus en attente alors qu'une partition grande est libre



→ Multiprogrammation avec partitions fixes

❑ File unique

- **Une file pour toutes les partitions**
- Attribuer la partition libérée au 1er processus qui y tient.



❑ **Problème** : Perte de mémoire
(1 petit processus dans 1 grande partition)

❑ **Solution**

→ Attribuer la partition libérée au plus grand processus qui y tient
➔ ce qui peut entraîner une attente pour les petits processus

→ Réserver une partition pour les petits processus

→ Choisir le processus ayant trop attendu

Swapping

→ Le va-et-vient (Swapping)

Le va-et-vient est mis en œuvre lorsque tous les processus ne peuvent pas tenir simultanément en mémoire.

→ On doit alors en déplacer temporairement certains sur une mémoire provisoire, en général, une partie réservée du disque (***swap area*** ou ***backing store***).

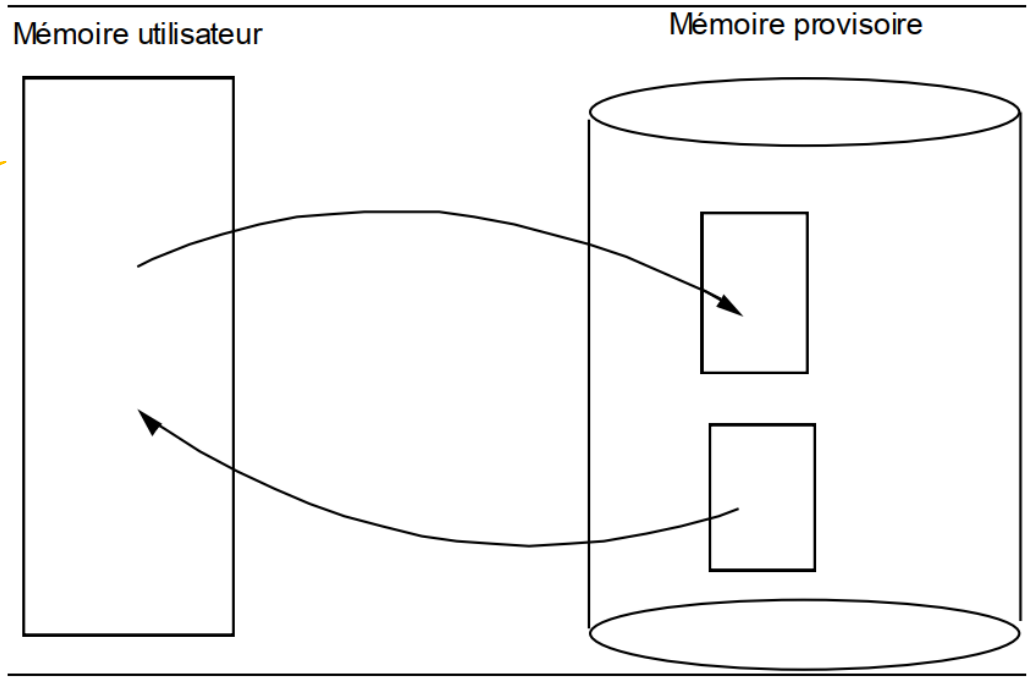
- Sur le disque, la zone de **va-et-vient** d'un processus peut être allouée à la demande dans **la zone de va-et-vient** générale (***swap area***). Quand un processus est déchargé de la mémoire centrale, on lui recherche une place.
- Les places de va-et-vient sont gérées de la même manière que la mémoire centrale.
- La zone de va-et-vient d'un processus peut aussi être allouée une fois pour toute au début de l'exécution.
- Lors du déchargement, le processus est sûr d'avoir une zone d'attente libre sur le disque.

Swapping

→ Le va-et-vient (Swapping)

Le système exécute pendant un certain quantum de temps les processus en mémoire puis déplace un de ces processus au profit d'un de ceux en attente dans la mémoire provisoire.

Le système de va-et-vient, bien qu'il permette de compenser le manque de mémoire nécessaire à plusieurs utilisateurs, ne permet cependant pas l'exécution de programmes de taille supérieure à celle de la mémoire centrale.



Pagination

→ Pagination : concept de base

La **pagination** est une technique de gestion de la mémoire dans les systèmes informatiques.

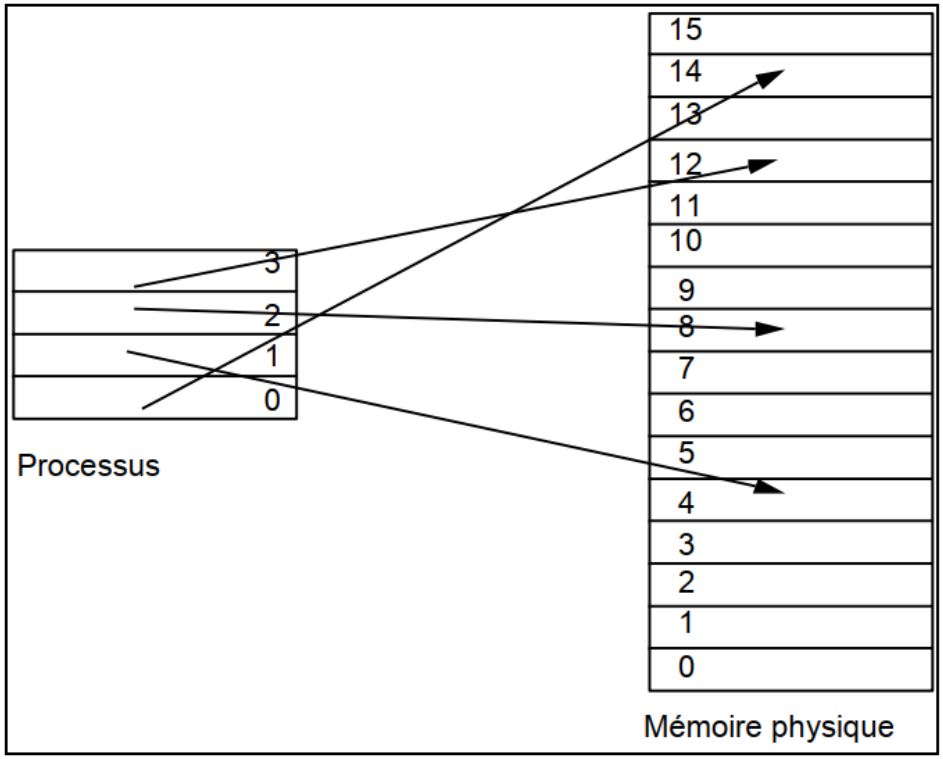
- Elle consiste à diviser la mémoire vive (**RAM**) et les processus en blocs de tailles fixes appelés pages, chaque page étant numérotée de manière unique.
- **Les *pages mémoire*** sont souvent appelées *"frames"* ou *"cadres"* tandis que les ***pages de processus*** sont simplement appelées *"pages"*.
- Lorsqu'un programme ou un processus nécessite une certaine quantité de mémoire, le système d'exploitation alloue des pages de mémoire à ce processus.
- Si toutes les pages disponibles sont déjà utilisées, le système peut déplacer certaines pages inutilisées de la mémoire vive vers le disque dur, libérant ainsi de l'espace pour les nouvelles pages. Ce processus est appelé "***swap***" ou "***échange de pages***".

Pagination

La **pagination** permet d'avoir en mémoire un processus dont les adresses sont non contiguës. Pour réaliser ceci, on partage l'espace d'adressage du processus et la mémoire physique en pages de quelques kilo-octets. Les pages du processus sont chargées à des pages libres de la mémoire

- On conserve l'emplacement des pages par une table de transcodage (ou table de pages) :

0	**14**
1	**4**
2	**8**
3	**12**



Si un programme nécessite 10 pages de mémoire et que seules 5 pages sont disponibles en RAM, le système d'exploitation peut déplacer 5 pages inutilisées de la mémoire vive vers le disque dur pour libérer de l'espace pour les nouvelles pages du programme.

Segmentation

→ Segmentation : concept de base

Contrairement à la pagination, où la mémoire est divisée en pages de taille fixe, la **segmentation** divise la mémoire en segments de **taille variable**.

- Chaque segment peut contenir une partie d'un programme ou d'un processus.
- Les segments sont numérotés de manière unique et peuvent être alloués à un programme ou à un processus en fonction de ses besoins en mémoire.
- Cette technique de gestion de la mémoire permet une utilisation plus efficace de la mémoire en permettant l'allocation de segments de taille variable en fonction des besoins de chaque programme ou processus.

La segmentation est une technique de gestion de la mémoire plus complexe que la pagination, mais elle peut offrir une utilisation plus efficace de la mémoire dans certains cas.

Segmentation

- L'accès aux segments se fait via une **table de segments**.
- L'adresse logique est constituée du numéro de segment et d'un offset.
- Chaque entrée de la table comporte **l'adresse de départ (de base)** du segment et sa **taille**.

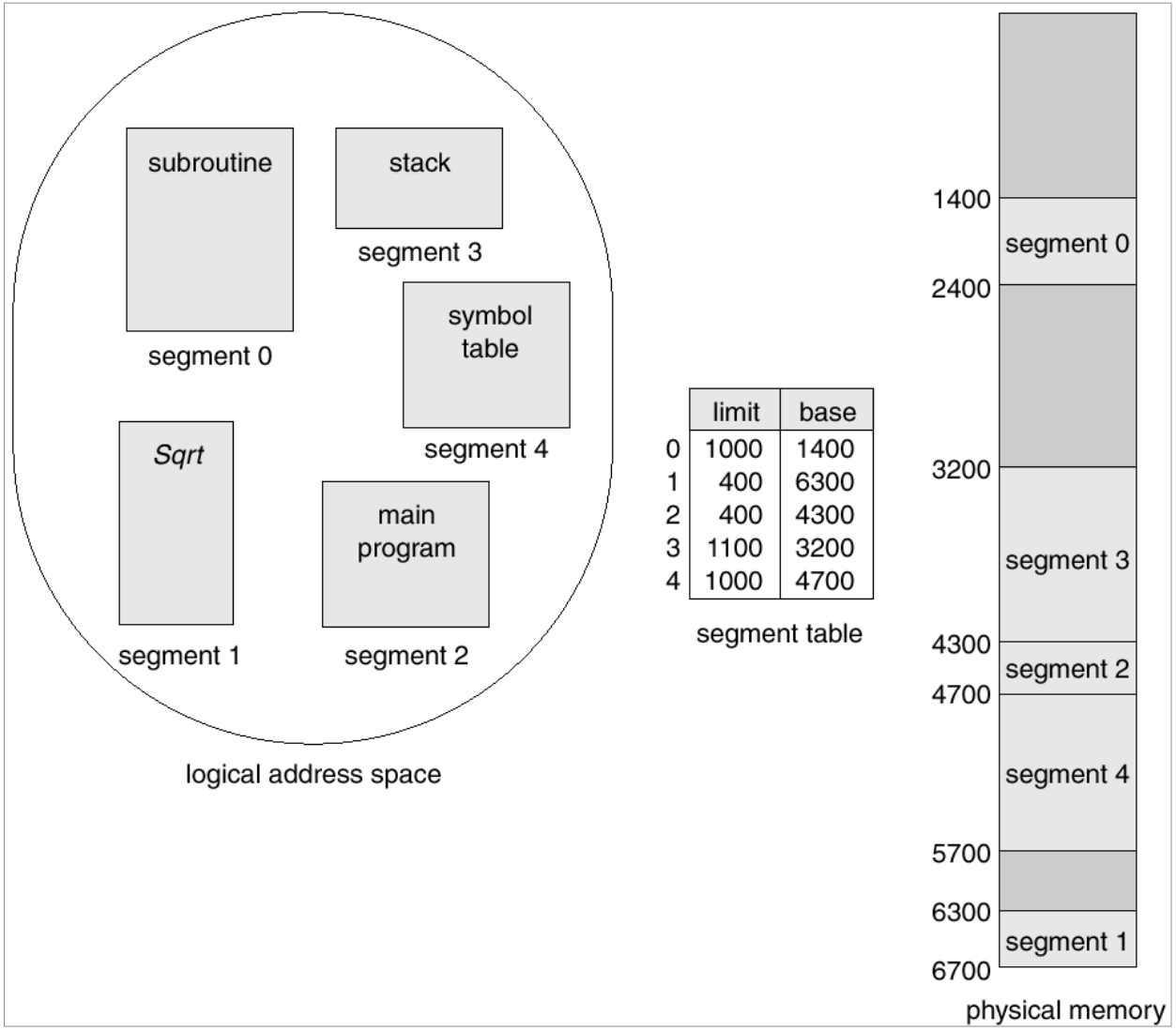
> **l'offset** est utilisé pour calculer l'adresse physique d'un emplacement mémoire à partir d'une adresse logique.
>
> **Adresse physique = adresse de base du segment + offset**

Exemple

Si un programme nécessite une grande quantité de mémoire pour stocker des images ou des fichiers multimédias, il peut être plus efficace d'utiliser la segmentation pour allouer un segment de taille suffisante pour stocker ces données plutôt que de diviser les données en petites pages de taille fixe.

Segmentation

Exemple



Segmentation avec Pagination

- Le problème avec la **segmentation** est que l'unité d'allocation de mémoire (le segment) est de longueur variable
- La **pagination** utilise des unités d'allocation de mémoire fixe, éliminant donc ce problème
- La segmentation et la pagination concernent des problèmes différents. Ce sont deux techniques qui **peuvent se combiner** :
  - La **segmentation** découpe les **processus** en zones linaires pouvant être gérées différemment selon que ces segments sont propres au processus, qu'ils sont partagées, lus, écrits ou exécutées et de manière à protéger les processus entre eux.
  - La **pagination** découpe la **mémoire** en **pages** non contiguës mais de même taille. Elle procure aux processus des espaces d'adresse continus (nécessaires aux segments). Les pages mémoires peuvent n'être allouées que lorsqu'un processus en a besoin. On obtient de la sorte une mémoire virtuelle de taille supérieure à la mémoire réelle.
  - Une entrée de la table des segments contient l'adresse de base d'une table des pages pour ce segment

Les systèmes d'exploitation qui gèrent ces deux techniques simultanément administrent **une table de segments et plusieurs tables de pages**.

Segmentation avec Pagination

→ Segmentation avec pagination

- Le problème avec la **segmentation** est que l'unité d'allocation de mémoire (le segment) est de longueur variable
- La **pagination** utilise des unités d'allocation de mémoire fixe, éliminant donc ce problème
- La **segmentation** et la **pagination** concernent des problèmes différents. Ce sont deux techniques qui **peuvent se combiner** :
  - ❑ La **segmentation** découpe les **processus** en zones linaires pouvant être gérées différemment selon que ces segments sont propres au processus, qu'ils sont partagées, lus, écrits ou exécutées et de manière à protéger les processus entre eux.
  - ❑ La **pagination** découpe la **mémoire** en **pages** non contiguës mais de même taille. Elle procure aux processus des espaces d'adresse continus (nécessaires aux segments). Les pages mémoires peuvent n'être allouées que lorsqu'un processus en a besoin. On obtient de la sorte une mémoire virtuelle de taille supérieure à la mémoire réelle.
  - ❑ Une entrée de la table des segments contient l'adresse de base d'une table des pages pour ce segment

Les systèmes d'exploitation qui gèrent ces deux techniques simultanément administrent **une table de segments et plusieurs tables de pages**.

La mémoire virtuelle

→ Mémoire virtuelle

- **La mémoire virtuelle** permet d'exécuter des programmes dont la taille excède la taille de la mémoire réelle (vive).
- **La mémoire virtuelle** n'est pas gérée octet par octet. Elle est découpée en blocs de taille fixe, souvent ***4ko***, appelés ***pages***. Un programme et ses données y occupent un nombre entier de pages.
- Pour ce faire, les processus ainsi que la mémoire réelle sont divisés en pages, un processus étant composé d'un ensemble de pages. L'ensemble complet d'un processus constitue l'espace d'adressage ou la mémoire virtuelle.
- La mémoire virtuelle réside sur le disque. Contrairement à la **pagination** décrite précédemment, seule une partie des pages est chargée en mémoire à un moment donné. Cette partie chargée en mémoire est appelée espace physique ou réel.

La mémoire virtuelle

→ **Mémoire virtuelle**

Exemple