1- Introduction

- La mémoire principale est un espace de stockage temporaire sert a stocker les données et les programmes quand le processeur les exécutent.

- C'est un donc espace partagé entre les processus système et les processus utilisateurs.

OS	Processus utilisateurs

1.1 Problématique

- Avant d'être exécuté, un programme doit être charger a la mémoire centrale,
- pour effectuer le chargement, le système alloue un espace de mémoire libre et il y place le processus.
- Il libérera cet emplacement une fois le programme se termine.
- Comment doit on structurer ou organiser la mémoire pour allouer les espaces nécessaires au différents processus? .

2- Organisation

- Un espace mémoire est un ensemble de mots (généralement de taille 16 bits) ayant chacun son propre adresse, c'est le plus petit espace mémoire qu'on peut définir.
- les opérations sur les mots sont : lecture ou écriture d'un mot.
- Cependant, la quantité minimale de mémoire pouvant être allouée n'est pas le mot, mais un certain nombre fixe de mots(512 mots=1024 octets), appelé bloc.
- Toute sous ensemble de N blocs contigus de la mémoire est appelé *zone ou partition de taille N*.
- les opérations sur les zones sont: allocation ou libération d'une zone.

2.1 Adressage

- Adresses Physiques (absolu) ces sont des adresses comptées a partir de la première adresse de la mémoire centrale.

- Adresses logiques(relatives), il s'agit des adresses comptées a partir de la première adresse de la zone attribuée au processus.

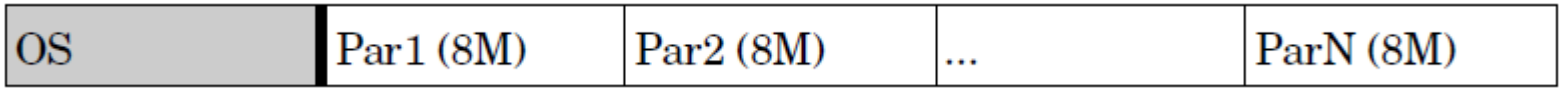
2.2 Structuration de l'espace mémoire Utilisateur

- La structuration de l'espace mémoire consiste à organiser cet espace en le divisant en des partitions de taille fixe ou variable afin d'y allouer à tous les processus utilisateurs.

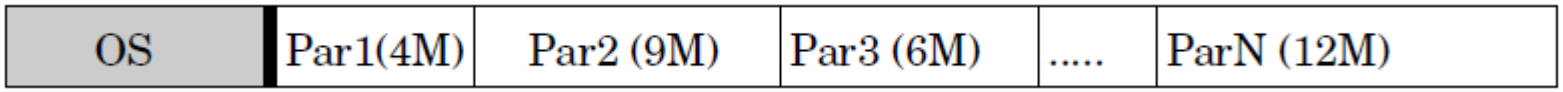
2.2.1 Structuration par Partitionnement fixe (1)

- L'espace utilisateur est découpé en plusieurs zones (partitions) de tailles homogènes ou hétérogènes.

- *Exemple:*
  - Partitions de même taille



  - Partitions de tailles hétérogènes.



2.2.1 Structuration par Partitionnement fixe (2)

- Chaque partition est caractérisée par:
  - Adresse
  - Taille
  - Indivisible: Ne peut contenir qu'un seul processus
  - Etat:
    - Libre
    - Occupée (Allouée)

2.2.1.1 Algorithmique d'allocation

Principe:

- Processus P de taille T(P) à charger en MC →Trouver une partition Pi libre tel que: Taille(Pi) ≥ T(P)
- S'il existe une telle partition alors le processus sera chargé immédiatement,
- si ce n'est pas le cas (aucune partition de taille supérieur a T(P)) il sera différé (attendre la libération d'une partition convenable).

Problèmes:

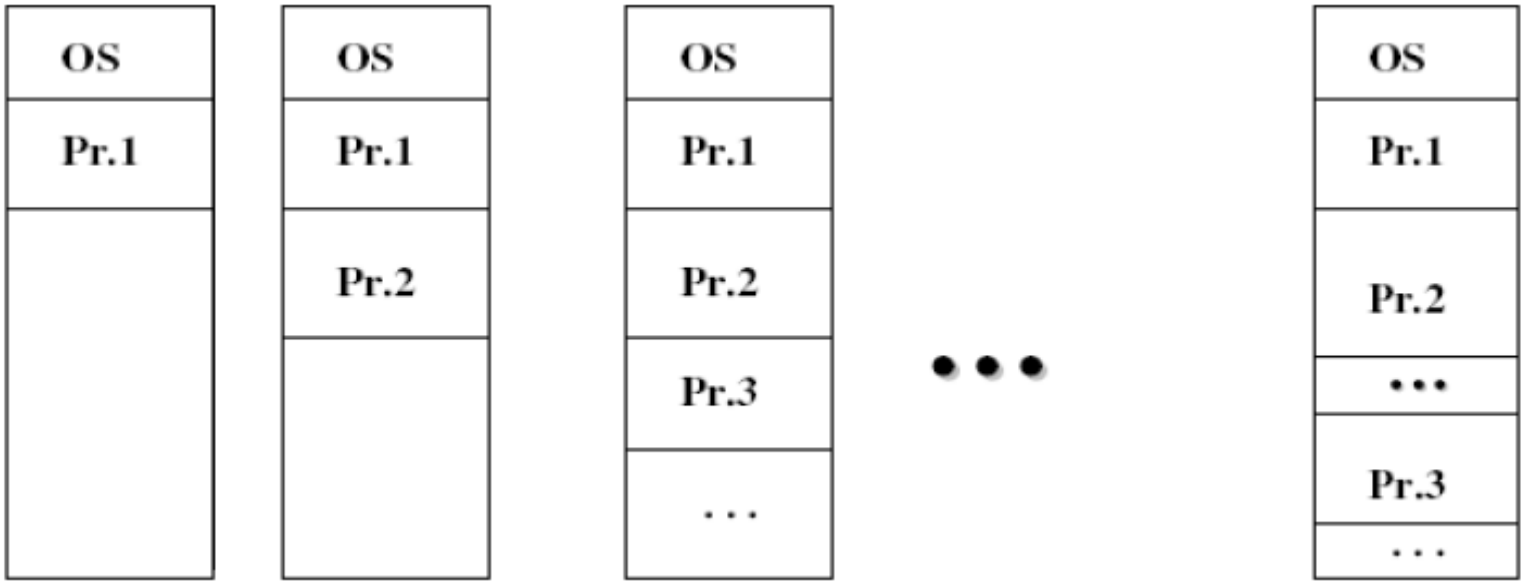
- Taille T(P) supérieure à toutes les partitions (libres ou occupées).
- si T(Pi) > T(P) alors on a un espace résiduel (Fragment) inutilisable→Problème de la fragmentation interne.

2.2.2 Structuration par Partitionnement variable (1)

- Au départ l'espace utilisateur contient une seule partition formée de l'espace tout entier

- les partitionnement sont crées au fur et a mesure de l'allocation:

2.2.2 Structuration par Partitionnement variable (2)



Partitionnement créé au fur et à mesure de l'allocation

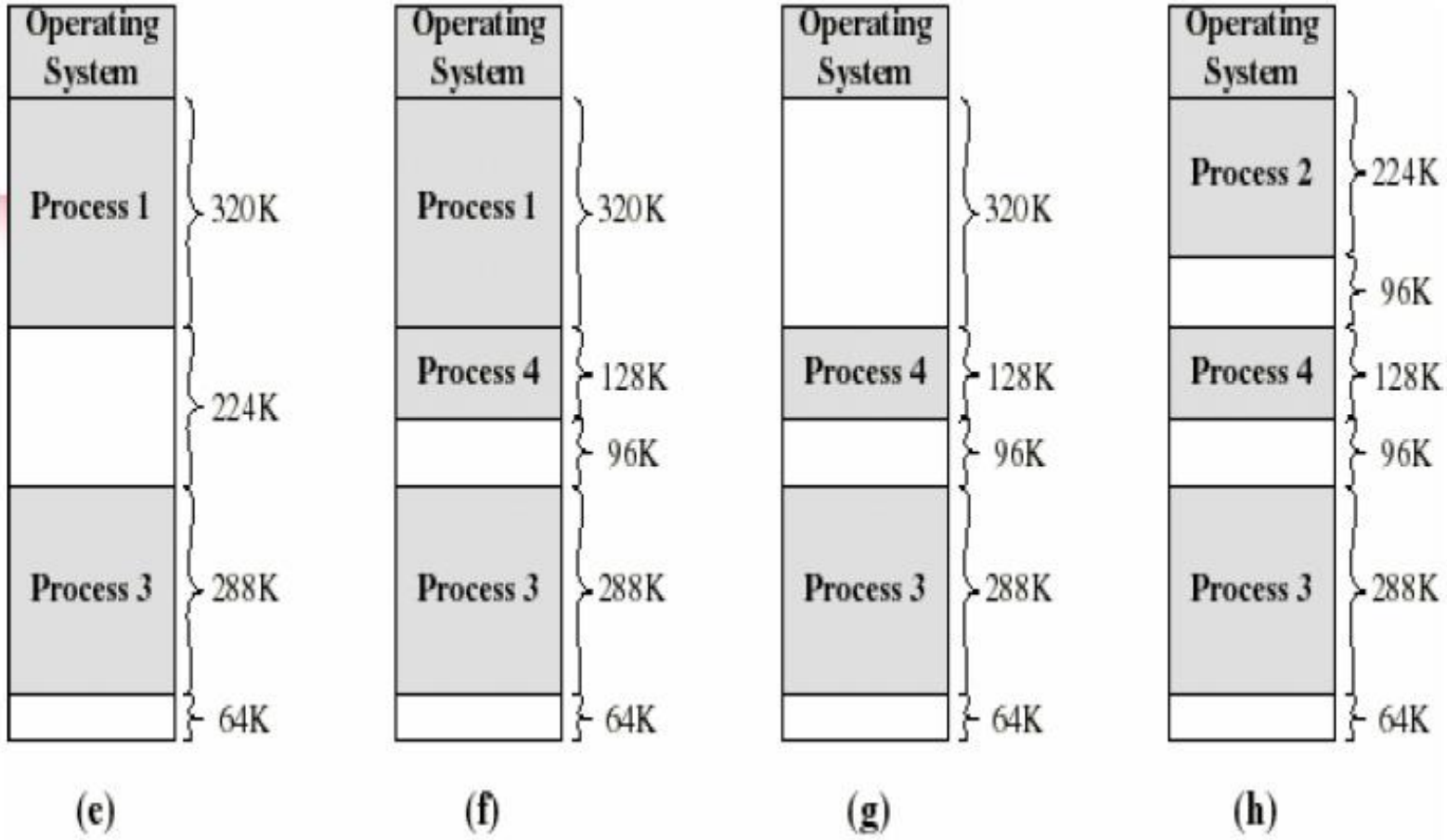
2.2.2 Structuration par Partitionnement variable (3)

- chaque partition corresponde à un processus chargé en MC, sa taille est égale donc à la taille de son processus.

  - Taille(espace alloué) = Taille(processus chargé)
    →Pas de fragmentation interne

  - Lorsque le processus se termine, il libère sa partition
    →une nouvelle partition libre.

2.2.2 Structuration par Partitionnement variable (4)



2.2.2 Structuration par Partitionnement variable (5)

- Remarques :
  - Pas de partitionnement initial
  - Une seule partition libre au départ
  - Partitionnement créé au fur et a mesure de l'allocation
  - Partitions créées correspondent aux processus chargés en MC
  - Pas de fragmentation interne :Taille(espace alloué) = Taille(processus chargé)

2.2.2.1 Algorithmique d'allocation (1)

Principe

- Processus P de taille T(P) à charger en MC →Trouver un espace égal à sa taille→Allocation immédiate ou différée

- Terminaison des processus :
  - Libèrent leurs partitions→Existence de plusieurs partitions libres

- *choix d'une partition libre ?*

2.2.2.1 Algorithmique d'allocation (2)

Algorithmes

- *First Fit : « première zone libre » on prend la première partition libre de taille suffisante (qui peut contenir le* processus qu'on désire charger).

- *Best Fit: « meilleur ajustement » on alloue au processus la partition de taille suffisante et qui* donne le plus petit résidu.

- *Worst Fit: Le « pire ajustement »* - on choisi la

plus grande : Partition qui donne le plus grand résidu

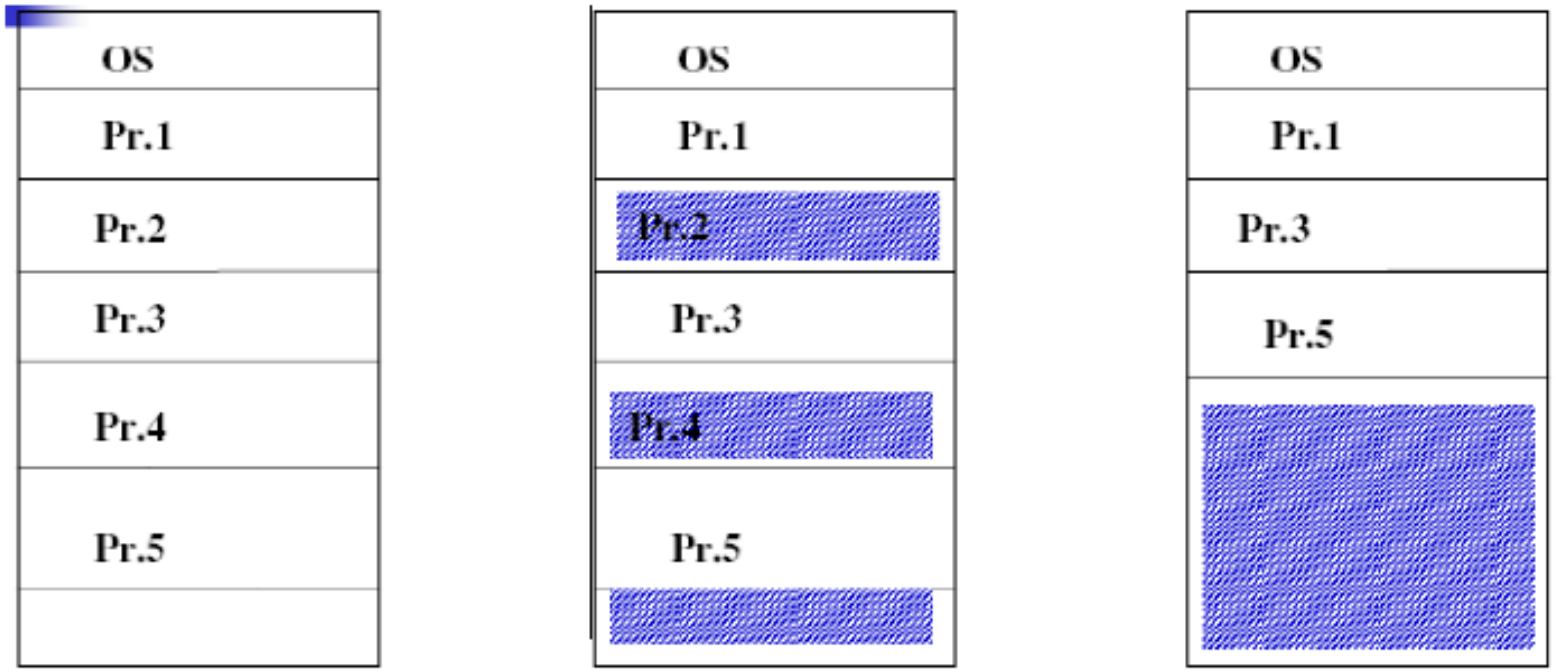
2.2.3 Problème (1)

- Supposons qu'on a un processus P de taille T(P) à charger en MC mais, aucun espace contigu libre ne peut supporter P : tous les espaces libres ont des tailles < à T(P). Cependant la somme des espaces libres est >> T(P).

→Problème de la fragmentation externe.

- **Solution:** Déplacer les zones allouées pour obtenir des partitions libres contiguës,
- Celles-ci seront ensuite fusionnées pour former une seule partition libre.
- Cette technique est appelée Compactage.

2.2.3 Problème (2)



a)

b) Libération espace

c) Compactage

Principe de compactage

- Déplacer les processus en MC :
  - Utiliser la technique de swap (c'est une partie de Disk)
- Regrouper tous les fragments libres:
  - Nécessite des swap
- Créer une partition libre
- Réalisée périodiquement ou sur demande