Chapitre5 : Gestion de la mémoire

5.1- Introduction. :

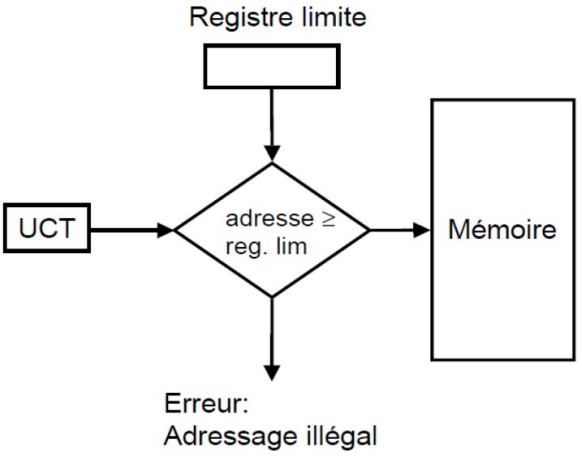
La mémoire principale est le lieu où se trouvent les programmes et les données quand le processeur les exécute. La nécessité de gérer la mémoire de manière optimale est toujours fondamentale, car elle n'est, en général, jamais suffisante en raison de la taille continuellement grandissante des programmes.
Les fonctionnalités attendues d'un gestionnaire efficace de la mémoire sont les suivantes :

- Connaître les parties libres de la mémoire.
- Allouer de la mémoire aux processus en évitant autant que possible le gaspillage.
- Récupérer la mémoire libérée par la terminaison d'un processus.
- Offrir au processus des services de mémoire virtuelle. de taille supérieure à celle de la mémoire physique disponible.

5.2- Stratégies d'allocation de la mémoire :

5.2.1- Une seul zone contiguë (Système mono tâche) : Dans ce cas, la mémoire est divisée en deux zones contiguës. L'une est allouée en permanence au système d'exploitation. La deuxième zone est réservée pour le chargement d'un seul processus. La gestion de la mémoire s'avère simple car le système d'exploitation doit garder trace de deux zones mémoires.
L'inconvénient majeur de cette stratégie est la mauvaise gestion de la mémoire car un seul processus occupe la totalité de l'espace mémoire.

Pour protéger le code et les données du système d'exploitation des changements que peuvent provoquer les programmes des utilisateurs, il y a un registre limite qui contient l'adresse à partir de laquelle commencent les instructions et données des programmes des utilisateurs. Chaque adresse (instruction ou donnée) générée par un programme est comparée avec le registre limite. Si l'adresse générée est supérieure ou égale à la valeur du registre, on l'accepte, sinon on refuse l'accès et on interrompt le processus avec un message d'accès illégal de mémoire.



5.2.2 Partitions multiples : Cette stratégie constitue une technique simple pour la mise en œuvre de la multiprogrammation. Cette technique nécessite la présence de plusieurs processus en mémoire. La mémoire est donc partagée entre le système d'exploitation et plusieurs processus. Il se pose cependant le problème suivant : ***comment organiser la mémoire de manière à faire cohabiter efficacement plusieurs processus tout en assurant la protection des processus?***

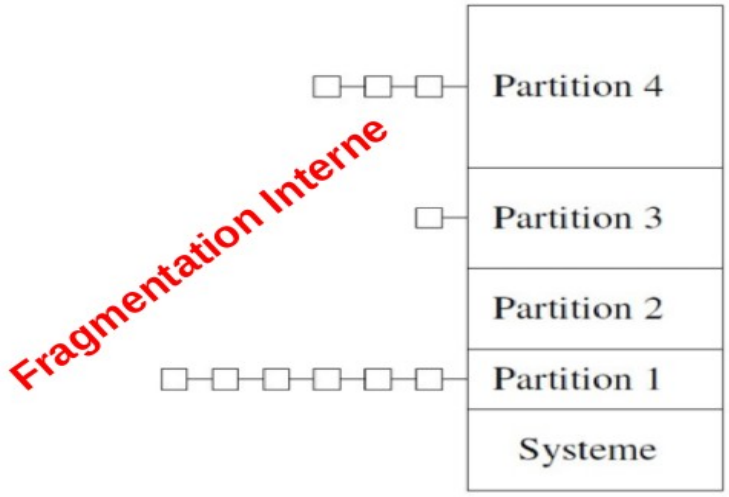
Deux cas sont alors à distinguer :

5.2.2.1 Partitions multiples statiques : Selon cette stratégie, l'espace mémoire est divisé en plusieurs partitions de tailles fixes. Le nombre et la taille de chaque partition sont fixés à la génération du système. Le statut de chaque partition est détenu par une table de description des partitions. Cette stratégie génère le problème de la fragmentation interne (voir 4.3).

a)- Chargement des programmes:

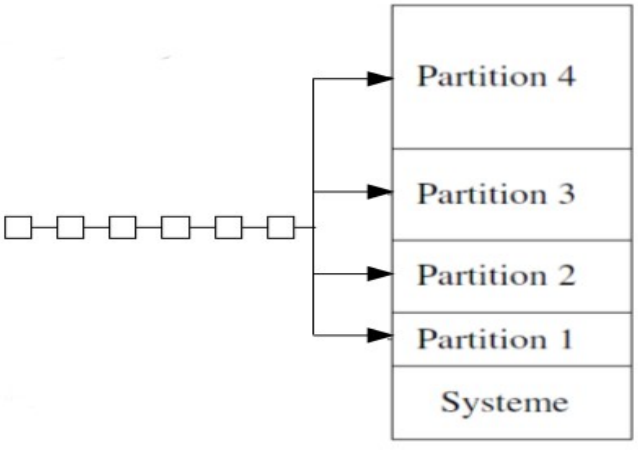
al.)- Programme absolu :

Un programme est absolu quand la liaison entre les objets du programme et les adresses mémoire est statique. Dans ce cas, le programme est lié à une partition donnée et ne peut s'exécuter en dehors de cette partition. Le gestionnaire de la mémoire associe à chaque partition une file des programmes en attente d'exécution.



a2)- Programme relogeable :

Un programme est dit relogeable quand la liaison entre les objets du programme et les adresses mémoire est virtuelle (dynamique). Dans ce cas, le programme n'est lié à aucune partition et **peut** être chargé et exécuté dans la première partition libre et de taille suffisante. Le gestionnaire de la mémoire associe une seule file de programmes en attente.



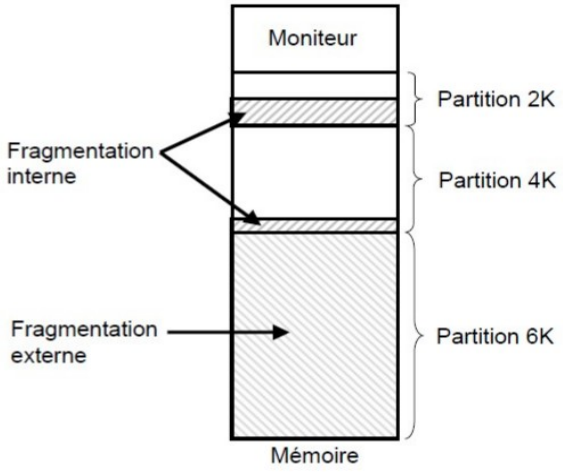
5.2.2.2 Partitions multiples dynamiques: Selon cette stratégie, la mémoire est partitionnée dynamiquement selon la demande. A chaque processus est allouée une partition exactement égale à sa taille. Quand le processus termine son exécution, sa partition est récupérée par le système pour être allouée à un autre programme complètement ou partiellement selon la demande.

5,2.2.2.1- Stratégies de placement : le placement des processus dans les partitions se fait elon un certain nombre de stratégies possibles :

- **First Fit « Premier qui convient » :** Dans cette stratégie, la table des partitions est triée ar ordre des adresses croissantes. Pour l'allocation d'une partition, la recherche commence à artir de la partition libre de plus basse adresse et continue jusqu'à la rencontre d'une partition ont l'a taille est au moins égale à celle du programme en attente.
- **Best Fit « meilleur qui convient » :** Dans cette stratégie, la tables des partitions est triée Ir tailles croissantes. Pour l'allocation d'une partition, la recherche commence à partir de la rtition libre de plus petite taille et continue jusqu'à la rencontre d'une partition dont la taille ;t au moins égale à celle du programme en attente.
- **Worst Fit « pire qui convient » :** Dans cette stratégie, on choisi la partition libre de plus •ande taille. On doit parcourir toute la table des partitions sauf si elle est trièe.

3- Fragmentation mémoire : On distingue deux types de fragmentation :

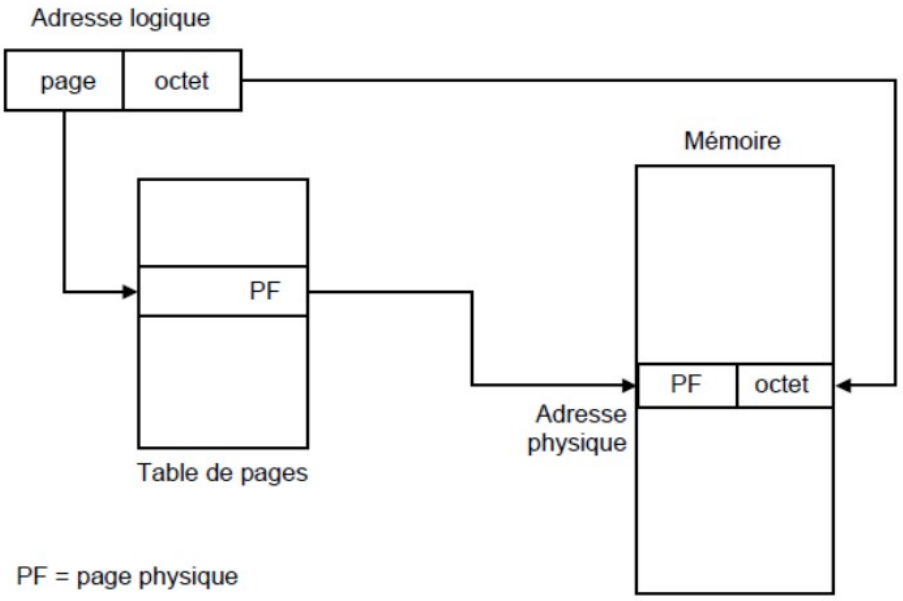
- **Fragmentation interne :** Lorsqu'une partition mémoire de taille N est alloué à . processus de taille M<N. La partie non occupée par le processus (de taille N-M) est pelée fragmentation interne. C'est-à-dire, un espace libre gaspillé à l'intérieur de la rtition.
- **Fragmentation externe :** La fragmentation externe se produit lorsqu'un)cessus de taille M cherche une partition libre de taille suffisante et que toutes les partitions ces sont de taille P_i avec Pi<M.



5.4- Le compactage (défragmentation) : Afin de pouvoir résoudre le/problème de la fragmentation avec ces deux types, le gestionnaire de la mémoire utilise la technique de compactage (défragmentation) qui consiste à déplacer le contenu de la mémoire vers un sens afin d'aboutir à un seul espace libre contiguë (un seul bloc).

5.5- Pagination : C'est une stratégie qui consiste à diviser l'espace d'adressage d'un processus en partitions égales appelées ***pages logiques.*** De même, l'espace mémoire est divisé en blocs de même taille appelés ***pages physiques ;*** ce qui facilitera la correspondance d'une page à un bloc.

La correspondance entre page logique et page physique est transparente à l'utilisateur. Elle est réalisée selon le schéma suivant :



Chaque adresse générée par le processeur est divisée en deux parties : un numéro de page (p) et un déplacement page (d). Le numéro de page est utilisé comme index dans la table des pages du processus qui contient l'adresse de base de chaque page dans la mémoire physique. Cette adresse de base est combinée au déplacement page pour définir l'adresse mémoire physique.qui sera envoyé à la mémoire.

La pagination permet d'éliminer le problème de fragmentation externe. Le problème de fragmentation interne reste toujours présent.

Remarque : Si P est la taille de la page et U une adresse logique. L'adresse paginée (p,d) est déduite à partir des formules suivantes :

p=U/P.(division entière).

d=U%P. (reste de la division).

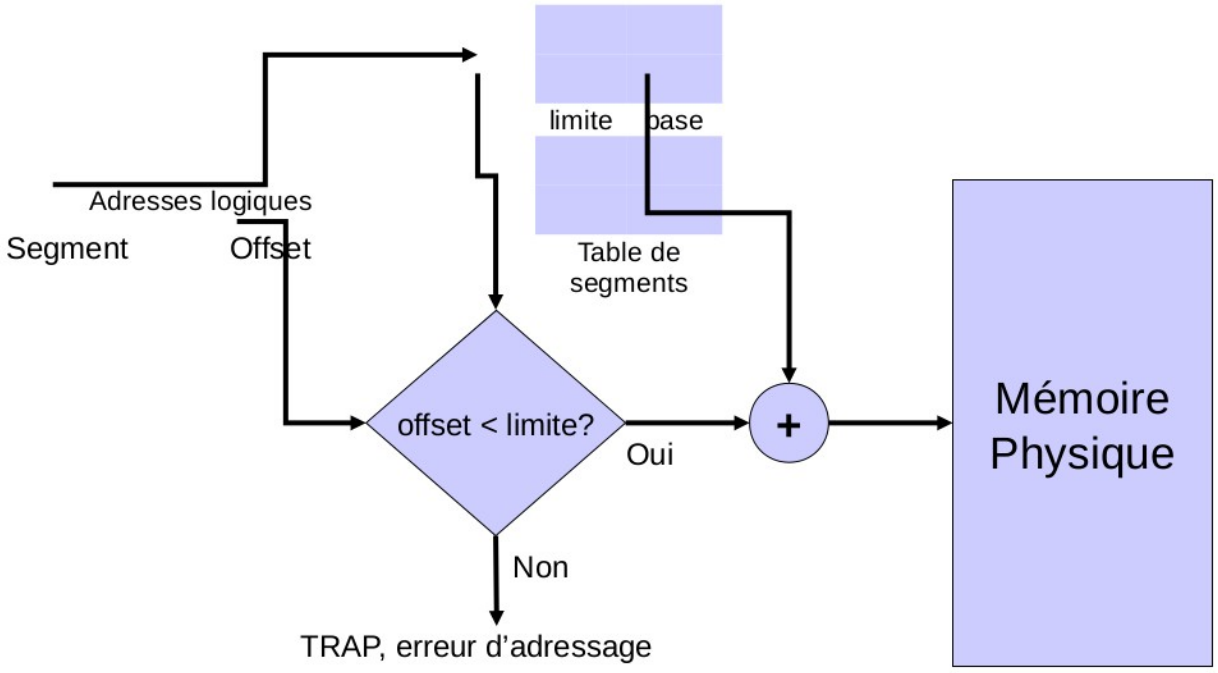
5.6- Segmentation :

La segmentation découpe l'espace d'adressage d'un processus en plusieurs segments à l'intérieur desquels les adresses sont relatives au début des segments. Cette technique prend en considération la vue utilisateur de la mémoire. En effet, un processus est composé d'un ensemble d'unités logiques :

- **Les différents codes :** le programme principal, les procédures, les fonctions bibliothèques.
- **Les données initialisées.**
- **Les données non initialisées.**
- **Les piles d'exécution.**

L'idée de la technique de segmentation est d'avoir un espace d'adressage à deux dimensions. On peut associer à chaque unité logique un espace d'adressage appelé segment. L'espace d'adressage d'un processus est composé d'un ensemble de segments de tailles différentes. Ces segments sont de tailles différentes (Fragmentation externe).

La correspondance entre une adresse logique et une adresse physique est réalisée selon le schéma suivant :



Une adresse logique est composée d'un numéro de segment (s) utilisé comme index dans la table des segments et d'un déplacement (d). Chaque entrée de la table des segments donne la base du segment et sa limite dans la mémoire. Le déplacement dans le segment doit être compris entre 0 et la limite du segment. Dans le cas contraire, une erreur d'adressage se produit. Pour retrouver l'adresse physique, on ajoute le déplacement à la base du segment.

La segmentation permet d'éliminer le problème de fragmentation interne. Le problème de la fragmentation externe reste toujours présent.

5.7- Segmentation paginée :

La segmentation peut être combinée avec la pagination. Chaque segment est composé d'un ensemble de pages. Les adresses générées par les compilateurs et les éditeurs de liens, dans ce cas, sont alors des triplets :
<numéro du segment, numéro de page, déplacement dans la page>
Le schéma de traduction d'adresses par segmentation paginée est montré sur la figure suivante :