|  |  |  |
| --- | --- | --- |
| •••••••• | C:\Users\HP\Desktop\LogoUASZ.jpg | •••••••• |

**Système d’Exploitation**

\_\_\_                                                         **I                        \_\_\_**

## **Gestion de la memoire**

**Sommaire**

❶**Monoprogrammation**

❷ **Multiprogrammation**

**Avec partitions fixes**

**Avec partitions variables**

**Algorithmes de sélection de partition**

❸ **La mémoire virtuelle**

**Pagination**

**Segmentation**

## **Introduction**

**En multiprogrammation** le SE charge plusieurs processus en RAM. La **Gestion de la Mémoire (GE)** est du ressort du système de gestion de la mémoire ou gestionnaire de la mémoire qui est parti intégrante du noyau du SE. Les principales **fonctionnalités de GE sont:**

* Connaître les parties libres de la RAM
* Allouer la RAM au processus en évitant autant que possible le gaspillage
* Récupérer la mémoire libérée après la fin d’un processus
* Offrir aux processus une mémoire virtuelle supérieure à celle de la RAM au moyen des techniques de va-et-vient et de pagination

## **Monoprogrammation**

Pour un système monoprocesseur et **monotâche** le SE se trouve au niveau des adresses basses de la RAM. Pour les système embarqués(consoles de jeux,téléphones mobiles ect.), le SE est dans la ROM (voir **Figure1**).



**Figure1: Emplacement du SE**

En monoprogrammation on un aà la fois un un seul processus en RAM. Si la taille du processus est supérieur à celle de la RAM, on utilisation des segments de recouvrements (overlay). Le programme est composé d’une zone permanente et un ensemble d’overlays interchangeables correspondants le plus souvent aux grandes fonctionnalités du programme comme le montre la **Figure2**.



Figure2: Méthode des segments de recouvrement

## **multiprogrammation**

En multiprogrammation, on peut avoir à la fois plusieurs processus en RAM. Ainsi chaque processus est chargé dans un emplacement de la RAM appelé partition. On distingue deux types de multiprogrammation selon le caractère des partitions: la multiprogrammation avec partitions fixes et la multiprogrammation avec partitions variables.

**III.1. Multiprogrammation avec partitions fixes**

La RAM est divisée en partitions de **taille identique**. L’espace non utilisé à la fin d’une partition est perdu: **fragmentation interne**.



**Figure3: Fragmentation interne**

Pour réduire la **fragmentation interne**, La RAM est divisée en partitions de **tailles inégales.** Ainsi la gestion des processus plus compliquée. On peut avoir deux type de file d’attente.

**- File d’attente pour chaque partition:** un nouveau processus est placé dans la file d’attente de la plus petite partition pouvant le contenir

Un problème possible est que les grandes partitions peuvent être vides, et les petites partitions pleines.

- **File unique:** dès qu’une partition se libère, on y place le plus grand processus de la file d’attente qui peut y tenir. Ce qui sésavantage les petites tâches.

D’autres algorithmes cherchent la première tâche pouvant se tenir dans la partition.

 

**Figure4: File d’attente pour chaque partition** **Figure5: File unique**

**III.2. Multiprogrammation avec partitions variables**

Le SE alloue une zone mémoire à un processus selon son besoin. Un problème possible est **fragmentation externe. Le** c**ompactage**  peut être une solution pour rassembler les zones libérées avec l’espace non alloué. La surcharge due aux déplacements des processus constitue un inconvenient.



**Figure6: Fragmentation externe**

Une autre possibilité est de gérer la mémoire libre par des **mécanismes** et **algorithms.**

**III.3. Mécanismes et algorithmes de gestion des zones libre**

Lorsque plusieurs zones mémoires sont libres, le SE doit utiliser des mécanismes pour identifier les zones libres. Parmi ces mécanisme on a:

-La gestion par tables de bits

- La gestion par listes chaînées

- La gestion par subdivision

Le SE doit également pouvoir utiliser un algorithme pour sélectionner la zone mémoire dans laquelle le processus sera chargé, et pour cela on a des algorithmes comme:

* L’algorithme de la première zone libre (Fisrt Fit)
* L’algorithme de la zone libre suivant (Next Fit)
* L’algorithme du meilleure ajustement(Best Fit)
* L’algorithme du plus grand résidu(Worst Fit)

**III.3.1. La gestion par tables de bits**

La mémoire est divisée en unités par et on fait correspondre chaque unité dans une table de bits comme le montre la **Figure7**.



Figure7: Table de bits

Dans la table de bi on a: 0=libre, 1=occupée

Pour allouer k unités contigües selon l’algorithme(FF, BF, WF,NF),le SE doit parcourir la table de bits à la recherche de k zéros consécutifs.

**III.3.2. La gestion par listes chaînées**

Chaque segment est représenté par son adresse de début et sa taille, et

deux bits (bit d’état occupé (O), Libre (L)).



**Figure8: Liste chainées**

**III.3.2. La gestion par subdivision**

La gestion par subdivision (buddy system ou allocation des copains) propose un compromis entre l’allocation de **taille fixe** te celle de **taille variable**. Le SE ne manipule que des blocs de mémoire dont la taille est égale à une puissance de 2. Initialement, il existe un seul blocs dont la taille est égale à celle de la mémoire. Un processus reçoit une unité de mémoire qui est la plus petite puissance de 2 supérieure à sa taille (**voir Figure8**)



**Figure8: Mécanisme de gestion par subdivision**

**III.4. Algorithmes de sélection des partitions**

**III.4.1. Algorithme de la première zone libre(First Fit)**

Le processus est alloué à la première zone mémoire libre rencontrée et suffisamment grand pour le contenir. L’algorithme favorise les zones en début de mémoire.

**III.4.2. Algorithme de la zone libre suivante (Next Fit)**

Cette algorithme se veut équitable en commençant la recherche de la zone à allouer partir de la zone dernièrement allouée et non en début de mémoire. Aucune zone libre n’est favorisée.

**III.4.3. Algorithme du meilleur ajustement (Best Fit)**

Le processus est alloué à la plus petite zone libre suffisamment grand pour le contenir.

Il nécessité de parcourir intégralement les zones libres de la mémoire. Il établit une correspondance presque parfaite entre la zone mémoire et la taille du processus. La partie restante de la zone allouée est trop petite, quasiment inutile.

**III.4.4. Algorithme du plus grand résidu (Worst Fit)**

Cet algorithme évite la création de zones petites et inutiles en choisissant la plus grande zone restante . Il nécessité de parcourir intégralement les zones libres de la mémoire.

## **Pagination**

**VI.1. Fonctionnement**

La pagination est l’allocation de zones de mémoire non contigües pour un même processus

La mémoire physique est découpée en zones de **taille fixe: cadres de page.** Les processus divisés en blocs: **pages** (de même taille que les cadres)**.** La taille d’une page est une puissance de 2. Une  **Adresse logique** est composée: numéro de page concaténé (‖) avec le déplacement dans la page.Une **Table de pages** fait la liaison entre numéro de page et le cadre de page (une table par processus) voir **Figure9 et Figure10**. Ainsi **l**e SE stocke la table de pages ou de segments en mémoire. Sur certains systèmes, un registre de gestion de mémoire pointe vers la table de page ou de segment du processus élu.

****

**Figure9: Pagination**

****

**Figure10: Exemple de pagination**

Pour chaque processus, une **table de pages** stocke le numéro du cadre de page alloué à chaque page. L’adresse physique est générée en concaténant le déplacement de la page au numéro du cadre de page extrait de la table de pages comme le montre le schéma de la Figure11.



**Figure11: Calcul d’adresse en pagination**

Le défaut de pages est l’accès à une page non présente en mémoire physique**. Dans ce cas** le SE choisit un cadre de page et écrit si nécessaire son contenu en mémoire virtuelle puis

charge en mémoire la page qui faisait défaut. Ceci implique un algorithme pour le choix de la page à décharger. Parmi ces algorithmes on a: First In First Out,

**IV.2. FIFO**

La page la plus ancienne sera supprimée en premier. Un problème possible: elle peut être la page la plus utilisée. Le FIFO est rarement utilisé pour le remplacement de pages.

**IV.3. LRU: (Least Recently Used)**

La page la moins récemment utilisée sera déchargée

Evidemment celles les plus utilisées durant les dernières instructions seront certainement les plus demandées dans le future.

**VI.4. Non récemment utilisée: NRU (Not Recently Used)**

L’algorithme choisit au hasard une page de classe basse. Les pages sont classées selon le schéma suivant:

* Classe 0: non lue, non écrite
* Classe 1: non lue, écrite
* Classe 2: lue, non écrite
* Classe 3: lue, écrite

**IV.5. LRU: (Least Frequently Used)**

La page la moins fréquemment utilisée sera déchargée. On associe un compteur à chaque page, et à chaque interruption, on ajoute (incrémente) le compteur. Ainsi la page avec le plus petit compteur est remplacée.

## **V. segmentation**

La segmentation divise le programme en blocs plus petits appelés **segments** et alloués indépendamment à la mémoire. Contrairement à la pagination, les segments sont de **taille variable**. Le SE conserve une **liste de zones de mémoire libres** à allouer aux segments. Une **Adresse logique** est compose du numéro de segment plus (+) le déplacement dans le segment. La

**Table de segments** est composée du N° segment comme indexe et de l’adresse de départ du segment en mémoire RAM et de la taille du segment. La **Taille des segments** doit être

une puissance de 2.

Pour chaque processus, une **table de segments** stocke l’indexe du segment (numéro), son adresse départ, sa taille. L’adresse physique est générée en ajoutant le déplacement à l’adresse de départ du segment extrait de la table de segments comme le montre la Figure12.



**Figure12: Calcul d’adresse en segmentation**

## **VI. segmentation avec pagination**

Dans le cas de la segmentation avec pagination, pour chaque processus, une **table de segments** stocke l’indexe du segment (numéro), son adresse départ, sa taille

L’adresse physique est générée en ajoutant le déplacement au numéro de cadre de page extrait de la table de pages comme le montre la **Figure13**.



**Figure12: Calcul d’adresse en Segmentation paginée**