BEN HAMOUDA Amel
HARDY Elise
Groupe 3

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Rapport
Simulation de gestion de mémoire virtuelle

Programmation Système

BEN HAMOUDA Amel HARDY Elise

Groupe 3

BEN HAMOUDA Amel
HARDY Elise

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

Sommaire

BEN HAMOUDA Amel
HARDY Elise

UNIVERSITÉ PARIS-EST MARNE-LA-VALLÉE

I. Objectif du projet

L'objectif du projet est d'étudier les mécanismes à l' œuvre pour la gestion des pages mémoires. On devait donc réaliser un programme capable de simuler la gestion d'un espace mémoire virtuel bien supérieur à notre espace réel.

Pour cela, on a du simuler la gestion de la mémoire paginée, et donc fournir un service capable de mettre en réserve et de rappeler des pages selon les demandes de l'utilisateur.

II. Élaboration du projet

A. Modules

Pour remplir l'objectif de ce Projet on a donc utilisée 6 modules :

→ argument : qui gère toute la partie des options en ligne de commande ;
→ fifo : qui effectue le remplacement FIFO ;
→ lru : qui effectue le remplacement LRU ;
→ deuxiemeChance : qui effectue l'algorithme de la Deuxième Chance ;
→ structure : qui gère toute la partie « *mécanique* » du projet avec des structures et des manipulations de liste, ...
→ main : qui assemble tout le programme.

B. Choix de programmation

1.Général

Pour tous les algorithmes nous avons décidées d'utiliser des listes chaînées et non des tableaux pour faciliter les insertions et retraits.

Nous avons aussi créer une structure pour permettre de sauvegarder toutes les informations nécessaires ainsi que toutes les listes :

```
typedef struct allElement {
    Liste *l1;
    Liste *ldern;
    Liste *ltrie;
    int *taberr;
    int sizetab;
    int max;
    int nbespace;
    int algo;
} AllElement;
```

La liste l1 représente les pages visibles et demandés
La liste ldern est une pile, celle-ci permet de savoir quelle pages doit être remplacé (le premier chaînons est la page à remplacer :la plus ancienne)
la liste ltrie représente les pages en réserve, celles qui ont déjà été appelées au moins une fois.
Le tableau taberr permet de connaître la position d'une page.
L'indice i du tableau comprend la position de la page i.
0 si la page n'a jamais été appelées, 1 si elle est déjà présente dans l1 et 2 si elle est en réserve.
Ce tableau évite de devoir re-parcourir la liste à chaque fois.

Les entiers suivant représentes des données essentielle comme le nombre de pages, la taille mémoire réelle, l'algorithme utilisée,…

A un moment nous avions pensé à faire une liste trié pour ne par avoir à parcourir la liste. Il aurait suffit de comparer page demandé avec celle de la liste et si la page était plus petite que celle de la liste cela aurait siginifer que la page n'était pas utilisée. Mais cette méthode était compliqué à maintenir d'ou l'idée du tableau avec les positions.

2. Remplacement FIFO

L'algorithme FIFO n'utilise pas la pile (liste ldern) pour connaître le nombre à retirer.

En effet l'utilisation d'un pointeur seul suffit. Ce pointeur se déplace d'un rang vers la droite à chaque insertions.

3. Remplacement LRU

L'algorithme LRU utilise quand à lui utilise la liste ldern car l'utilisation d'un pointeur ne suffit plus. On est obligé de connaître l'ordre d'arrivées des pages.

4. Algo de la 2ème chance version 1

Pour simuler le bit des pages nous avons utilisée un masque représenté par un char.

Cette algorithme est un mélange avec LRU il enlève le moins utilisé si son bit est à 0.

5.Algo de la 2ème chance version 2

Même principe que celui du dessus mais utilise FIFO on prend le premier rentré et on l'enlève si son bit est à 0.

III. Conclusion

Ce projet nous a permit de connaître et comprendre les différents algorithmes que l'on a implémenté. Et surtout de comprendre le mécanisme de la gestion des pages en mémoire.

IV. Annexe

A. Instructions de compilation

Pour compiler le programme avec les modules il suffit de lancer le **makefile** par la commande ***make*** dans le terminal.

B. Documentations

Pour la documentation voir le manuel de l'utilisateur.