Travail pratique #3 - IFT-2245

Julien Allard et Carlos Sanchez

27 Avril 2017

1 Introduction

En tant qu'utilisateur d'ordinateurs ou d'appareil mobile, on se plaint souvent de la lenteur des système ou du manque de mémoire. Il est intéressant de constater que beaucoup d'efforts ont été faits pour rendre la gestion de mémoire plus efficace. Grâce àce TP, nous avons eu la possibilité de coder un gestionnaire de mémoire virtuelle par pagination et de voir les enjeux d'un tel système. Danse ce rapport, nous allons décrire les choix de conceptions faits ainsi que les difficultés que nous avons rencontrés au cours de ce TP.

2 Rapport de Julien Allard

Conception

J'ai travaillé principalement sur la lecture et l'écriture dans le backing store, ainsi que sur le TLB. Pour le TLB, j'ai choisi d'utiliser un algorithme LRU pour faire chercher la prochaine victime. Cela semblait être un choix évident puisque nous avons implémenté le remplacement de page par la même façon.

Problèmes et surprises

La lecture et l'écriture sur le backing store était assez simple et directe. En connaissant l'arithmétique de pointeur, il était assez simple de faire les fonctions de lecture et écriture. Un problème que j'ai eu était que le fichier parse.y contenait une erreur lors de l'ouverture du fichier backstore au niveau de l'accès.

Pour le TLB, j'ai ajouté un tableau qui garde un compteur globale des accès pour chaque élément du TLB. Lorsqu'il faut choisir une entrée du TLB à évincer, on cherche la valeur minimale dans le tableau et l'index correspond à l'élément à enlever. Un problème de cette approche est que le nombre d'accès mémoire dans un vrai système est de l'ordre de milliers par secondes. Premièrement, c'est un approche qui coûte assez cher et de deux, on arrive au maximum d'un int très rapidement. Nous avons pensé à utiliser le temps en secondes, mais nous sommes tomber sur le problème que notre programme exécute plusieurs accès mémoire en une même seconde. Finalement, nous avons retourné avec notre idée du départ.

Un autre problème rencontré était que lorsqu'on trouvait une frame, on ne faisait pas de différence entre une frame qui était vide et une frame qui a été remplacé. Ceci causait des problèmes durant la gestion de la mémoire virtuelle. La solution implémentée est que l'on regarde séparément si une frame est libre, sinon on doit évincer une frame.

3 Rapport de Carlos Sanchez

Conception

De mon côté je me suis chargé de la gestion de la mémoire virtuelle. J'ai implémenté les fonctions pour gérer les frames de la mémoire physique avec un algorithme LRU, le même qu'on a utilisé pour le TLB, ainsi que les fonctions du page table.

Problèmes et surprises

Ma plus grosse surprise est à quel point le travail s'est déroulé sans problème majeur. Le fonctionnement d'une mémoire virtuelle était bien détaillée dans le livre et nous avions d'autre ressources dans le livre qui nous ont été suggéré dès le départ. La compréhension de l'énoncé était beaucoup plus facile et directe, nous savions quelle direction prendre lors de l'implémentation du travail.

Je crois aussi que d'avoir fait le travail précédent en C m'a aidé car les notions étaient fraîche dans ma mémoire et je n'ai pas été ralentit par les technicalités du langage et de l'environement de scriptage.

Un problème que nous avons rencontré cependant est l'ambiguitée présentée par le bit readonly du page table. Comme plusieurs personnes se sont questionné sur le forum du cours, nous n'étions pas certain de comment nous devions interpréter ce bit. Est-ce que c'était un bit de protection ou d'état? Rien dans l'énoncé ne nous donnait une idée sur les restrictions d'écriture ou de lecture des pages. Nous l'avons donc interprété comme bit d'état pour indiquer si la page a été lue seulement ou si on a écrit à l'intérieur.

4 Conclusion

En conclusion, ce travail pratique nous a appris sur la gestion mémoire. De plus, il nous a permis de raffiner nos connaissances avec le langage C, malgré qu'il y avait moins de gestion de mémoire.