Architecture des Ordinateurs & Systèmes d'Exploitation
TP 12

Objectifs : Observer, comprendre et commenter des exemples de programme VisualOS qui mettent en avant la hiérarchie de mémoire. Il faut tout de même noter que les exemples qui suivent sont basés sur les règles de fonctionnement propres à VisualOS.

Exercice 1

Lorsque le processus demande une page, on a un accès a la mémoire physique. Si on a un échec, on va chercher la page dans l'espace de swap pour la charger en mémoire physique.

Dans le cas d'un accès en lecture, la page est chargée et reste identique à celle qui est toujours dans l'espace swap, on a donc un affichage vert de la page en mémoire physique et de la case en mémoire virtuelle.

Dans le cas d'un accès en écriture, la page sera modifiée dés son arrivée dans la mémoire physique. La case en mémoire virtuelle référençant cette page sera donc en noir. Ceci veut dire que si on a besoin d'effacer la page, on doit la réécrire dans l'espace de swap. C'est là qu'est la différence avec l'accès en lecture : on a pas besoin d'aller réécrire dans l'espace swap puisque les pages sont identiques.

Exercice 2

On explique la présence de la barre rouge par le temps d'accès de la tête de lecture à la page (stockée en espace swap) requise par les processus.

La durée d'exécution est différente car les accès aux pages demandées par le processus 2 ne donnent que des échecs. Le processus 1 accède à la page 0 qui est toujours en mémoire physique, on ne perd pas de temps en accès à la swap. Pendant que les pages du processus 2 sont chargées en mémoire physique, du temps processeur est alloué au processus 1, qui se termine donc plus tôt.

Exercice 3

Il y a 40 cases en mémoire physique. On ne peut pas surcharger la mémoire virtuelle.

Quand la mémoire physique est pleine, on efface la plus ancienne page accédée (quel que soit le processus auquel elle est allouée) pour la remplacer par celle nécessaire.

Exemple 1 :

La partie rouge est plus grande après la surcharge qu'avant car à chaque page a été accédé en écriture donc à chaque fois qu'on appelle une page, elle a été modifié. On doit donc libérer une case : on réécrit dans l'espace swap la plus ancienne page, puis on charge la page demandée en mémoire physique dans la case libérée. On a donc un accès à l'espace swap pour écriture puis en lecture.

Exemple 2 :

Cas 1 : Le processus accède à des pages en lecture qui sont toujours en mémoire physique, donc les pages n'ont pas besoin d'être réécrites en espace swap si on a besoin des cases.

Le processus accède toujours à de nouvelles pages en écriture, donc il faut toujours libérer une case. On rencontre 2 cas : la case appartenait au processus 1, dans ce cas on va en espace swap pour lecture de la nouvelle page ; la case appartient au processus 2, on doit donc la réécrire, on va donc en espace swap pour écrire la plus ancienne, puis pour lire la nouvelle demandée par le processus 2.

Cas 2 : Dans cet exemple, le processus 2 fait un accès en lecture parmi tous les autres en écriture. Lorsque la case de cette page devra être libérée, on ira en espace swap pour la lecture de la nouvelle page sans écrire l'ancienne (accédée en lecture donc non-modifiée). L'anomalie n'est en fait qu'accès swap en lecture parmi d'autres en écriture – lecture.

L'écriture en espace swap ne se fait que lorsque la case de la page doit être libérée et lorsque cette page a été accédée en écriture.

La gestion de la mémoire est en écriture allouée et différée.