info524 : Systèmes d'exploitation
TD 3 et 4 : gestion de la mémoire

Pierre Hyvernat et Rodolphe Lepigre
Pierre.Hyvernat@univ-smb.fr
Rodolphe.Lepigre@univ-smb.fr

Question 1. Le temps d'accès à un mot est d'environ
- 2ns dans le cache,
- 10ns dans la RAM,
- 10ms sur le disque dur.

Quel est le temps d'accès moyen à un mot si l'accès au cache fonctionne dans 95% des cas et l'accès en RAM dans 99% des cas restant.

Exercice 1 : Allocation de mémoire

On suppose que l'état de la mémoire RAM est décrit par le tableau suivant :

. **10** .	10	..... **20**	30	. **10** .	5	..... **30**	20	. **10** .	15	..... **20**	20

(Les tailles sont en Ko, et les blocs en **gras** sont utilisés, alors que les autres sont libres.)

Des requêtes d'allocation de mémoire arrivent dans cet ordre là : 20 Ko, 10 Ko, 5 Ko et 25 Ko.

Question 1. À quelles adresses sont alloués les blocs si on utilise la politique "First Fit" ?

Question 2. À quelles adresses sont alloués les blocs si on utilise la politique "Best Fit" ?

Question 3. À quelles adresses sont alloués les blocs si on utilise la politique "Worst Fit" ?

Question 4. À quelles adresses sont alloués les blocs si on utilise la politique "Next Fit" ?

Question 5. Pour chacune de ces politiques, chercher un exemple de demande d'allocation / désallocation qui est visiblement inefficace.

Question 6. En partant d'une mémoire libre de 1 Mo, utilisez le système des zones siamoises ("buddy system") pour allouer la mémoire des processus suivants :
- processus A, requête de 50 Ko
- processus B, requête de 150 Ko
- processus C, requête de 60 Ko
- processus D, requête de 60 Ko
- processus E, requête de 60 Ko
- processus D, fin
- processus C, fin
- processus E, fin
- processus A, fin
- processus F, requête de 125 Ko
- processus G, requête de 150 Ko
- processus F, fin
- processus G, fin
- processus B, fin

(Rappel : 1 Ko = 2^{10} o = 1024 o et 1 Mo = 2^{20} o = 1024 Ko.)

Exercice 2 : Mémoire virtuelle, pagination

Question 1. Pour simplifier les calculs, nous allons utiliser des pages de taille 1000 octets. Les cadres ont donc une taille de 1000 octets également. Normalement, la taille d'une page serait une puissance de 2 : 4096 octets en général.

La table de pages est la suivante :

Page	In/Out	Cadre
0	In	20
1	Out	22
2	In	37
3	In	15
4	Out	97
5	Out	15
6	In	03
7	In	11

Parmi les adresses virtuelles suivantes, lesquelles génèrent un défaut de page ?
- 6451
- 5421
- 123
- 3156

Pour celles qui ne génèrent pas de défaut de page, quelle est l'adresse physique référencée ?

Question 2. On suppose que le système ne comporte que quatre cadres de page :

cadre	Chargement	Dernière référence	Modification	Référence
0	167	374	1	1
1	321	321	0	0
2	254	306	1	0
3	154	331	0	1

Quelle page serait remplacée par :
- l'algorithme FIFO,
- l'algorithme LRU ("Least Recently Used"),
- l'algorithme NRU ("Not Recently Used"),
- l'algorithme de la deuxième chance ?

Question 3. Un système possède 3 cadres de page et fait référence aux pages suivantes :

0, 9, 0, 1, 8, 1, 8, 7, 8, 7, 1, 2, 8, 2, 7, 8, 2, 3, 8, 3

Combien de défauts de page sont générés si on utilise :
- le remplacement FIFO,
- le remplacement LRU,
- le remplacement optimal ?

Question 4. (Paradoxe de Belady)

On suppose qu'un programme référence les pages suivantes :

3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4

En utilisant le remplacement FIFO, combien de défauts de pages sont générés si on dispose de :
- trois cadres de page,
- quatre cadres de page ?

Qu'en pensez-vous ?

Question 5. On regarde le morceau de code C suivant :

```
char T[N];
for (int i=0 ; i<N ; i=i+M)
  T[i] = 0;
for (int j=0 ; j<N ; j=j+M)
  T[j] = T[j]+1;
```

Le système utilise la pagination avec des pages de 4Ko et un TLB de 64 pages.

Combien de pages sont-elles nécessaires pour stocker le tableau `T` ?

On s'intéresse à ce qui se passe lors de la seconde boucle :
- si `M` vaut `1`, à partir de quelle valeur de `N` aura t'on des *soft-miss* ?
- que se passe-t'il lorsque la valeur de `M` augmente ?

Question 6. Supposons qu'un ordinateur utilise des adresses 64 bits et que la table des pages soit une table à 3 niveaux. Les adresses virtuelles sont découpées en 4 parties :
- une partie sur a bits pour le premier niveau de table,
- une partie sur b bits pour le deuxième niveau de table,
- une partie sur c bits pour le troisième niveau de table,
- une partie sur o bits pour l'offset dans la page.

On a $a + b + c + o = 32$.

Quelles relations y a t'il entre la taille d'une page et a, b, c et o ?

Même question pour le nombre de pages disponibles.

Question 7. Un ordinateur utilise des adresses 32 bits. Chaque page fait 8Ko et la table des pages est à un seul niveau et chaque ligne de la table est constituée d'un mot de 32 bits.

Lorsqu'un processus se met en exécution, la table est copiée en mémoire principale à la vitesse d'un mot toute les 100 ns. Chaque processus reste sur le processeur pendant 100ms, en comptant le chargement de la table).

Combien de temps le processus passe-t'il à changer la table ?

Question 8. Donnez un exemple d'application où il est préférable de ne pas avoir de mémoire virtuelle.