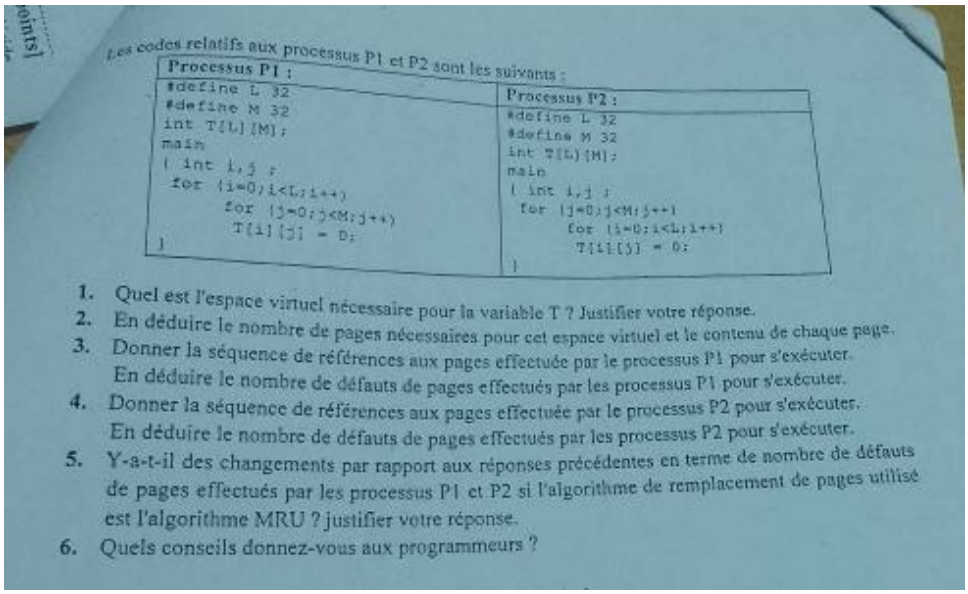
Examen 2020-2021

Exercice 3



Les codes relatifs aux processus P1 et P2 sont les suivants :

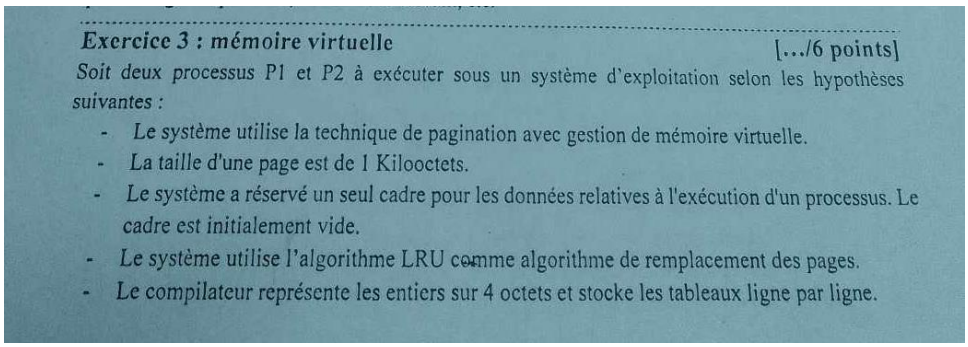
Processus P1 :

```
#define L 32
#define M 32
int T[L][M];
main
{ int i,j ;
 for (i=0;i<L;i++)
      for (j=0;j<M;j++)
       T[i][j] = 0;
}
```

Processus P2 :

```
#define L 32
#define M 32
int T[L][M];
main
{ int i,j ;
 for (j=0;j<M;j++)
      for (i=0;i<L;i++)
       T[i][j] = 0;
}
```

1. Quel est l'espace virtuel nécessaire pour la variable T ? Justifier votre réponse.
2. En déduire le nombre de pages nécessaires pour cet espace virtuel et le contenu de chaque page.
3. Donner la séquence de références aux pages effectuée par le processus P1 pour s'exécuter.
   En déduire le nombre de défauts de pages effectués par les processus P1 pour s'exécuter.
4. Donner la séquence de références aux pages effectuée par le processus P2 pour s'exécuter.
   En déduire le nombre de défauts de pages effectués par les processus P2 pour s'exécuter.
5. Y-a-t-il des changements par rapport aux réponses précédentes en terme de nombre de défauts de pages effectués par les processus P1 et P2 si l'algorithme de remplacement de pages utilisé est l'algorithme MRU ? justifier votre réponse.
6. Quels conseils donnez-vous aux programmeurs ?



Exercice 3 : mémoire virtuelle **[…/6 points]**

Soit deux processus P1 et P2 à exécuter sous un système d'exploitation selon les hypothèses suivantes :

- *Le système utilise la technique de pagination avec gestion de mémoire virtuelle.*
- *La taille d'une page est de 1 Kilooctets.*
- *Le système a réservé un seul cadre pour les données relatives à l'exécution d'un processus. Le cadre est initialement vide.*
- *Le système utilise l'algorithme LRU comme algorithme de remplacement des pages.*
- *Le compilateur représente les entiers sur 4 octets et stocke les tableaux ligne par ligne.*

1. Quel est l'espace virtuel nécessaire pour la variable T ? Justifiez votre réponse.

La variable T est un tableau de dimensions 32×32 d'entiers. Chaque entier occupe 4 octets. Donc, la taille totale de T est :

$$32×32×4 \text{ octets} = 4096 \text{ octets} = 4 \text{ Ko}$$

2. En déduire le nombre de pages nécessaires pour cet espace virtuel et le contenu de chaque page.

La taille d'une page est de 1 Ko. Donc, pour un tableau de 4 Ko, nous avons besoin de 4 pages. Les pages sont organisées de la manière suivante :

- **Page 0 :** T[0][0] à T[7][31]

- **Page 1 :** T[8][0] à T[15][31]
- **Page 2 :** T[16][0] à T[23][31]
- **Page 3 :** T[24][0] à T[31][31]

3. Donner la séquence de références aux pages effectuées par le processus P1 pour s'exécuter.

Le processus P1 accède aux éléments du tableau ligne par ligne. La séquence de références aux pages est donc :

- Accès à la page 0 (T[0][0] à T[7][31])
- Accès à la page 1 (T[8][0] à T[15][31])
- Accès à la page 2 (T[16][0] à T[23][31])
- Accès à la page 3 (T[24][0] à T[31][31])

4. En déduire le nombre de défauts de pages effectués par le processus P1 pour s'exécuter.

Comme le système utilise un seul cadre et l'algorithme LRU, chaque accès à une nouvelle page entraîne un défaut de page. Ainsi, pour P1 :

- Début : Page 0 -> défaut de page
- Changement à Page 1 -> défaut de page
- Changement à Page 2 -> défaut de page
- Changement à Page 3 -> défaut de page

Nombre total de défauts de pages = 4

5. Donner la séquence de références aux pages effectuées par le processus P2 pour s'exécuter.

Le processus P2 accède aux éléments du tableau colonne par colonne. La séquence de références aux pages est donc plus fréquente entre les pages :

- Accès alterné aux pages 0, 1, 2, 3 de façon répétitive pour chaque colonne.

6. En déduire le nombre de défauts de pages effectués par le processus P2 pour s'exécuter.

Pour chaque itération sur `i` et `j`, P2 accède à un nouvel élément de chaque page. Cela entraîne des défauts de page à chaque nouvel accès à une page différente :

- Début : Page 0 -> défaut de page
- Changement à Page 1 -> défaut de page
- Changement à Page 2 -> défaut de page

- Changement à Page 3 -> défaut de page

Chaque cycle de **`i`** et **`j`** répète ce processus :

Nombre total de défauts de pages pour N = 32 : $32 \times 4 = 128$32×4=128

7. Y a-t-il des changements par rapport aux réponses précédentes en termes de nombre de défauts de pages effectués par les processus P1 et P2 si l'algorithme de remplacement de pages utilisé est l'algorithme MRU ?

Avec l'algorithme MRU (Most Recently Used), le comportement peut légèrement varier, mais globalement pour un seul cadre :

- Pour P1, chaque nouvelle page entraînerait toujours un défaut de page.
- Pour P2, étant donné la nature de ses accès, l'algorithme MRU peut potentiellement entraîner plus de défauts si les pages récemment utilisées sont souvent les plus à remplacer.

8. Quels conseils donnez-vous aux programmeurs ?

- Pour améliorer les performances en termes de défauts de pages, il est crucial de comprendre les motifs d'accès de la mémoire.
- Accéder aux éléments de tableaux de manière séquentielle par ligne (comme P1) plutôt que par colonne (comme P2) peut réduire le nombre de défauts de pages.
- Optimiser le code pour une meilleure localité spatiale peut améliorer l'efficacité de la gestion de mémoire.