EXAMEN
(aucun document autorisé)

Exercice 1 : Questions de cours (5 pts - 20 minutes)

1.1 Qu'est-ce qu'une *mémoire paginée* ? À quoi sert-elle ? Citer ses avantages et inconvénients.
1.2 Qu'est-ce un Systèmes d'Exploitation ? Citer cinq de ses fonctions majeurs.
1.3 Qu'est-ce que la *TLB* ? Que contient la *TLB* et à quoi ça sert ?
1.4 Comment marche l'algorithme *LRU* ? Pourquoi cette technique fonctionne ?
1.5 Qu'est-ce qu'un *écroulement* ? Comment peut-on l'éviter ?

Bonus : Définir un défaut de page. Quelles sont les actions entreprises par le SE dans ce cas ?

Exercice 2 : MÉMOIRE VIRTUELLE (6 pts - 25 minutes)

La mémoire est segmentée et paginée. Les pages de segments et les cases de mémoire les contenant font 256 mots. Il y a au plus 256 segments et les segments sont limités à 256 pages.

Chaque mot de la table des segments contient un doublet (**NMS, @TP**) où **NMS** est le nombre de mots du segment et **@TP** le numéro de la case contenant sa table des pages. Si **TS[i].@TP = 0**, le segment **S[i]** est absent, c'est à dire que sa table des pages n'est pas présente en mémoire physique. Par convention : le segment **S[0]** est la table des segments elle même, **TS[0].NMS** contient le nombre de segments et **TS[0].@TP =** adresse de **TS**.

Chaque mot de la table des pages d'un segment **S[i]** contient le numéro de la page en mémoire où elle se trouve ainsi que les droits d'accès **(L**ecture, **E**criture et e**X**écution**)** et le bit de présence.

Une adresse virtuelle est notée **(s, d)** où **s** est le numéro de segment et **d** est l'adresse dans le segment.
Une tâche a 4 segments, y compris le segment 0. Le segment 1 a 500 mots, le segment 2 en a 2500, le segment 3 en a 5000.

Le table des segments, ainsi que les tables des pages des deux segments **S[1]** et **S[3]** sont seules chargées respectivement dans les cases de numéro 30, 31 et 32.

Les seules pages chargées sont : la page 0 du segment **S[1]** dans la case 40 en **(L,E)**, la page 2 du segment **S[2]** dans la case 50 en **(L,X)**, les pages 1, 2, 3, 4 du segment **S[3]** dans les cases 65, 66, 67, 68 en **E**. Les autres pages du segment **S[3]** en **L**.

2.1 Donner les valeurs contenues dans la table des segments et dans les tables des pages.

2.2 Donner l'adresse réelle correspondant à l'adresse virtuelle (3,780).

2.3 Que se passe-t-il pour des accès en écriture aux adresses virtuelles (1,512) , (3,1024), (3,1280)?

PS: *Les multiples de 256*
1 : 256; 2 : 512; 3 : 768; 4:1024; 5 : 1280; 6 : 1536; 7 : 1792; 8 : 2048...

Exercice 3 : Synchronisation par sémaphores (6 pts – 25 minutes)

Une illustration classique du problème de la synchronisation est celui du salon de coiffure. Dans le salon de coiffure, il y a un coiffeur C, un fauteuil F dans lequel se met le client pour être coiffé et N sièges pour attendre.

- S'il n'a pas de clients, le coiffeur C somnole dans le fauteuil F.
- Quand un client arrive et que le coiffeur C dort, il le réveille, C se lève. Le client s'assied dans F et se fait coiffer.
- Si un client arrive pendant que le coiffeur travaille :
  - si un des N sièges est libre, il s'assied et attend,
  - sinon il ressort.

Il s'agit de synchroniser les activités du coiffeur et de ses clients avec des sémaphores. Les sémaphores utilisés sont initialisés ainsi :

```
SEM * SCF=CS(0);
SEM * SP=CS(0);
SEM * SX=CS(1);

Coiffeur (){
    while (1){
      P (SP);
      P (SX);
      Attend = Attend -1;
      V (SCF);
      V (SX);
      Coiffer();
    }
}
```

```
Client (){
    P (SX);
       if (Attend < N){
          Attend = Attend + 1;
          V (SP);
          V (SX);
          P (SCF);
          SeFaireCoifferEtSortir();
       }
       else {
         V (SX);
         Sortir();
       }
}
```

a- Détailler le fonctionnement du coiffeur et de ses clients tels qu'ils sont représentés par les deux fonctions `Coiffeur` et `Client`.
b- Quel est le rôle de chacun des sémaphores `SCF`, `SP` et `SX` ?

Exercice 4 : Création de processus (4 pts - 20 minutes)

4.1 Qu'affiche chacun des segments des programmes suivants et par quel processus ? Quel est le nombre de processus créés dans chaque cas ? Donnez l'arbre des processus crées. On suppose que les appels systèmes **`fork`** et **`exec`** réussissent.

Processus 1 :

```
for (i=1; i<=3; i++){
  pid = fork();
  if (pid == 0) execlp("/bin/ls","ls",0);
  else printf("%d\n",i);
}
```

Processus 2 :

```
for (i=1; i<=3; i++){
  pid = fork();
  if (pid != 0) printf("%d\n",i);
}
```