Système d'Exploitation
- le 13 Mars 2018, Durée : 1h30min

Notes de cours et Diapositives mises à Dispositions autorisées
Toute réponse doit être justifiée
Machines à mémoire Interdites

Exercice 1 *Questions de Cours*

1. Rappelez les deux caractéristiques principales d'un système d'exploitation. **(2 pts)**
2. Expliquez en quoi le mécanisme des appels systèmes permet de protéger le matériel, le système d'exploitation et les programmes utilisateurs. **(2 pts)**
3. Déduire de la question précédente pourquoi `read` doit être implémenté en fonction système et non en fonction simple. **(2 pts)**
4. Laquelle de ces instructions devrait être seulement exécutée en mode noyau : **(2 pts)**
   (a) masquage des interruptions,
   (b) lire l'heure,
   (c) modifier l'heure,
   (d) modifier la pagination.

Exercice 2 *Producteur/Consommateurs*

Supposons que l'on dispose de plusieurs processus qui accèdent en même temps à une base de données par des opérations de lecture et écriture. Plusieurs lecteurs peuvent accéder à la base de données en lecture, mais lorsqu'un processus écrit, il est le seul à pouvoir accéder à la base (aucune lecture ni autre écriture ne sera possible tant qu'il n'a pas fini). Pour implémenter ceci, la solution suivante a été proposée.

```
semaphore mutex =1;
semaphore db = 1;
int rc = 0; //nombre de processus qui lisent ou veulent lire

// fonction exécuter pour les demandes de lecture
char* lecture (char *requete) {
      P(&mutex);
      rc+=1;
      if(rc==1)
        P(&db);
      V(&mutex);
      // opérations de lecture
      // ...
      P(&mutex);
      rc-=1;
      if(rc==0)
        V(&db);
      V(&mutex);
      // retour de valeur lecture pour traitement
}
```

```
//fonction exécutée pour les demandes d'écriture
void write (char *requete) {
     P(&db);
     // opérations d'écriture
     V(&db);
}
```

1. Expliquez pourquoi cette solution est correcte. **(3 pts)**
2. Expliquez pourquoi cette solution pénalise ceux qui veulent écrire. **(1 pt)**
3. Pour ne pas pénaliser ceux qui veulent écrire, on se propose de suspendre les lecteurs qui font une demande de lecture après une demande d'écriture non satisfaite. Expliquez pourquoi ceci permet de ne pas pénaliser ceux qui veulent écrire. **(2 pts)**
4. Implémentez la solution de la question (3). **(2 pts)**

Exercice 3 *Synchronisation*

1. Qu'est ce qu'une section critique ? **(1 pt)**
2. Disjkstra a proposé les deux solutions suivantes comme étant des solutions logicielles au problème d'accès à une section critique entre deux processus 1 et 2. Il a précisé qu'elles n'étaient pas tout à fait correctes. Expliquez pourquoi en donnant un exemple d'exécution ne respectant pas l'exclusion mutuelle pour la section critique. **(4 pts)**

Solution 1 :

```
boolean flag[2]; \\ Variable partagée
flag[0] = flag[1] = 0; \\ Initialisation
proc (int i) {\\ i=1 ou 2 suivant le processus qui execute proc
    while(flag[(i+1) mod 2])
         ; \\ Attendre que flag[(i+1) mod 2 ]== FALSE
    flag[i] = TRUE;
    <section critique>
    flag[i] = FALSE;
}
```

Solution 2 :

```
boolean flag[2]; \\ Variable partagée
flag[0] = flag[1] = 0; \\ Initialisation
proc (int i) { \\ i=1 ou 2 suivant le processus qui execute proc
     flag[i] = TRUE;
     while(flag[(i+1) mod 2 ])
          ; \\ Attendre que flag[(i+1) mod 2 ]== FALSE
     <section critique>
     flag[i] = FALSE;
}
```