Université A. Mira Bejaia --/ 09/2015
Faculté des Sciences Exactes Durée 2 h 00
Département d'Informatique
Année d'étude : 3eme Année Licence Académique

# Corrigé de l'Examen de Rattrapage de Systèmes d'Exploitation 2

**Exercice 1 : (4 pts)**
On considère le programme parallèle suivant :

```
var n : entier init 0 ; /* n entier initialisé à 0 */
begin
     parbegin
          Processus P1
          begin
               n := n + 1
          end
          Processus P2
          begin
               n := n - 1
          end
     Parend
end
```

1) Quel le résultat de son exécution dans les cas suivants :
   a) on considère les opérations sur la variable n sont indivisibles
   b) les opérations sur n ne sont pas indivisibles : elles peuvent être décomposées.
2) Dans le cas b) comment faire pour que la mise-à-jour de `n` se fasse en exclusion mutuelle.

**Solution :**
1)
a) Dans le cas de l'indivisibilité de l'accès à n, le résultat obtenu (après la fin de l'exécution des deux processus P1 et P2) est n = 0. (1 pt)
b) Dans le cas où l'on peut décomposer les opérations sur n le résultat n'est pas unique suivant la séquence d'exécution considérée on peut obtenir n = 0 ou n = 1 ou n = -1. (1pt)

En effet :

```
Processus P1
Début
(1) LOAD R1,n /* charger le registre R1 par n */
(2) ADD R1,1 /* additionner 1 à R1 */
(3) STORE n,R1 /* ranger (R1) dans la variable n */
Fin
Processus P2
Début
(1') LOAD R2,n /* charger le registre R2 par n */
(2') SUB R2,1 /* soustraire 1 de R2 */
(3') STORE n,R2 /* ranger (R2) dans la variable n */
Fin
```

Avec la séquence *(1), (2), (3), (1'), (2'), (3')* on obtient n = 0
Avec la séquence *(1), (1'), (2'), (3'), (2), (3)* on obtient n = 1

Avec la séquence *(1), (1'), (2), (3), (2'), (3')* on obtient n = -1

2) Pour que la mise-à-jour de `n` se fasse en exclusion mutuelle, il existe de nombreux moyens.

Un de ces moyens est l'utilisation de sémaphore : (0.5 pt)

```
var n : entier init 0 ; (0.25pt)
s : sémaphore init 1 ;(0.25pt)
Début
Parbegin
Processus P1
     Début
     P(s) (0.25pt)
          LOAD R1,n
          ADD R1,1
          STORE n,R1
     V(s) (0.25pt)
     Fin

Processus P2
     Début
     P(s) (0.25pt)
          LOAD R2,n
          SUB R2,1
          STORE n,R2
     V(s) (0.25pt)
     Fin
Parend
Fin
```

De cette manière le seul résultat qu'on peut obtenir est n = 0 ; car on exclue les séquences d'exécutions pouvant aboutir à n = 1 ou n = -1 : dès qu'un processus fait `P(s)`, l'autre processus ne pourra plus accéder à n jusqu'à ce que celui qui a fait `P(s)` la première fois libère l'accès à n en faisant `V(s)`.

**Exercice 2 : (6 pts)**

(Cet exercice c'est vous qui la proposez et je ne possède pas de solution)

**Exercice 3 : (4pts)**

Soit N processus Pi (i=1..N) et un processus Ps. Les processus Pi (i=1..N) remplissent une zone tampon pouvant contenir M objets, un seul à la fois étant autorisé à déposer son objet. Le processus Pi qui remplit la dernière case du tampon active le processus Ps qui fait alors l'impression de tous les objets déposés dans le tampon. Durant cette impression, les processus Pi (i=1..N) ne sont pas autorisés à accéder au tampon.

**Question**) Ecrire les algorithmes des processus Pi (i=1..N) et Ps.

**Solution :**
Si on note : nm = nombre d'objets contenus dans le tampon, le schéma des processus considérés peut s'exprimer comme suit : (solution préliminaire)

```
Processus Pi (i=1..N)
Debut
    <Produire un objet>
    <si le tampon est occupé Alors attendre
        sinon Bloquer l'accès au tampon>
    <Déposer l'objet>
    Nm :=nm+1
    <si nm=M alors activer Ps
        Sinon libérer l'accès au tampon>
Fin
```

```
Processus Ps
Debut
    Cycle
        <attendre jusqu'à être réveillé
        par un des processus Pi>
        <Imprimer tous les objets>
        Nm :=0
        < Libérer l'accès au tampon>
    FinCycle
Fin
```

On peut assimiler l'attente d'un processus au blocage de celui-ci dans une file de sémaphore. Soit Spriv un sémaphore privé au processus Ps qui y se bloquera en attente d'être réveillé ; et mutex un sémaphore d'exclusion mutuelle pour l'accès au tampon.
On peut écrire les algorithmes des processus Pi et Ps comme suit :

```
var Spriv,mutex : sémaphore init 0,1 ;
nm : entier init 0 ;
```

```
Processus Pi (i=1..N)
Debut
    <Produire un objet>
    P (mutex)
    <Déposer l'objet>
    Nm :=nm+1
    <si nm=M alors V(Spriv)
            Sinon V(mutex)
Fin
```

```
Processus Ps
Debut
    Cycle
        P(Spriv)
        <Imprimer tous les objets>
        Nm :=0
        V (mutex)
    FinCycle
Fin
```

**Barème approximative** : l'initialisation sur (0.5 pts), l'algorithme du prcessus Pi (1.5 pts), l'algorithme du processus Ps (1.5pts)

# Question de Cours (6pts)

**Q1)** Vrai ou faux
a) faux (0.5)
b) Vrai (0.5)

**Q2)** On propose les fonctions d'exclusion mutuelle suivantes, écrites en C, pour un système à deux processus:

```
int[] sc = {0,0};
void entrer_section_critique(int id) {
```

```
        sc[id]=1;
        while(sc[1-id]);
}
void sortir_section_critique(int id) {
        sc[id]=0;
}
```

a) Expliquez quel est le problème de cette solution. Illustrez sur un exemple concret.

*Réponse : Rien n'empêche deux processus de se marquer tous les deux en SC et donc de rentrer dans une boucle d'attente in_nie. P0 appelle SC et e_ectue sc[0]=1, puis est interrompu, P1 fait de même, donc attend P0, qui reprend la main et attend P1...* (1pt)

b) Proposez une solution simple pour éviter de problème.

*Réponse : Comme proposé dans le cours, on peut rajouter une variable de tour de priorité. Lors de l'entrée en SC, on donne le tour à l'autre puis on met son booléen SC à vrai. On attend alors tant que à la fois l'autre à la main et il est en SC. Si l'autre fait pareil, il nous donnera la main et donc on n'attendra plus.* (1 pt)

**Q3)** Dans un système informatique, on dispose de trois fichiers F1, F2 et F3 et de trois processus dont les programmes A, B et C ont les structures suivantes :

| Programme A | Programme B | Programme C |
|---|---|---|
| Actions A1 | Actions B1 | Actions C1 (lire *F3*) |
| Actions A2 (lire *F2*) | Actions B2 (écrire *F3*) | Actions C2 |
| Actions A3 | Actions B3 (lire *F1*) | Actions C3 |
| Actions A4 (écrire *F3*) | Actions B4 | Actions C4 (écrire *F2*) |
| Actions A5 | Actions B5 | Actions C5 |

Chaque fichier ne peut ni être lu et modifié en même temps, ni modifié par plusieurs processus en même temps.

a) Donner pour chaque fichier les sections critiques de A, B et C.
b) En déduire les sections en exclusion mutuelle.

**Réponse :**

a)
- F1 : pas de section critique car il n'est pas accédé en écriture (0.5 pt)
- F2 : A2 et C4(0.5 pt)
- F3 : A4, B2 et C1(0.5 pt)

b) Sections en exclusion mutuelle :
- A2 et C4 (F2) (0.75pt)
- A4, B2 et C1 (F3) (0.75pt)