Ministère de l'enseignement supérieur et de la recherche scientifique

Université de Jijel

Faculté des Sciences exactes et Informatique

Département d'informatique

Solutions des exercices

SYSTÈMES D'EXPLOITATION II

(Partie 1 : Chapitre 1 et 2)

Exercice 2 (série 2) :

- **Question 1 :** On utilise une seule variable booléenne M telle que M = vrai si un des processus se trouve dans sa section critique, faux sinon.
- **Correction :**

```
/*déclarations et initialisation des variables globales*/
int M = 0 ;//M est a faux
/* etat d'occupation de la ressource*/
```

```
Tâche P0
While (TRUE) do{
/*entree_SC */
while (M );/* attente active*/
M=1;//M est a vrai
Section_critique();
/*sortie_SC*/
M=0;
Section_non_critique() ;
}
```

```
Tâche P1
While (TRUE) do {
/*entree_SC */
while (M );/* attente active*/
M=1;
Section_critique();
/*sortie_SC*/
M=0;
Section_non_critique() ;
}
```

- Pour cette solution la condition de l'exclusion mutuelle n'est pas respectée : Une exécution « entrelacée » de l'entrée en Section critique, par les deux tâches entraîne le non-respect de l'exclusion mutuelle, entree_SC n'est pas atomique.

- **Question 2 :**On utilise une variable commune unique T telle que T = i si et seulement si la tâche Pi estautorisée à entrer en sa section critique (i = 0, 1).
- **Correction :**

/*déclarations et initialisation des variables globales*/
int T = 0 ; /* T vaut 0 ou 1 tâche autorisée à entrer en section critique*/

```
Tâche P0
While (True) do{
/*entree_SC */
while (T==1 );/* attente
active*/
Section_critique();
/*sortie_SC*/
T=1;
Section_non_critique() ;
}
```

```
Tâche P1
While (True) do{
/*entree_SC */
while (T==0 );/* attente active*/
Section_critique();
/*sortie_SC*/
T=0 ;
Section_non_critique() ;
}
```

- Avec cette solution la condition de déroulement n'est pas respectée : Cette solution règle le problème de l'exclusion mutuelle, une seule tâche peut entrer e SC(valeur de T). C'est un fonctionnement à l'alternat donc si un processus tombe en panne hors section critique, l'autre processus sera bloqué. La condition c) n'est pas respectée.

- **Question 3 :** On utilise c(i) variable booléenne attachée au processus Pi (i = 1, 2) C(i) = vrai si Pi est dans sa section critique ou demande à y entrer C(i) = faux si Pi est hors

de sa section critique Pi peut lire et modifier c(i), peut lire seulement C(j) si j différent de i.

- **Correction :**

```
/*déclarations et initialisation des variables globales*/ int C[2]={0,0};
```

```
Tâche P0
While (True) {
/*entree_SC */
while (C[1]);/* attente active*/
C[0]=1 ;
Section_critique();
/*sortie_SC*/
C[0]=0;
Section_non_critique() ;
}
```

```
Tâche P1
While (True) {
/*entree_SC */
while (C[0]);/* attente active*/
C[1]=1 ;
Section_critique();
/*sortie_SC*/
C[1]=0;
Section_non_critique() ;
}
```

- Avec cette solution L'exclusion mutuelle n'est pas garantie, chaque tâche peut trouver la variable C de l'autre tâche à faux et décider son entrée en SC.Une autre solution pourrait être que chaque processus positionne sa variable à true, puis teste la variable de l'autre processus, dans ce cas on risque un blocage mutuel indéfini.(c-a-d l'absence de blocage n'est respectée).
- **Question 4 :** On peut obtenir une solution correcte en combinant les solutions précédentes et enintroduisant une variable supplémentaire T servant à régler les conflits à l'entrée de la section critique, T n'est modifiée qu'en fin de section critique.
- **Correction :**

```
/*déclarations et initialisation des variables globales*/
int C[2]={0,0} ;int T=0
```

```
Tâche P0
while (true) {
/*entree_SC */
C[0]=1 ;
while (C[1]) { /* P1 est en SC ou a
demande a entrer en SC*/
while (T==1) C[0]=0 ;/*P1 est
prioritaire, retrait de P0*/
C[0]=1 ;/*T=0 P0 recandidate*/
}
Section_critique();
/*sortie_SC*/
C[0]=0;
T=1 ;
Section_non_critique() ;
}
```

```
Tâche P1
While (true) {
/*entree_SC */
C[1]=1 ;
while (C[0]) { /* P0 est en SC ou a
demande a entrer en SC*/
while (T==0) C[1]=0 ;/*P0 est
prioritaire, retrait de P1*/
C[1]=1 ;/*T=1 P1 recandidate*/
}
Section_critique();
/*sortie_SC*/
C[1]=0;
T=0 ;
Section_non_critique() ;
}
```

EXERCICE 3 (Série 3) :

Il suffit d'utiliser un sémaphore initialisé à N

```
var S : sémaphore init N ; /* S.val=N */
Processus Client
Début
P(S) /* prendre un chariot */
Effectuer les achats
V(S) /* libérer le chariot */
Fin
```

Exercice 4 (le carrefour par sémaphore)(série 3) :

X1, X2, SF1, SF2 : sémaphore : 1 ,1,1,0 ;

Changement

```
{int Feu = 1;
while(1)
{sleep(Duree_du_feu);
if (Feu == 1)
{ P(SF1); V(SF2); Feu = 2;}
else
 { P(SF2); V(SF1); Feu = 1;}}
  }
```

Traversee1

```
{ P(SX1);
P(SF1);
Traversee();
V(SF1);
V(SX1);
}
```

Traversee2

```
{ P(SX2);
P(SF2);
  Traversee();
 V(SF2);
V(SX2);
}
```

Exercice 1 (coiffeur /client) (série 4) :

Sclient, Scoiff, Mutex : sémaphore := 0 ,0,1 ;

Attente : entier :=0 ; **N : const**

```
Coiffeur
While (true) do
{
P(Sclient) ;
P(Mutex) ;
Attente :=Attente-1 ;
V(Scoiff) ;
V(Mutex) ;
Couper-cheveux () ;
}
```

```
Client
While (true) do
{
P(Mutex) ;
If( Attente < N) then
{ Attente := Attente +1 ;
V(sclient) ;
V(Mutex) ;
P(Scoiff) ;
Obtenir-coupe
}
Else
V(Mutex) ;
}
```

Exercice 2 (Série 4) (producteur /consommateur par sémaphore) :

Spécification : Contexte commun tampon de N cases

```
producteur
while (true) do{
produire un message;
déposer un message;
}
```

```
producteur
while(true) do{
retirer un message;
consommer le message;
}
```

Solution par sémaphore :

N : Const

Mutex,Vide,Plein : **Sémaphore** : 1,N,0 ;

```
Producteur
While ( true) do
{
Produire –objet () ;
P(vide) ;
P(Mutex) ;
Mettre –objet() ;
V(Mutex) ;
V(Plein) ;
}
```

```
Consommateur
While (true) do
{
P(plein) ;
P(Mutex) ;
Retirer-objet() ;
V(Mutex) ;
V(Vide) ;
Consommer-objet() ;
}
```

Exercice 5 (série 3) : (producteur /consommateur par sémaphore)

S1,S2,S3,S4,mutex1,mutex2 : sémaphore init n1,0,n2,0,1,1 ; /* ni=taille de Ti, i=1,2 */

```
Processus P1
While (true) do {
<Produire un message>
P(S1) ;
P(mutex1) ;
<Dépôt du message dansT1> ;
V(mutex1)
V(S2) ;
}
```

```
Processus P2
While (true) do{
P(S2) ;
P(mutex1) ;
<Prélever un message deT1> ;
V(mutex1) ;
V(S1) ;
<Traiter le message> ;
P(S3) ;
P(mutex2) ;
<Dépôt du résultat dans T2> ;
V(mutex2) ;
V(S4) ;
}
```

```
Processus P3
While (true) do
{P(S4) ;
P(mutex2) ;
<Prélever un message de T2> ;
V(mutex2) ;
V(S3) ;
<Traitement du message> ;
}
```

Exercice 3 (série 4) (la rivière par moniteur) :

Type rivière : Moniteur ;

Var : nombreA : nat

nombreB : nat

suspendA: condition

suspendB: condition

procedure ArriveeA
if nombreB > 0
then wait suspendA
signal suspendA
end if
nombreA := nombreA + 1
end ArriveeA

procedure ArriveeB
if nombreA > 0
then wait suspendB
signal suspendB
end if
nombreB := nombreB + 1
end ArriveeB

procedure FinA
nombreA := nombreA - 1
if nombreA = 0 then signal
suspendB endif
end FinA

procedure FinB
nombreB := nombreB - 1
if nombreB = 0 then signal
suspendA endif
end FinB

Initialisation :

nombreA := 0 ;

nombreB := 0 ;

End rivière

process P_A
rivière .ArriveeA
LA TRAVERSEE DE A VERS B.
rivière.FinA
end P_A
process P_B
rivière .ArriveeB
LA TRAVERSEE DE B VERS A
rivière.FinB
end P_B