UNIVERSITE A.MIRA-BEJAIA
FACULTE DES SCIENCES EXACTES
DEPARTEMENT INFORMATIQUE

TRAVAUX DIRIGES
PROGRAMMATION SYSTEME
Série 1 : Section critique

Exercice 1 :

1) Quelles sont les conditions que doit vérifier une solution du problème de la section critique ?
Soient P1 et P2 deux processus s'exécutant en parallèle et accédant à une même section critique. Soit *libre* une variable partagée de type booléen initialisée à vrai. Considérons les codes des deux processus P1 et P2.

```
/* code de P1 */
while (1) {
<section non critique>
while (!libre) ;
      libre = 0;
<section critique>
      libre = 1;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
while (!libre) ;
      libre = 0;
<section critique>
      libre = 1;
}
```

2) Cette solution réalise-t-elle l'exclusion mutuelle des deux processus ?

On utilise maintenant une variable partagée *qui*, de type entier, initialisée par le numéro d'un processus (1 ou 2). Les codes de P1 et P2 sont :

```
/* code de P1 */
while (1) {
<section non critique>
while (qui != 1) ;
      qui = 1;
<section critique>
      qui = 2;
```

```
/* code de P2 */
while (1) {
<section non critique>
while (qui != 2) ;
      qui = 2;
<section critique>
      qui = 1;
```

3) Cette solution réalise-t elle l'exclusion mutuelle ?

Dans cette troisième tentative on utilise un tableau tbool partagé de deux booléens initialisé à {0,0}. Les codes de P1 et P2 sont :

```
/* code de P1 */
while (1) {
<section non critique>
while (tbool[1]) ;
tbool[0] = 1;
<section critique>
tbool[0] = 0;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
while (tbool[0]) ;
tbool[1] = 1;
<section critique>
tbool[1] = 0;
}
```

4) Montrer que cette solution ne réalise pas l'exclusion mutuelle.

Changeons les codes de nos deux processus :

```
/* code de P1 */
while (1) {
<section non critique>
tbool[0] = 1;
while (tbool[1]) ;
<section critique>
tbool[0] = 0;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
tbool[1] = 1;
while (tbool[0]) ;
<section critique>
tbool[1] = 0;
}
```

5) Monter qu'il y a interblocage.

Cette dernière solution est une combinaison des solutions précédentes. Elle consiste à utiliser un tableau de deux booléens tbool[2] = {0, 0} ; et un entier qui. Les codes de nos deux processus sont les suivants :

```
/* code de P1 */
while (1) {
<section non critique>
tbool[0] = 1;
qui = 0;
while (tbool[1] && qui == 0)
;
<section critique>
tbool[0] = 0;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
tbool[1] = 1;
qui = 1;
while (tbool[0] && qui ==
1) ;
<section critique>
tbool[1] = 0;
}
```

6) Montrer que cette solution est correcte. C'est-à-dire qu'elle réalise l'exclusion mutuelle et ne présente pas de cas d'interblocage.

Exercice2 : Donner la solution du problème de la section critique et qui vérifie les conditions d'exclusion mutuelle en utilisant les verrous puis les sémaphores. Quelle est la différence entre les deux outils ?

Exercice3 : Soit l'exécution parallèle des deux processus suivants :

```
ProcessusA :
Debut
     Faire toujours
          T1 ;
     Fait
fin
```

```
ProcessusB :
Debut
     Faire toujours
          T2 ;
     Fait
fin
```

Utilisez les sémaphores pour synchroniser les 2 processus pour que les taches T1 et T2 :

i) Ne s'exécutent pas simultanément.
ii) S'exécutent toujours dans l'ordre : T1T2T1T2T1T2...
iii) S'exécutent toujours dans l'ordre : T1T2T2T1T2T2T1T2T2...

-FIN-