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

Série 1 : Section critique

Exercice 1 :

1) Quelles sont les conditions que doit vérifier une solution du problème de la section critique ?
R : 4 conditions : i) Exclusion mutuelle, ii) déroulement, iii) pas de famine (pas d'interblocage)

2) Cette solution (une variable booléenne) réalise-t-elle l'exclusion mutuelle des deux processus ?
R : si les deux processus arrivent en même temps (cas simultannée), ils modifient en même temps la variable booléenne *libre* à vrai et entrent simultanément en SC. Donc 1ere condition n'est pas vérifiée.

3) Cette solution (une variable entière *qui*) réalise-t elle l'exclusion mutuelle ?
R : le cas initial ! supposons *qui =1* et P1 ne veut pas entrer en SC. P2 veut entrer en SC mais est bloqué car *qui==1* ! donc 2eme condition n'est pas vérifiée.

4) Montrer que cette solution (2 variables booléennes) ne réalise pas l'exclusion mutuelle.
R : le cas initial et simultanée ! les deux variables booléennes sont initialisée à 0. P1 et P2 vérifient chacun la variable de l'autre (0) et entrent simultanément en SC ! donc 1ere condition non vérifiée.

5) Monter qu'il y a interblocage.
R : cas simultanée ! P1 et P2 modifient en même temps chacun sa variable (1) puis vérifient en même temps chacun la variable de l'autre (1) et se bloquent mutuellement ! donc 3eme condition non vérifiée (interblocage).

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.
R : l'EM est vérifiée car si P1 arrive avant P2 (ou l'inverse) alors P2 sera bloqué à cause de la variable booléenne. Si P1 et P2 arrivent en même temps alors la variable *qui* laissera passer un seul processus. On peut aussi le démontrer par l'absurde :

$$\begin{cases} P1\ en\ SC \\ et \\ P2\ en\ SC \end{cases} \Rightarrow \begin{cases} \begin{cases} tbool[0] = vrai \\ et \\ tbool[1] \neq vrai\ ou\ qui \neq 0 \end{cases} \\ et \\ \begin{cases} tbool[1] = vrai \\ et \\ tbool[0] \neq vrai\ ou\ qui \neq 1 \end{cases} \end{cases} \Rightarrow \begin{cases} tbool[0] = vrai\ et\ tbool[0] = faux \\ et \\ tbool[1] = vrai\ et\ tbool[1] = faux \\ ou \\ qui = 1\ et\ qui = 0\ (impossible) \end{cases}$$

La dernière proposition est fausse donc la première est fausse aussi.
Le déroulement est vérifiée car si P1 ne veut pas entrer en SC alors la variable booléenne permettra à P2 d'entrer en SC. Ca suit le raisonnement précédent.
Il est impossible d'avoir un interblocage. Raisonnons par l'absurde.

$$\begin{cases} P1\ bloqué \\ et \\ P2\ bloqué \end{cases} \Rightarrow \begin{cases} tbool[1] = vrai\ et\ qui = 0 \\ et \\ tbool[0] = vrai\ et\ qui = 1 \end{cases} \Rightarrow \begin{matrix} tbool[1] = vrai\ et\ tbool[0] = vrai \\ et \\ qui = 0\ et\ qui = 1\ (impossible) \end{matrix}$$

Donc la 1ere proposition est fausse.

Exercice2 :

Solution Semaphores :	Solution Verrous :	Différence :

Shared semaphore S=1 ;
P(int i){
While (TRUE){
P(S);
SC;
V(S);
}}
	Shared Verrou V=ouvert ;
	P(int i){
	While (TRUE){
	Verrouiller(V);
	SC;
	Déverrouiller(V);
	}}
		Le verrou ne peut être utilisé que pour la SC.
		Le sémaphore, étant un distributeur de tickets, est utilisé pour résoudre beaucoup de problèmes de synchronisation.

Exercice3 :

i) Ne s'exécutent pas simultanément.
R : un seul sémaphore
Shared semaphore S=1 ;

```
ProcessusA :
Debut
      Faire toujours
       P(S) ;
            T1 ;
       V(S) ;
      Fait
fin
```

```
ProcessusB :
Debut
      Faire toujours
       P(S) ;
            T2 ;
       V(S) ;
      Fait
fin
```

ii) S'exécutent toujours dans l'ordre : T1T2T1T2T1T2...
R : deux sémaphores
Shared semaphore SA=1, SB=0 ;

```
ProcessusA :
Debut
      Faire toujours
       P(SA) ;
            T1 ;
       V(SB) ;
      Fait
fin
```

```
ProcessusB :
Debut
      Faire toujours
       P(SB) ;
            T2 ;
       V(SA) ;
      Fait
fin
```

iii) S'exécutent toujours dans l'ordre : T1T2T2T1T2T2T1T2T2...
R : deux sémaphores
Shared semaphore SA=1, SB=0 ;

```
ProcessusA :
Debut
      Faire toujours
       P(SA) ;
            T1 ;
       V(SB) ;
       V(SB) ;
       P(SA) ;
      Fait
fin
```

```
ProcessusB :
Debut
      Faire toujours
       P(SB) ;
            T2 ;
       V(SA) ;
      Fait
fin
```