

Série 2

11 avril 2005

A rendre le: 18 avril 2005

A rendre dactylographiée

Exercice 1 : Exclusion Mutuelle

Considérons la solution de l'exclusion mutuelle par attente active présentée dans la figure 1. Montrer que cette solution est fautive en construisant son diagramme d'états (seule la partie du diagramme montrant une violation de section critique est nécessaire).

Figure 1 Exclusion mutuelle par attente active

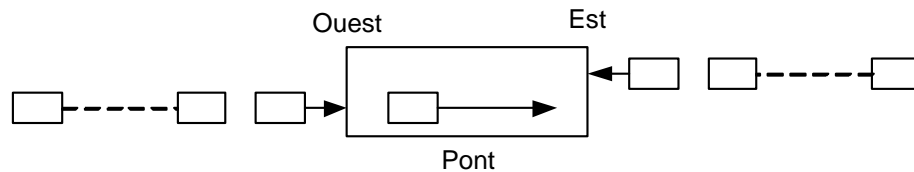
```
1: type noprocessus = 1 .. 2;
2: module sc;                                     {Section critique}
3: defines début, fin;
4: var
5:   demande: array noprocessus of boolean ;
6:   tour: noprocessus ;
7: procedure début(no: noprocessus)
8:   demande[no] := true ;                         {Annonce demande d'entrer en section critique}
9:   tour := no ;
10: repeat
11:                                     {On ne fait rien}
12:   until not demande[3 - no] or (tour = no)
13: end
14: procedure fin(no: noprocessus)
15:   demande[no] = false ;
16: end

17: Initialisation:
18:   demande[1] := false ;
19:   demande[2] := false ;
```

Exercice 2 : Sémaphores et compteurs de Robert

On considère un pont étroit où les véhicules circulent dans les deux sens mais ne peuvent pas se croiser. Le but est d'implémenter, à l'aide de *sémaphores*, un module qui exporte les procédures `entree_ouest()`, `sortie_est()`, `entree_ouest()` et `sortie_est()` qui respecte les contraintes suivantes :

On suppose un flot quelconque de véhicules arrivant dans les deux sens. De plus, on impose :



- Un seul véhicule à la fois se trouve sur le pont à un moment donné.
- Si des véhicules attendent des 2 côtés, ils traversent le pont en alternance (un véhicule *est* → *ouest*, puis un véhicule *ouest* → *est*, etc.)
- Si des véhicules ne sont présents qu'à une extrémité du pont, alors ils peuvent traverser le pont les uns après les autres jusqu'à ce qu'un véhicule se présente dans le sens opposé.

Il faut :

1. spécifier le problème à l'aide des compteurs de Robert.
2. implémenter, à l'aide de *sémaphores*, un module qui exporte les procédures `entree_ouest()`, `sortie_est()`, `entree_ouest()` et `sortie_est()` qui respecte les contraintes ci-dessus.

Exercice 3 : Verrous (exercice 4.7.3 page 67 du livre)

Utilisez les verrous pour résoudre le problème de l'allocation des imprimantes vu en cours.

Indications. Utiliser deux verrous : l'un pour mettre les processus en attente lorsqu'il n'y a pas d'imprimante disponible, et l'autre pour protéger les données manipulées par les procédures `Allouer` et `Liberer`. De plus, il s'agit d'appliquer le motif de code suivant :

```
Verrouiller(mutex);
while (not condition) do
    Deverrouiller(mutex);
    Verrouiller(attente_de_condition);
    Verrouiller(mutex);
end while;
Deverrouiller(mutex);
```

Exercice 4 : Problème des lecteurs-rédacteurs

Considérons la solution au problème des lecteurs et des rédacteurs de la figure 2.

Que se passe-t-il si dans la procédure `début_lecture` on déplace `P(lr)` après `V(mutex)` ?

Figure 2 Problème des lecteurs-rédacteurs (avec priorités égales) résolu à l'aide des sémaphores

```
1: module lecteurs_rédacteurs;  
2: defines début_lecture, fin_lecture, début_écriture, fin_écriture;  
  
3: var  
4:   nb_Lecteurs : integer;  
5:   lr: sémaphore  $\leftarrow$  1;  
6:   r: sémaphore  $\leftarrow$  1;  
7:   mutex: sémaphore  $\leftarrow$  1;  
  
8: procedure début_lecture  
9:   P(lr);  
10:  P(mutex);  
11:  nb_Lecteurs  $\leftarrow$  nb_Lecteurs + 1;  
12:  if nb_Lecteurs = 1 then  
13:    P(r);  
14:  end if  
15:  V(mutex);  
16:  V(lr);  
17: end  
  
18: procedure fin_lecture  
19:   P(mutex);  
20:   nb_Lecteurs  $\leftarrow$  nb_Lecteurs - 1;  
21:   if nb_Lecteurs = 0 then  
22:     V(r);  
23:   end if  
24:   V(mutex);  
25: end  
  
26: procedure début_écriture  
27:   P(lr);  
28:   P(r);  
29: end  
  
30: procedure fin_écriture  
31:   V(r);  
32:   V(lr);  
33: end  
  
34: procedure initialisation  
35:   nb_Lecteurs  $\leftarrow$  0  
36: end
```
