

Corrigé Série 2

Série rendue le: 18 avril 2005

Exercice 1 : Exclusion Mutuelle

On a deux processus. Les états du programme sont au format $\langle pos1, pos2, demande[1], demande[2], tour \rangle$. $pos1$ et $pos2$ sont les positions du processus 1 et du processus 2 dans l'algorithme:

1. avant la ligne 8 (donc au départ de la procédure *début*)
2. à la ligne 11 (après l'affectation des variables et avant l'évaluation de la condition de la boucle)
3. dans la section critique (avant l'appel à la procédure *fin* et après la fin de l'appel à *début*).

La figure 1 montre que les deux processus peuvent atteindre un état où ils sont simultanément dans la section critique. La solution proposée est donc fausse!

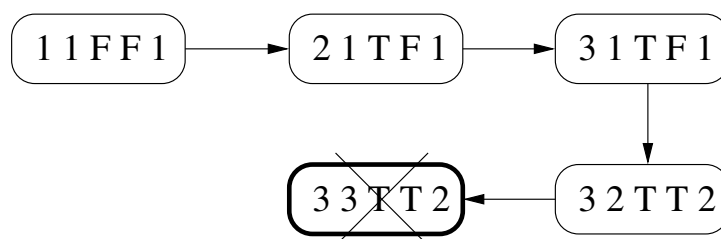


Figure 1: Violation de la section critique dans l'algorithme de Peterson modifié

Exercice 2 : Sémaphores et compteurs de Robert

Spécification à l'aide des compteurs de Robert

On définit trois compteurs : *actPont*, *attOuest* et *attEst* qui désignent le nombre de véhicules actuellement sur le pont (tout sens confondu), le nombre de véhicules qui attendent de traverser d'ouest en est et le nombre de véhicules qui attendent de traverser d'est en ouest respectivement. De plus, on définit une variable *sens* qui désigne le sens courant pour traverser le pont (alternance).

Traversée est \rightarrow ouest possible si :

$actPont == 0$ **and** ($sens == \text{est-ouest}$ **or** $attOuest == 0$)

Traversée ouest \rightarrow est possible si :

$actPont == 0$ **and** ($sens == \text{ouest-est}$ **or** $attEst == 0$)

Solution avec les sémaphores

```
semaphore pont=1
semaphore ouest=1
semaphore est=1

entree_ouest()
{
    P(ouest)
    P(pont)
}

sortie_est()
{
    V(pont)
    V(ouest)
}

entree_est()
{
    P(est)
    P(pont)
}

sortie_ouest()
{
    V(pont)
    V(est)
}
```

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

```

module Imprimantes;
const
  MAXIMPR = ...;
var
  i : integer;
  occupee : array 1..MAXIMPR of boolean;
  mutex : verrou; (* protection des donnees *)
  file : verrou; (* file d'attente *)
  alloc : verrou; (* un seul processus dans Allouer *)
  nbimpr : integer; (* imprimantes disponibles *)
  demande : integer; (* demandes en attente *)

procedure Allouer (var imp : integer);
code
  Verrouiller(alloc);
  Verrouiller(mutex);
  while (nbimpr < 1) do
    demande := demande + 1;
    Deverrouiller(mutex);
    Verrouiller(file);
    Verrouiller(mutex);
  end while;
  nbimpr := nbimpr + 1;
  imp := 1;
  while (occupee[imp]) do
    imp := imp + 1;
  end while;
  occupee[imp] := true;
  Deverrouiller(mutex);
  Deverrouiller(alloc);
end Allouer;

```

```

procedure Libérer (imp : integer);
code
  Verrouiller(mutex);
  nbimpr := nbimpr + 1;
  occupee[imp] := false;
  if (demande > 0) then
    demande := demande - 1;
    Deverrouiller(file);
  end if;
  Deverrouiller(mutex);
end Libérer;

```

```

code (* initialisation *)
  demande := 0;
  nbimpr := MAXIMPR;
  while i < MAXIMPR do
    occupee[i] := false;
    i := i + 1;
  end while
  Verrouiller(file);
end Imprimantes;

```

Exercice 4 : Problème des lecteurs-rédacteurs

Interblocage (Deadlock). Par exemple un lecteur exécute $P(r)$ et avant d'exécuter $P(lr)$ un rédacteur exécute $P(lr)$ suivi de $P(r)$. Ainsi le lecteur attend sur $P(lr)$ alors que le rédacteur attend sur $P(r)$.