Nom : Prénom : Option : IEA IOE

IUP MIAGe 3eme année — Année universitaire 94-95

Examen Programmation Parallèle — Alain Griffault

Lundi 22 mai, 10 heures - 12 heures
Tous documents autorisés

Exercice 1 : Partage de ressources avec des sémaphores

Une bibliothèque contient uniquement cinq (5) exemplaires d'un ouvrage. Ces exemplaires sont régulièrement d'une part lus par des lecteurs, d'autre part modifiés par un écrivain à partir de l'original en sa possession.

En terme d'algorithmes :

```
Algorithme de l'ecrivain
begin
while true do
  begin
  (* modifier l'original *)
  (* modifier les 5 exemplaires *)
  end
end
```

```
Algorithme d'un lecteur
begin
while true do
  begin
  (* consulter un des exemplaires *)
  end
end
```

Modifiez ces algorithmes par l'adjonction de sémaphores afin de garantir l'unique contrainte suivante : **Deux lecteurs simultanés ont toujours des exemplaires identiques.**

Pour chacun des sémaphores utilisés, décrivez brièvement son rôle.

Exercice 2 : Le problème de l'exclusion mutuelle

Ce problème est relativement simple. Il s'agit de garantir à un processus qu'il sera le seul à un instant donné, à exécuter une portion critique de son code. En d'autres termes, plusieurs processus exécutent le programme suivant en parallèle :

```
Algorithme d'un processus
begin
while true do
  begin
  < section non critique >
  (* essayer d'entrer en section critique *)
  < section critique >
  (* sortir de la section critique *)
  end
end
```

En 1966, H. Hyman a publié dans le cas de deux processus une solution qui utilise les trois variables globales `flag[0] : booleen, flag[1] : booleen, token : 0..1`, initialisées respectivement à `false, false, 0`.

```
Algorithme du processus I
begin
while true do
  begin
  < section non critique >
  flag[I] := true;
  while token <> I do
    begin
    while flag[1-I] = true do
      begin
      end
    token := I
    end
  < section critique >
  flag[I] := false;
  end
end
```

```
Programme OCCAM de la solution.
VAR flag[2], token:
PROC HYMAN(VALUE I) =
  WHILE TRUE
    SEQ
      < section non critique >
      flag[I] := TRUE
      WHILE token <> I
        SEQ
          WHILE flag[1-I] = TRUE
            SKIP
          token := I
      < section critique >
      flag[I] := FALSE:
SEQ
  token := 0
  flag[0] := false
  flag[1] := false
  PAR
    HYMAN(0)
    HYMAN(1):
```

Montrez, en donnant une exécution possible de ce programme, que cette solution est fausse du fait qu'il est possible que les deux processus soient simultanément en section critique.

Instant	Processus 0	Processus 1	flag[0]	flag[1]	token
0 (initial)			false	false	0
1					
2					
3					
4					
5					
6					
7					
8					
8					
10					
11					
12					

Exercice 3 : Modélisation de piles de messages.

En vous inspirant des exercices traités en cours et en travaux dirigés, écrivez en OCCAM ou en CSP une description du comportement des processus suivants :

1. une pile (ie LIFO) *parfaite* pouvant contenir **un message**.
2. une pile (ie LIFO) *avec perte possible* pouvant contenir **un message**.
3. une pile (ie LIFO) *avec duplication possible* pouvant contenir **un message**.
4. une pile (ie LIFO) *parfaite* pouvant contenir **N messages**.
5. une pile (ie LIFO) *avec perte possible* pouvant contenir **N messages**.
6. une pile (ie LIFO) *avec duplication possible* pouvant contenir **N messages**.