EST Salé
Département Informatique
LP BigData

Programmation Concurrente en Java
TP N°3 : Exclusion mutuelle

Travail à rendre par mail à l'adresse
abdelali.lasfar@est.um5.ac.ma
avant le 20 janvier 2023

Rappels

Définitions :
- **Ressource critique** : une ressource partageable à un seul point d'accès (qui ne peut être attribuée qu'à une seule tâche à un instant donné) est dite ressource critique.
- **Section critique** : la phase d'utilisation par une tâche d'une ressource critique est dite section critique.

-La programmation de l'exclusion mutuelle se décompose en trois parties :
- **entrée**
- **section critique**
- **sortie**

Objectifs : les algorithmes de l'exclusion mutuelle
- Algorithme de Dekker
- Les sémaphores

Algorithme de Dekker

On se propose de programmer l'exclusion mutuelle entre deux tâches parallèles pour l'accès à une section critique : les seules opérations indivisibles sont l'affectation d'une valeur à une variable et le test de la valeur d'une variable.

Principe de la solution :

- Définir un ensemble de variables d'état, communes aux contextes des deux tâches.
- L'autorisation d'entrée en Section Critique sera définie par des tests sur ces variables, et l'attente éventuelle sera programmée comme une attente active (répétition cyclique des tests).

On supposera par la suite que les tâches sont cycliques et que leur comportement est le suivant :

```
While (true) {
Entrée_en_section_critique
```

Section_critique
Sortie_de section_critique S
ection_non_critique }

Exercice 1
On utilise une seule variable booléenne M telle que M = vrai si une des tâches se trouve dans sa section critique, faux sinon.
Écrire le programme
Vérifier que l'exclusion mutuelle ne peut être ainsi programmée.

Exercice 2
On utilise une variable commune unique T telle que T = i si et seulement si la tâche Pi est autorisée à entrer en sa section critique (i = 0, 1).
Écrire le programme d'une tâche.
Montrer que la solution ne vérifie pas la condition (: le blocage d'une tâche hors de sa section critique peut empêcher l'autre d'entrer en sa section critique) mais vérifie les autres conditions.

Exercice 3
On utilise C(i) variable booléenne attachée à la tâche Pi (i = 0, 1)
C(i) = vrai si Pi est dans sa section critique ou demande à y entrer
C(i) = faux si Pi est hors de sa section critique
Pi peut lire et modifier C(i), peut lire seulement C(j) si j différent de i.
Écrire le programme de la tâche Pi Vérifier qu'on ne peut obtenir qu'une solution satisfaisant aux conditions a), c), d) ou b), c), d) mais non aux quatre.

Exercice 4
On peut obtenir une solution correcte en combinant les solutions précédentes et en introduisant une variable supplémentaire T servant à régler les conflits à l'entrée de la section critique, T n'est modifiée qu'en fin de section critique.
L'ensemble des variables est:
- C(i) avec la signification précédente (Question 3)
- T avec la signification précédente (Question 2)

S'il y a conflit (C(i) = C(j) = vrai), Pi et Pj exécutent une séquence d'attente où T a une valeur constante. Si T= j, alors Pi annule sa demande en positionnant C(i) à faux, Pj peut alors entrer en section critique. Pi attend que T= i et refait sa demande en positionnant C(i) à vrai.
Écrire le programme de Pi. Vérifier les 4 conditions.