IF223 - Algorithmique Distribuée - TD1 : Exclusion mutuelle avec des primitives

Rohan Fossé

rfosse@labri.fr

Dans tous les exercices qui suivent nous faisons l'hypothèse qu'aucun des processus n'est fautif.
Rappel: `await(condition)` impose au processus d'attendre tant que `condition` n'est pas vérifiée.

Exercice 1

Ecrire un algorithme qui résout le problème de l'exclusion mutuelle pour n processus. L'algorithme doit assurer la propriété d'absence d'interblocages (*deadlock-freedom*).
Vous avez à disposition un objet partagé linéarisable dit *test&set*. La valeur initial d'un objet x de type test&set est 0. L'état de l'objet peut être 0 ou 1.

Il peut être manipulé à l'aide des méthodes suivantes :

- $x.test\&set()$ écrit la valeur 1 en x et retourne la valeur précédente.
- $x.reset()$ écrit la valeur 0 en x.

Dire si l'algorithme proposé garantit la propriété de *starvation−freedom*. Justifier la réponse.

Exercice 2

Ecrire un algorithme qui résout le problème de l'exclusion mutuelle pour n processus. Chaque processus a un identifiant unique dans l'ensemble des valeurs {1, 2,. . ., n}. L'algorithme doit assurer la propriété *starvation−freedom*. Vous avez à disposition une file partagée linéarisable, nommé q. Au début la file est vide. Les méthodes pour la manipuler sont :

- $q.Enfiler(v)$ insère la valeur v dans la file
- $q.Defiler()$ qui retourne la valeur en tête et la supprime de la file
- $q.Tete()$ qui retourne la valeur en tête de la file. Cette méthode ne change pas l'état de la file.

Exercice 3

```
Algorithme 1 :  Programme pour le processus i

1: Flag[i] ← true;
2: await(Turn = i or (Flag[Turn] = false))
3: await(x.Test&Set = 0)
4: section critique;
5: Flag[i] ← false;
6: if (Flag[Turn] = false) then
7:    Turn ← (Turn + 1) mod n
8: x.reset;
```

L'algorithme 1 résout le problème de l'exclusion mutuelle pour n processus. Chaque processus a un identifiant unique dans l'ensemble des valeurs {1, 2, . . . , n}.

- x est un objet partagé linéarisable de type *test&set* (décrit à la question 1).
- *Flag* est un tableau de n cases tel que pour tout i = 1, . . . n, $Flag[i]$ est un registre linéarisable dont la valeur peut être *true* ou $false$. La valeur initiale de chaque registre est $false$.
- $Turn$ est un registre linéarisable dont la valeur $\in$ {1, 2, . . . , n}. La valeur initiale est 1.

1. On suppose que les processus 1 et 2 sont les seuls à vouloir entrer en section critique. Est-il possible que le processus 2 entre en section critique avant le processus 1 ? Justifier.
2. Quelle propriété assurent les lignes 1, 2, 5 et 6 de l'algorithme ?