

Programmation Concurrente

Problèmes classiques

Exercice 1

Initialisation :

```
servings = 0;
mutex = mutex_init()
empty = semaphore(0)
full = semaphore(0)
```

Le chef :

```
chef() {
    while (true) {
        wait(empty)    // attend que la casserole soit vide
        put_servings_in_pot(N)
        servings = N
        post(full)     // signale les cannibales que la casserole est pleine
    }
}
```

Chaque cannibal :

```
cannibal() {
    while (true) {
        lock(mutex)
        if (servings == 0) {
            post(empty) // signale au chef que la casserole est vide
            wait(full)  // attend le remplissage de la casserole
        }
        get_serving_from_pot()
        servings--
        unlock(mutex)
        eat()
    }
}
```

Exercice 2

Soit un pont situé sur une route à sens unique que des véhicules traversent toujours dans le même sens. Deux types de véhicules traversent le pont : des voitures et des camions. La masse d'un camion est exactement le double de celle d'une voiture. Le pont peut supporter une charge maximale correspondant à 8 voitures (ou 4 camions, ou 6 voitures et 1 camion, etc.). Les voitures et camions font un tour en campagne pendant un certain temps (long) puis traversent le pont (rapide). Le nombre total de voitures et camions est arbitraire.

Voici le pseudo-code d'une solution modélisant ce système à base de sémaphore :

```
bridge = semaphore(8)

car() {
    while (true) {
        drive_around(long_time)
        wait(bridge)
        cross_bridge(short_time)
        post(bridge)
    }
}

truck() {
    while (true) {
        drive_around(long_time)
        wait(bridge)
        wait(bridge)
        cross_bridge(short_time)
        post(bridge)
        post(bridge)
    }
}
```

Question 1

Montrez que la solution proposée ci-dessus souffre de deadlock (interblocage).

- Le pont est vide
- 8 camions arrivent et chacun est préempté exactement après avoir fait le premier `wait(bridge)` mais avant de faire le deuxième `wait(bridge)`
- N'importe quel véhicule arrivant ensuite bloquera complètement le système

Question 2

Proposez une solution garantissant:

- Aucun deadlock.
- Respect de la charge maximale supportée par le pont.

```
bridge = semaphore(8)
enter = mutex_init()

car_thread {
    loop {
        drive_around(long_time)
        lock(enter)
        wait(bridge)
        unlock(enter)
        cross_bridge(short_time)
        post(bridge)
    }
}

truck_thread() {
    loop {
        drive_around(long_time)
        lock(enter)
        wait(bridge)
        wait(bridge)
        unlock(enter)
        cross_bridge(short_time)
        post(bridge)
        post(bridge)
    }
}
```