université
PARIS-SACLAY

Programmation en Python
Master 2 Réseaux Télécoms
TP 5

1 Situation de compétition

On considère le code suivant (voir le fichier `race_condition.py`) :

```
counter = 0


def racing_function():
    global counter
    while True:
        counter += 1
        counter -= 1
```

La fonction `racing_function` incrémente puis décrémente un compteur global (la variable `counter`).

1. Lancez la fonction `main()` du fichier `race_condition.py` avec la valeur de la constante `NUM_THREADS` égale à 1. Pourquoi de temps en temps le programme affiche la valeur du compteur autre que 0 ? Est-ce qu'il peut afficher d'autres valeurs que -1, 0 ou 1 ?
2. Augmentez nombre de threads d'exécution (`NUM_THREADS`). Pourquoi le résultat est changé ?
3. Modifiez le code en ajoutant un verrou primitif global (un objet de la classe `threading.Lock`) et en protégeant par ce verrou la partie sensible du code. Que doit-on faire pour que les valeurs affichées soient toujours égales à 0 ?

2 Interblocage

On considère le code suivant (voir le fichier `deadlock.py`) :

```
from threading import Thread, Lock
import time

class RoadLane(Thread):
    def __init__(self, name, delta_t=0.1):
        Thread.__init__(self)
        self.name = name
        self.delta_t = delta_t
        self.counter = 0
        self.right_priority = None
        self.engaging = Lock()

    def run(self):
        while True:
            if self.right_priority:
                with self.right_priority:
                    with self.engaging:
                        print(f"{self.name}:{self.counter}")
                        self.counter += 1
            time.sleep(self.delta_t)
```

La classe `RoadLane` représente une file de circulation sur une route. Un objet de cette classe détient les références sur les deux verrous primitif, un dénommé `engaging` et l'autre dénommé `right_priority` (le dernier n'est pas initialisé lors de la création de l'objet). La méthode `run` représente une circulation plus ou moins dense, suivant la valeur de l'intervalle entre deux voitures `delta_t`. On peut construire une intersection avec la priorité à droite avec plusieurs objets du type `RoadLane`. Par exemple, le code suivant :

```
e=RoadLane('eastbound')
n=RoadLane('northbound')
e.right_priority = n.engaging
```

crée une intersection de deux routes à sens unique, et établit la règle de priorité (les voitures allant vers l'est doivent céder le passage à celles allant vers le nord).

Complétez le code de la fonction `model_crossroad()`. Cette fonction doit créer une intersection de deux routes chacune avec deux files de circulation (une file dans chaque sens), comme sur la Figure 1. Ensuite, on lance tous les threads et observe le comportement du programme. Essayez le programme avec differentes valeur de `delta_t`. A partir de quelle densité de circulation observe-t-on des interblocages ?

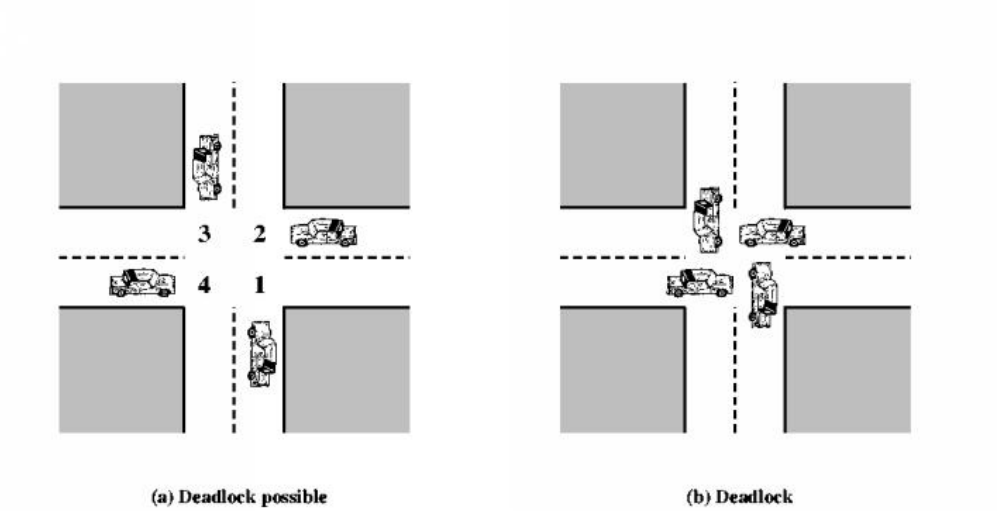

FIGURE 1 – Un interblocage dans la vie réelle.

3 Pédagociciel de multiplication

On se propose de créer un logiciel d'apprentissage de multiplication pour l'école primaire (voir le fichier `tabmul.py`). Le programme s'exécute dans deux threads, un thread principal qui s'occupe de l'entrée de l'utilisateur et un thread secondaire qui propose les nouveaux exercices, compte le temps imparti pour chacun (5 secondes est une valeur raisonnable) et vérifie le résultat. Si l'utilisateur entre le mot `quit` au lieu d'un nombre, le programme s'arrête (les deux threads doivent se terminer !). Voici un exemple d'une session :

```
6 x 5 =
30
OK
3 x 3 =
9
OK
6 x 2 =
11
Nope!
5 x 8 =
40
OK
```

```
7 x 4 =
Too late!
8 x 9 =
kadlfkasdjf
Not an integer
5 x 8 =
40
OK
3 x 6 =
quit
4 out of 7 correct answers. Bye!
```