TP 1
Concurrence et synchronisation en Python – INFO3 S5
Sémaphores

Objectifs du TP : Reprendre les éléments théoriques vus en cours sur les sémaphores, puis en TD, et les appliquer à la conception de solutions en langage Python aux problèmes posés par la concurrence de processus s'exécutant en parallèle

1 L'exclusion mutuelle

Lisez le programme source Python contenu dans de fichier « exclusionMutuelle-semaphores.py », qui reprend en Python l'exemple de compétition sur une ressource critique vue en Cours et dans l'exercice 1 du TD 1.

Vous devez y retrouver :

- L'utilisation du module threading (import threading),
- L'utilisation du module tprint (import tprint),
- Une variable globale commune RC (ressource critique) non protégée,
- Deux threads prenant en charge les fonctions : thread_P1() et thread_P2(),
- Le programme principal qui créé les 2 threads (Thread(()), puis les lance (start()), et enfin attend leur terminaison (join())

1.1 Lancez tel quel l'exécution du programme « exclusionMutuelle-semaphores.py ». Que constatez-vous ? Expliquez pourquoi il est nécessaire de protéger la variable RC. Repérez dans le code les *sections critiques* qui doivent être protégées par un mécanisme d'*exclusion mutuelle.*

1.2 Lancez à nouveau le code après avoir décommenté la ligne « from tprint import tprint ». Expliquez l'intérêt de la fonction tprint.

1.3 En repartant du *réseau de Pétri* dessiné en TD (TD 1 exercice 1) pour résoudre ce problème, éditez-le dans le simulateur de réseau de Pétri (jpns) disponible sur la page madoc du cours, puis exécutez-le en mode *pas à pas* (attention à utiliser l'option «settings-parallel-manual » ou «settings-parallel-random », essayez les 2 possibilités). Observez le comportement du réseau de Pétri avec différentes conditions initiales du sémaphore (0,puis 1,2). Qu'en déduisez-vous ?

1.4 Complétez le code Python afin de résoudre le problème de concurrence à l'aide d'un sémaphore. Puis vous testerez votre solution avec trois valeurs initiales de sémaphore distinctes : 0, 1, 2

NB : Un sémaphore S est créé et initialisé à 3 par « `Semaphore(3)` », et manipulable par les opérations «`S.acquire()`» pour `P(S))` et « `S.release()` » pour `V(S).`

2 L'étreinte fatale

Reprenez les solutions *réseau de Pétri* et *pseudo code* vues en TD (TD 1, exercice 3).

2.1 Lisez le code Python contenu dans le programme « etreinte-fatale-semaphores.py ». Exécutez le programme Python. Que constatez-vous ? Et que constatez-vous en simulant le *réseau de Pétri* correspondant sous jpns ?

2.2 Editez et simulez sous jpns le *réseau de Pétri* permettant de corriger ce problème. Vérifiez que votre solution est correcte.

2.3 Implémentez votre solution à l'aide sémaphores dans le programme python. Puis testez-la.

3 L'alternance

Reprenez les solutions *réseau de Pétri* et *pseudo code* vues en TD (TD 1, exercice 2).

3.1 Lisez le code Python contenu dans le programme « ping pong-semaphores.py ». Exécutez le programme Python. Que constatez-vous ? Et que constatez-vous en simulant le *réseau de Pétri* correspondant sous jpns ?

3.2 Editez et simulez sous jpns le *réseau de Pétri* permettant d'implémenter l'exclusion mutuelle et l'alternance stricte des 2 processus (ping-pong-ping…). Vérifiez que votre solution est correcte.

3.3 Implémentez votre solution à l'aide sémaphores dans le programme python. Puis testez-la.

4 Le problème des Producteurs Consommateurs

Reprenez les solutions *réseau de Pétri* et *pseudo code* vues en TD (TD 1, exercice 4).

4.1 Lisez le code Python contenu dans le fichier « producteur-consommateur-semaphores.py » et le module « tampon_lifo.py » . Exécutez ce programme Python. Que constatez-vous ? Et que constatez-vous en simulant le *réseau de Pétri* correspondant sous jpns ?

4.2 Editez et simulez sous jpns le *réseau de Pétri* permettant d'implémenter la contrainte C1 : l'exclusion mutuelle sur les opérations deposer() et retirer du tampon. Vérifiez que votre solution est correcte (interblocages, famine, …).

4.3 Implémentez votre solution à l'aide sémaphores dans le programme python. Puis testez-la.

4.4 Répéter les questions 4.2 et 4.3 pour résoudre les 2 contraintes C1 et C2, puis les 3 :C1 et C2 et C3

5 La chaine d'assemblage d'un aéroplane

Reprenez les solutions *réseau de Pétri* et *pseudo code* vues en TD (TD 1, exercice 5).

5.1 Lisez le code Python contenu dans le fichier « aeroplane-semaphores.py ». Exécutez ce programme Python. Que constatez-vous ? Et que constatez-vous en simulant le *réseau de Pétri* correspondant sous jpns ?

5.2 Editez et simulez le *réseau de Pétri* résolvant ce problème à l'aide de sémaphores. Vérifier qu'aucune configuration interdite n'est atteinte, et qu'aucun interblocage ne se produit.

5.3 Implémentez votre solution à base de sémaphores dans le programme Python, puis testez-la.