Python et Unix

CSII 2 – Examen Semestre 2, 2003

Durée : 2 heures
Aucun documents autorisés

Exercice 1 : Python

Quel est l'effet de l'instruction `a=3` si a est déjà déclarée, si a n'existe pas encore ? (1 point)

Quelle est la différence entre un tuple et une liste ? Comment les déclarer ?(1 point)

Soit L une liste, comment accéder : (1,5 points)

- Au iéme élement ?
- Aux 3 premiers élements ?
- Au dernier élement ?

Qu'est-ce qu'un dictionnaire python ? Comment le déclarer ? (1 point)

Quelles sont les principales opérations qu'il permet ? (1 point)

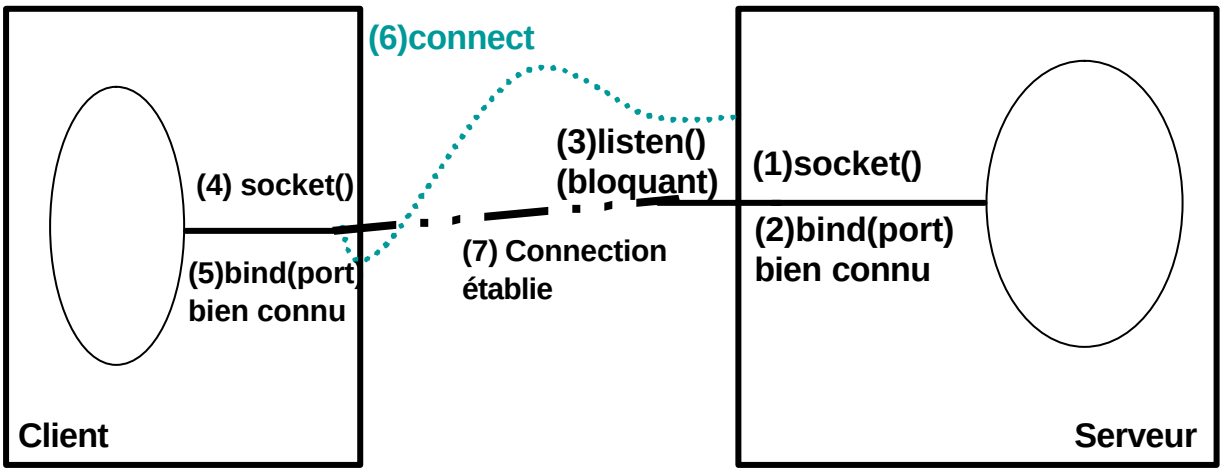
Comment créer un module python ? Comment s'en servir dans un autre script ? (1 point)

Quelles sont les structures de contrôle proposées en python ? Ecrivez un petit exemple (4 lignes) illustrant l'utilisation de chacune d'elle). (1,5 point)

Exercice 2 : Sockets

Question 1 :

Le schéma suivant exprime le schéma d'établissement d'une connection TCP entre un client et un serveur. Corrigez et expliquez les erreurs qu'il contient. (3 points)



Exercice 3 : Semaphore

Nous allons à présent supposer que le tunnel est suffisamment grand pour contenir trois voitures simultanément. Pour cela nous allons utiliser un threading.Semaphore. A la différence d'un verrou, un sémaphore dispose d'un compteur qui est décrémenté de 1 à chaque acquisition et réincrémenté à chaque release. L'appel à acquire est bloquant si le compteur est à la valeur 0. On initialise la valeur du compteur du sémaphore à la création :

```
monSem = threading.Semaphore(5)
while (1) :
        if (monSem.acquire(0)) :
                print « Ok »
        else :
                print « fini »
                break
monSem.release() …
```

Implémentez cette limite à trois voitures dans le tunnel simultanément à l'aide d'un sémaphore.

Exercice 4 : wait et notify

Il existe également des mécanismes de synchronisations similaire au java avec wait, notify et notifyAll
On instancie un objet Condition :
monMonitor = threading.Condition() :

On peut acquire et release cet objet, mais l'on dispose aussi de primitives de plus haut niveau. Il faut acquérir l'objet Condition (ou *monitor* en java) avant tout appel aux primitives qui suivent :

monMonitor.wait()
Relache le moniteur, attends (sleep) qu'un autre thread notifie le moniteur que « la voie est libre », et réacquiert le moniteur avant de poursuivre l'éxecution.

nomMonitor.notify()
Reveille un processus (aléatoirement) en attente sur ce moniteur, qui poursuivra son éxecution.

nomMonitor.notifyAll()
Reveille tous les processus en attente sur ce moniteur. Permet de mettre en concurrence les threads pour une ressource, ou d'éviter l'attente active en cas de besoin de synchronisation entre threads.

Toutes ces méthodes ne peuvent être utilisées que si l'on a préalablement acquis (monMonitor.acquire() et monMonitor.release() l'exclusivité sur le moniteur)

Nous allons ajouter un camion au système: le camion est prioritaire sur les voitures, et il est interdit d'avoir des voitures en même temps que le camion dans le tunnel.

Il correspond à un écrivain : dans le modèle lecteurs/écrivains d'accès à une ressource, les lecteurs peuvent accéder simultanément à la ressource, mais l'écriture est en exclusion mutuelle de toute autre opération de lecture ou écriture.

Les voitures (lecteurs) avant d'entrer dans le tunnel (section critique) consultent une variable `camionEnAttente` booléenne ; si elle est vraie ils se mettent en atente sur un moniteur (Condition) `Camion`, avant de demander le sémaphore de l'exercice 3 qui limite toujours à trois le nombre de voitures dans le tunnel.
Le camion (écrivain) positionne la variable `camionEnAttente` booléenne à vrai quand il arrive au tunnel, et acquiert trois fois le sémaphore pour rentrer dans le tunnel.

Ajoutez un temps aléatoire avant l'appel à `EntrerTunnel`, de l'ordre de 0-3 pour les voitures et 0-1,5 pour le camion.

Adaptez l'affichage pour faire apparaître plus nettement le camion.

Question subsidiaire : (répondre en commentaire dans le source)

Que se passe-t-il s'il on introduit deux écrivains (camions) dans le système ?
Quels problèmes/cas de figure peut-on rencontrer ?
Proposez une solution qui permette l'accès à un seul écrivain ou à plusieurs lecteurs, dans un système ou les demandes de lecture et d'écriture arrivent de façon aléatoire.
Implémentez votre solution, en faisant boucler indéfiniment 12 threads dont deux écrivains, avec un sleep aléatoire avant de demander une lecture/ecriture.