

<p style="text-align: center;"><b>info201 : Système d'exploitation</b> <b>TD 4 : processus</b></p>
--

Pierre Hyvernats  
Laboratoire de mathématiques de l'université de Savoie  
bâtiment Chablais, bureau 17, poste : 94 22  
email : [Pierre.Hyvernats@univ-smb.fr](mailto:Pierre.Hyvernats@univ-smb.fr)  
www : <http://www.lama.univ-smb.fr/~hyvernats/>

### Partie 1 : questions de cours

*Question 1.* Qu'est-ce qu'un *processus* ?

*Question 2.* Donnez certaines des informations que le système d'exploitation doit gérer pour manipuler les processus.

### Partie 2 : États

Les 3 états principaux pour un processus sont :

- en exécution
- prêt à être exécuté
- bloqué

*Question 1.* Donnez, pour chacune des transitions suivantes, une situation où un processus peut changer d'état :

- en exécution → bloqué
- en exécution → prêt à être exécuté
- prêt à être exécuté → en exécution
- bloqué → prêt

*Question 2.* Un processus peut-il passer de l'état bloqué à l'état en exécution directement ?

### Partie 3 : ordonnancement

*Question 1.* Les deux processus  $P_1$  et  $P_2$  ont chacun besoin de 1 minute de temps de processeur pour faire ses calculs. Chacun est *bloqué* pendant 50% de son temps total d'exécution. (Cela signifie qu'à chaque instant, chaque processus a une probabilité 1/2 d'être bloqué.)

- Quelle est la durée d'exécution du processus  $P_1$  seul ?
- Quelle est la durée d'exécution des deux processus  $P_1$  et  $P_2$  s'ils sont exécutés séquentiellement ? (Autrement dit, le processus  $P_2$  commence à s'exécuter lorsque le processus  $P_1$  est terminé.)
- Quelle est la durée d'exécution des deux processus  $P_1$  et  $P_2$  s'ils sont exécutés avec un ordonnanceur "multiprogrammation" qui permet de changer de processus au bout d'un court laps de temps ?

*Question 2.* L'ordonnanceur préemptif le plus simple est le "tourniquet" :

- les processus sont rangés dans une *file* ("first in, first out") à leur création,
- le premier processus est exécuté pendant un court interval de temps (appelé *quantum*, typiquement de l'ordre de quelques dizaines de millisecondes),
- à la fin du quantum, le processus en exécution est arrêté et remis *en queue de file*.

On considère les processus fictifs suivant :

processus	création (ms)	durée (ms)
$P_1$	0	200
$P_2$	99	300
$P_3$	199	200
$P_4$	399	100

Donnez l'ordonnancement de ces processus quand le quantum de temps est de 100ms.

Question 3. L'ordonnancement "tourniquet" peut-il provoquer des famines ?

Question 4. En pratique, les processus ont une *priorité*. Les processus *plus prioritaires* devraient avoir plus facilement accès au processeur.

Une manière de gérer les priorités serait d'avoir une file par priorité. L'ordonnanceur choisit alors le premier processus dans la file de plus haute priorité.

Quel problème cela pose t'il ?

Question 5. Cherchez une manière d'éviter "le plus possible" les problèmes du tourniquet avec priorité.

Question 6. Par défaut, des processus différents utilisent des espace mémoire indépendants. Les *processus léger* peuvent partager de la mémoire, et donc des variables.

Le code python suivant définit 2 processus légers qui vont chacun incrémenter la valeur de N 1 000 000 fois.

```
from threading import Thread

N = 0          # variable globale

# fonction qui incrémente plusieurs fois la variable globale
def incr():
    global N    # nécessaire pour modifier une variable globale
    for _ in range(1000000):
        N = N + 1

# on définit 2 processus légers. Ils partagent la mémoire
t1 = Thread(target=incr)
t2 = Thread(target=incr)

# on les lance, ils s'exécutent "en parallèle"
t1.start()
t2.start()

# on attend qu'ils se terminent
t1.join()
t2.join()

print(f"La valeur finale de N est {N}.")
```

Quelle est la valeur finale de N attendue ?

Question 7. L'exécution sur mon ordinateur donne

```
$ python3 test.py
```

```
La valeur finale de N est 1492525.
```

Expliquez ce qui s'est passé.