PERRICHET Théotime

GORVIEN Mathis

Groupe D

Projet CS-PC 2021-22

I/ Introduction :

Au sein de ce projet, vous pourrez voir les réalisations que les auteurs ont pu réaliser durant les séances de TP allouées ainsi que pendant leur temps libre. 8 sujets ont été traité pour la réalisation de ce projet que sont :

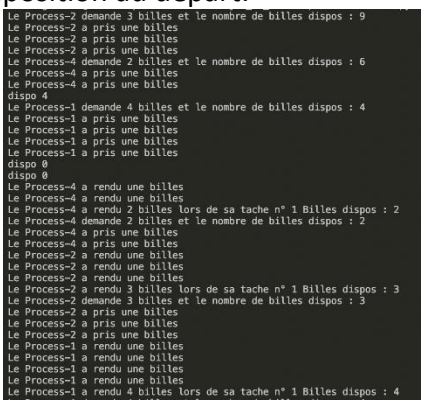
- Course hippique
- Gestionnaire de Billes
- Estimation de PI
- Exercice : faites des calculs
- Tri rapide
- Game of Life
- Simulation d'un restaurant
- Contrôleur de température et pression

L'objectif de ces fichiers étaient de transposer ou créer les programmes décrits dans le cahier des charges en utilisant le parallélisme et en utilisant la bibliothèque multiprocessing de python.

II/ Sujets :

- Gestionnaire de Billes

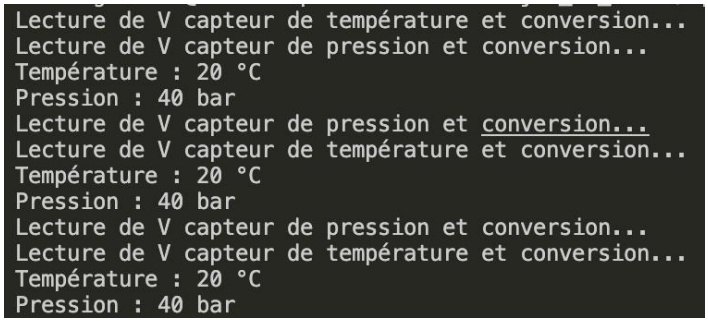
Dans cet exercice, le but était de mettre des ressources limitées à disposition de process qui utilisent ces ressources. Si un process n'a pas le nombre de ressources suffisantes pour travailler alors il attend qu'elles soient de nouveau disponibles à la suite de la fin d'un process qui rend ses ressources une fois sa tâche effectuée. Pour cela, 4 processus sont lancés et effectuent 4 fois une tâche en utilisant chacun un nombre de ressources différentes. De plus, un contrôleur vérifie que le nombre de billes disponible ne dépasse pas le nombre de billes mises à disposition au départ.



Nous pouvons donc observer les actions des différents process en détail.

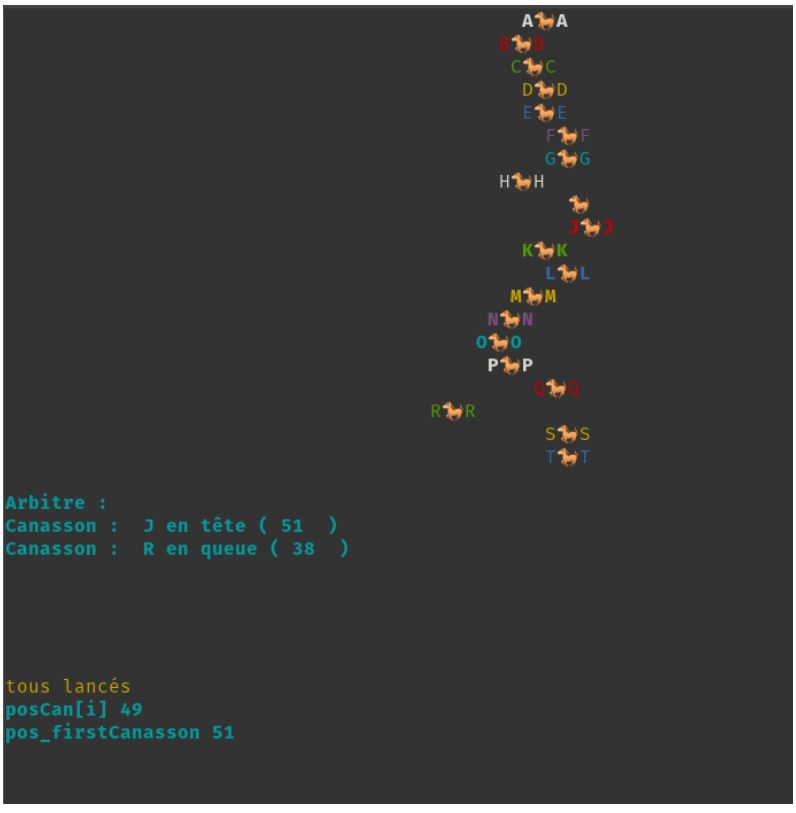
- Contrôleur de température et de pression

Le but de cet exercice était de mettre en place un système multi-tâche. Différents processus sont mis en route, un pour la température, un pour la pression. Ces processus lisent et convertissent les valeurs des capteurs pour que l'écran (autre process) puisse faire l'affichage de la pression et de la température. Un dernier processus sert de contrôleur et gère l'action du chauffage et de la pompe en fonction des retours fourni par la lecture capteurs en parallèle. L'affichage donné par cette fonction n'est pas très intéressant car aucun capteur ne vient modifier les valeurs de pressions et température.



- Course hippique

Dans cet exercice, on cherche à réaliser un arbitre qui affiche en permanence le cheval qui est en tête ainsi que celui qui est dernier une course hippique, et qui à la fin de la course, affiche le gagnant. L'affichage est réalisé sans interface graphique, mais en effaçant et réécrivant du texte. Dans notre cas, nous avons choisi de mettre des emojis pour avoir plus de réalisme avec une lettre dédié à chacun pour bien les différentier.



- Game of Life

Il s’agit d’une grille dont les cases représentent soit un "être" vivant soit rien. L’état d’une case peut être modifié en fonction de son voisinage selon les règles décrites ci-dessous :

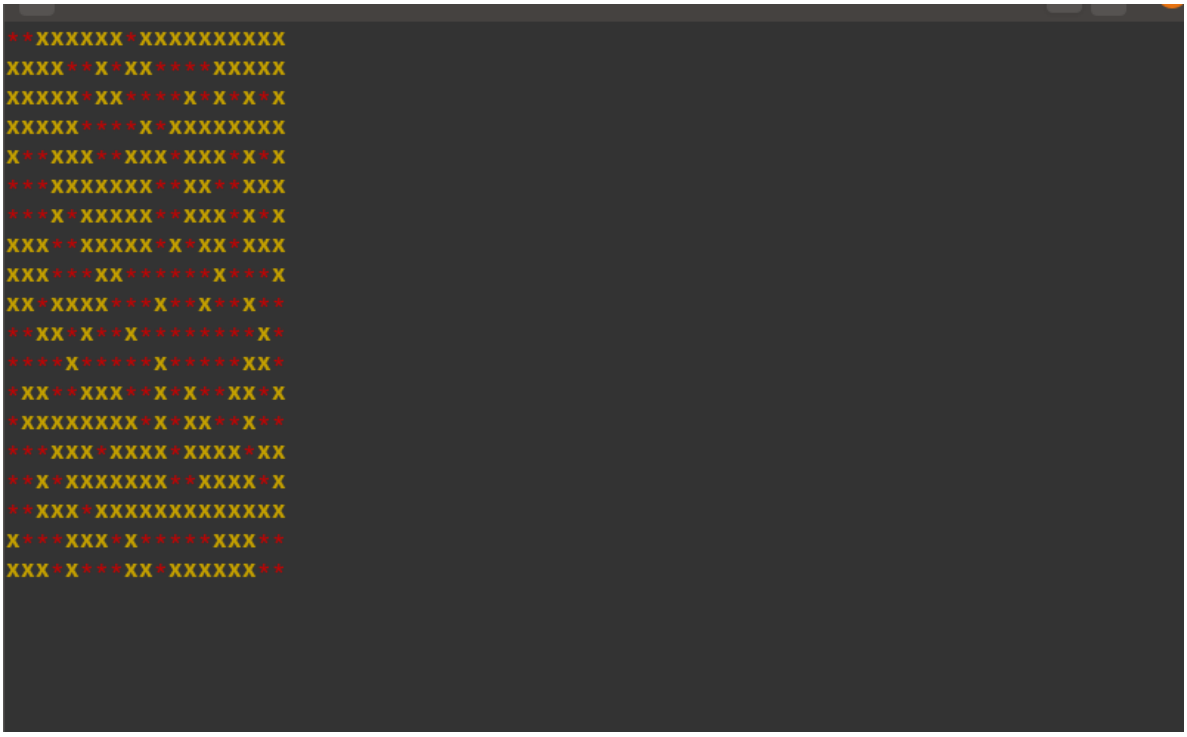
◦ Toute cellule vivante ayant moins de deux voisins vivants meurt, comme si cela était dû à une sous-population.

◦ Toute cellule vivante avec deux ou trois voisins vivants vit à la génération suivante.

◦ Toute cellule vivante avec plus de trois voisins vivants meurt, comme si cela était dû à une surpopulation.

◦ Toute cellule morte ayant exactement trois voisins vivants devient une cellule vivante, comme par reproduction.

Le but de cet exercice est de faire une version concurrente de ce jeu (avec les mêmes mécanismes que la course hippique).



On retrouve en rouge les cellules vivantes et en jaune les cellules mortes.

- Simulation d’un restaurant

Dans ce sujet, on souhaite simuler en temps réel le suivant :

1. simule des commandes de clients dans un restaurant

2. un certain nombre de serveurs en salle enregistrent ces commandes et les transmettent à la cuisine pour préparation

3. après leur préparation, les serveurs délivrent ces commandes aux clients

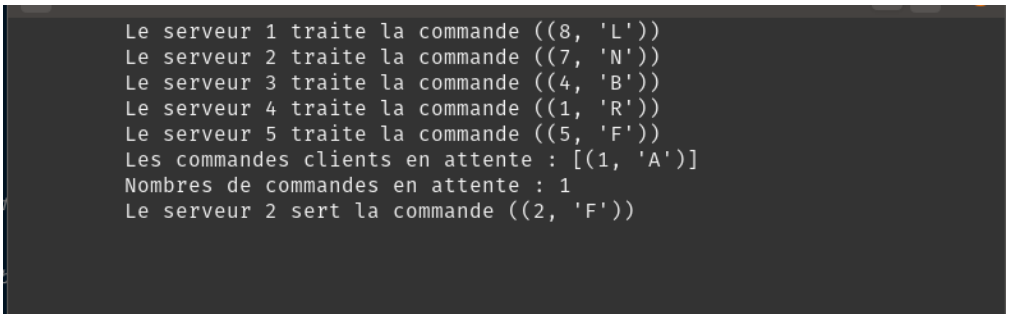
Pour cela, utilise 3 processus, des processus *Clients*, des processus *Serveurs* et un processus *Major d’homme.*

L’affichage des informations sera exclusivement géré par le major d’homme. Il affichera :

◦ les commandes des clients (les paires (id, menu)) dès leur émission

◦ le serveur qui prend cette commande en charge et simule sa préparation (par un délai)

◦ le client qui reçoit sa commande préparée



- Tri rapide

Dans cet exercice, on doit trier une liste selon la méthode suivante :

Pour trier un tableaux T de N éléments,

- Désigner une valeur du tableau (dit le Pivot p)

- Scinder T en deux sous-tableaux T1 et T2 tels que les valeurs de T1 soient ≤ p et celles de T2 soient > p

- Trier T1 et T2

- Reconstituer T en y plaçant T1 puis p puis T2

Le pivot est le premier élément du tableau.

Chaque Processus sous-traite à un processus fils la moitié du tableau qui lui est assigné et s'occupe lui-même de l'autre moitié.

On génère un tableau aléatoirement et on le trie ensuite.

```
PS C:\Users\titan\Desktop\Projet_CS_PC> & C:/Users/titan/anaconda3/python.exe c:/Users/titan/Desktop/Projet_CS_PC/Quick_Sort.py
tableau de base [39, 4, 63, 31, 89, 99, 43, 62, 38, 55, 48, 52, 95, 99, 49, 15, 89, 84, 78, 32, 30, 54, 85, 12, 87, 7, 99, 82, 31, 79, 99, 60, 67, 13, 39, 83, 57,
 5, 22, 22, 49, 12, 85, 88, 76, 87, 2, 49, 33, 59, 11, 32, 34, 81, 36, 9, 38, 79, 8, 47, 36, 86, 15, 29, 26, 18, 52, 43, 2, 90, 50, 52, 23, 78, 92, 15, 53, 67, 23
, 81, 41, 49, 78, 20, 23, 91, 5, 17, 73, 63, 34, 63, 38, 21, 17, 88, 39, 67, 6, 68]
tableau trié croissant :  [2, 2, 4, 5, 5, 5, 6, 7, 8, 9, 11, 12, 12, 13, 15, 15, 15, 17, 17, 18, 20, 21, 22, 22, 23, 23, 23, 26, 29, 30, 31, 31, 32, 32, 33, 34, 34,
36, 36, 38, 38, 38, 39, 39, 39, 41, 43, 43, 47, 48, 49, 49, 49, 49, 50, 52, 52, 52, 53, 54, 55, 57, 59, 60, 62, 63, 63, 63, 67, 67, 67, 68, 73, 76, 78, 78, 78, 79
, 79, 81, 81, 82, 83, 84, 85, 85, 86, 87, 87, 88, 88, 89, 89, 90, 91, 92, 95, 99, 99, 99, 99]
```

- <u>Exercice : faites des calculs</u>

Dans ce sujet, on réalise un échange entre un client demandeur, un serveur calculateur. On dispose de m demandeurs et de n calculateurs, donc lorsque le résultat est calculé et déposé dans la Queue par un processus calculateur, il y ajoute l'identifiant du demandeur. Ainsi, le demandeur peut filtrer la Queue des résultats pour trouver les réponses à ses demandes.

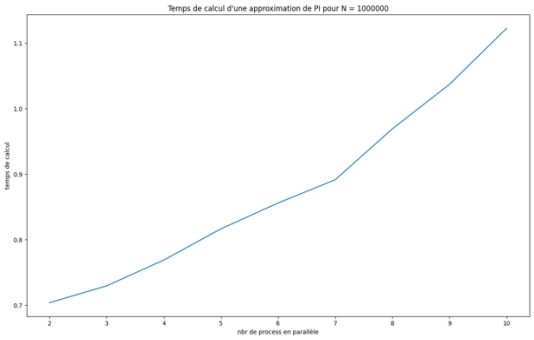
```
Le père 1744 demande :  28 * 48
Le Pere 1744 reçoit  1344
##################################################
Le père 9132 demande :  11 + 4
Le Pere 9132 reçoit  15
##################################################
Le père 12984 demande :  32 / 41
Le Pere 12984 reçoit  0.7804878048780488
##################################################
Le père 16636 demande :  43 * 42
Le Pere 16636 reçoit  1806
##################################################
Le père 1744 demande :  23 - 40
Le Pere 1744 reçoit  -17
##################################################
Le père 9132 demande :  44 * 36
Le père 12984 demande :  44 - 26
Le Pere 9132 reçoit  1584
##################################################
Le Pere 12984 reçoit  18
##################################################
Le père 16636 demande :  21 - 20
Le Pere 16636 reçoit  1
##################################################
```

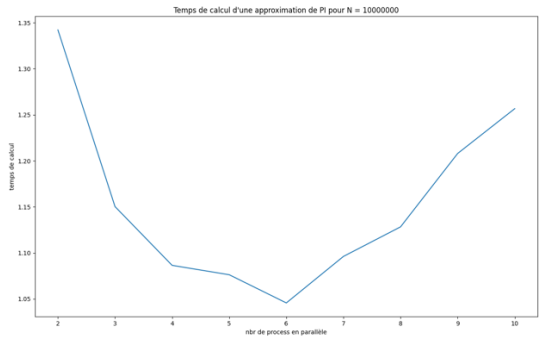
- <u>Calcul de pi avec process en parallèle</u>

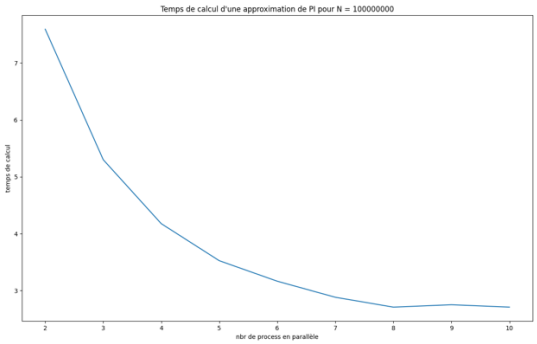
Dans ce script, nous avons adapté la fonction de calcul de pi par la méthode de l'arctan afin que plusieurs process calculent un certain nombre de partitions répartis sur chaque process. La somme des partitions calculées en parallèle donne pi. L'intérêt ici est d'étudier l'efficacité en termes de rapidité d'exécution d'un tel calcul en fonction du nombre de processus en parallèle et du nombre N de partitions de pi.

```
estimation de pi en 2 process : PI = 3.1415927410125732
Durée du calcul multi Process : 0.7707 s


estimation de pi en 3 process : PI = 3.1415908336639404
Durée du calcul multi Process : 0.81 s


estimation de pi en 4 process : PI = 3.141592502593994
Durée du calcul multi Process : 0.8041 s


estimation de pi en 5 process : PI = 3.141592502593994
Durée du calcul multi Process : 0.8268 s


estimation de pi en 6 process : PI = 3.141584634780884
Durée du calcul multi Process : 0.8683 s
```

Nous avons donc réalisé des essais pour N=10^6, 10^7, 10^8:







Les graphiques montrent bien l'importance du choix du nombre de processus à faire tourner en parallèle pour effectuer certaines tâches. Plus le nombre d'itération est grand, plus il devient intéressant de faire tourner des process en parallèle. Cependant, dans le cas où N=10^7, nous remarquons que le calcul de pi est plus rapide pour n=6 process en parallèle. Cela montre qu'il n'est pas toujours intéressant de choisir un nombre important de processus à faire tourner en parallèle, il faut faire des compromis.