Rapport de projet:

Programmation concurrentielle

Sujet : Simulation du producteur- consommateur

Introduction :
Ce rapport détail le travail effectué sur le projet sur la simulation de producteur et consommateur.
Dans la réalisation du projet, j'ai implémenté deux versions du travail demandé :
L'un avec le multithreading et l'autre avec le multiprocessing.

Problématique :
Il s'agit de simuler un processus d'exécution distribuée. D'un côté on a des producteurs notamment qui sont des programmes qui effectuent des tâches et retournent les résultats de leurs travaux. D'un autre coté on a les consommateurs qui attendent les résultats retournés par les producteurs afin de s'en servir.

Méthode :
Dans cette simulation on aura besoin aussi d'accélérer les résultats, ce qui nous amène à faire intervenir les multithreading et multiprocessing qui sont des classes python pour paralléliser les exécutions.
Nous allons aussi nous servir de la classe Queue() de python pour stocker les produits par les producteurs.
Ces données stockées sont dans mon cas des entiers aléatoires compris entre **1** et **60.**

A-Classes Producteurs :
class ProducteurThread(Thread), class ProducteurProcess(Process)
Cette première classe permet de produire les items et les stocker dans la variable **queue** en ajoutant en fin de liste. On peut être confronté à un problème lorsque deux threads ou process accèdent instantanément à la liste **queue**. Dans ce cas on a mis en place des **verrou**s qui empêchent les autres threads d'ajouter dans la **queue** si un thread l'occupe déjà.

B-Classes Consommateurs :
class ConsumerThread(Thread), class ConsumerProcess(Process)
La deuxième classe permet de consommer ce qu'a produit la première classe en supprimant au début de liste. Comme les classes producteurs pour limiter l'accès instantané à la queue les threads ou process bloqués si la queue est déjà en occupée.

C- Les donctions Main de Multithreading et Multiprocessing :
def **main_threading**(nbre_producer, nbre_consumer, duration).
def **main_processing**(nbre_producer, nbre_consumer, duration).
nbre_producer: Nombre de thread qui sont des producteurs
nbre_consumer: Nombre de thread qui sont des consommateurs
duration : définie la durée d'exécution

D- Exécution
Dans la condition **if __name__ == "__main__"**
Il est possible de paramétrer le nombre de producteur **nbre_producer**, le nombre de consommateur **nbre_consumer**.
Il est aussi possible de définir la durée de simulation en seconde dans la variable **duration.**