Rapport de projet : Producteurs/Consommateurs

I. Introduction:

Notre projet s'agit d'un exemple informatique de synchronisation de ressources dans différents contextes de programmation concurrente, notamment en environnement multi-thread. Il s'agit de partager entre deux tâches (producteur et consommateur) une zone de mémoire tampon utilisée comme une file.

Le producteur génère une donnée et la dépose dans la file, simultanément, le consommateur retire les données de la file. La coopération Producteurs-Consommateurs est une communication de messages entre des processus appelés Producteurs et d'autres appelés Consommateurs.

II. Choix faits:

Nous avons commencé par deux ***class (Producer/Consumer)*** où chacun contient les méthodes ***wait(), run()*** et ***produce_item*** (***Producer)*** / ***consume_item (Consumer)***.

- ***produce_item()*** : Elle ajoute un chiffre aléatoire (***nums)*** dans un tableau (***items***) avec ***append*** ou dans une **Queue** .
- ***consume_item()***: Elle retire le premier chiffre ajouté dans un tableau avec ***pop()***, les chiffres aléatoires nous permettaient de voir si la valeur retirée était bien la première de la liste.
- ***wait()*** : ***time.sleep()*** y est intégré avec un temps aléatoire.
- ***run()*** : Le producteur va attendre puis produire, et le consommateur va attendre puis consommé.

III. Essai:

1) List:

Nous avons essayé de faire des Lock() pour que le consommateur et producteur s'alterne la file afin qu'il n'y ai pas de conflit, on a aussi ajouté des conditions à fin que le consommateur ne monopolise pas la file si elle est vide car cela va créer un inter-blocage entre les deux processus. On a aussi ajouté une condition sur le statut de liste afin qu'un consommateur n'essaye pas de tirer un produit d'une file vide.

2) Queue:

Pour un deuxième temps on a utilisé les queues, qui nous ont paru plus efficaces car le concept de verrouillage de ressource est directement inclus dans les conditions de la queue ce qui facilite grandement le fonctionnement, de plus dans une queue on a limité la taille de la file.

IV. Conclusion:

Pour conclure on peut dire que ce paradigme peut se résoudre avec plusieurs solutions même avec des Sémaphores, dans notre solution on a fait quelques tests pour vérifier le bon fonctionnement de notre implémentation.