# Exercices sur les Threads Sans Sémaphores

## Exercice 1 : Création de Threads de Base

Écrivez un programme où plusieurs threads affichent leur ID (ou un message) dans la console.

- **Tâches :**
  - Créez 5 threads.
  - Chaque thread doit afficher son ID ou un message unique.
  - Assurez-vous que tous les threads s'exécutent de manière indépendante.

**Exemple de sortie attendue :**

```
Thread 1 : Bonjour depuis le thread 1
Thread 2 : Bonjour depuis le thread 2
...
```

## Exercice 2 : Communication entre Threads avec des Variables Partagées

Créez un programme où deux threads communiquent à l'aide d'une variable partagée.

- **Tâches :**
  - Un thread incrémente une variable partagée, et un autre affiche sa valeur.
  - Implémentez une synchronisation de base à l'aide de `threading.Lock` ou d'autres mécanismes pour éviter les conditions de concurrence.

**Comportement attendu :** La variable partagée est mise à jour correctement, sans conditions de concurrence.

## Exercice 3 : Simulation d'une Condition de Concurrence

Créez un programme pour démontrer une condition de concurrence.

- **Tâches :**
  - Créez deux threads qui incrémentent chacun un compteur partagé 100 fois.
  - Exécutez le programme plusieurs fois et observez si le compteur atteint la valeur attendue.
  - Discutez de la manière dont les conditions de concurrence se produisent dans de tels scénarios.

**Comportement attendu :** Les étudiants devraient observer des incohérences dans la valeur finale du compteur.

## Exercice 4 : Producteur-Consommateur sans Sémaphores

Créez un programme producteur-consommateur en utilisant uniquement `threading.Condition` ou `threading.Event`.

- **Tâches :**
  - Un thread producteur génère des nombres et les place dans une liste partagée.
  - Un thread consommateur lit et supprime les nombres de la liste.
  - Utilisez une variable conditionnelle ou des événements pour assurer une bonne coordination entre les threads.

## Exercice 5 : Traitement de Fichier en Parallèle

Écrivez un programme où plusieurs threads traitent différentes parties d'un fichier texte en parallèle.

- **Tâches :**
  - Divisez un fichier texte en $N$ parties (une par thread).
  - Chaque thread compte le nombre de mots dans sa partie.
  - Combinez les résultats à la fin et affichez le total des mots comptés.

## Exercice 6 : Éviter les Deadlocks

Créez un scénario avec deux threads qui pourrait entraîner un deadlock en raison d'une contention sur les ressources.

- **Tâches :**
  - Simulez un deadlock en créant deux ressources partagées et deux threads qui les verrouillent dans un ordre différent.
  - Modifiez le code pour éviter le deadlock sans utiliser de sémaphores (par exemple, en ordonnant les ressources de manière cohérente).

## Exercice 7 : Implémentation d'un Pool de Threads

Écrivez une implémentation basique d'un pool de threads où vous gérez un nombre fixe de threads pour traiter un nombre dynamique de tâches.

- **Tâches :**
  - Créez une file d'attente de tâches (des instructions simples d'affichage).
  - Utilisez des threads pour consommer les tâches de la file.
  - Assurez-vous que les threads se terminent correctement lorsqu'il n'y a plus de tâches.

## Exercice 8 : Multiplication de Matrices avec des Threads

Écrivez un programme pour effectuer la multiplication de deux matrices à l'aide de threads.

- **Tâches :**
  - Attribuez chaque ligne de la matrice résultante à un thread séparé.
  - Combinez les résultats pour générer la matrice finale.

## Exercice 9 : Implémentation d'une Barrière Personnalisée

Implémentez une barrière personnalisée (point de synchronisation) sans utiliser de sémaphores ou de barrières standard.

- **Tâches :**
  - Utilisez un compteur partagé et `threading.Condition` pour bloquer les threads jusqu'à ce que tous aient atteint la barrière.

— Démontrez son utilisation avec plusieurs threads effectuant un calcul en plusieurs étapes.