# TP9 (3h) : Encore des threads + exercice récapitulatif

**Exercice 1 :**

Proposer une fonction permettant de calculer (A\*B)2 pour deux matrices carrées A et B, et permettant d’utiliser des *threads* pour calculer chaque élément de la matrice résultat en parallèle des autres. Les matrices opérandes et résultat peuvent être représentées par des **array numpy.**

Le nombre de threads utilisés est N2 ou N est la taille des matrices A et B. Un thread correspondant à l’élément (i,j) doit :

* calculer la valeur de l’élément (i,j) de A\*B
* attendre que le calcul de tous les autres éléments soit fini
* calculer la valeur de l’élément (i,j) de (A\*B)2

Pour les synchronisations, utiliser un objet de type **Barrier**, cf. [*https://docs.python.org/3/library/threading.html#barrier-objects*](https://docs.python.org/3/library/threading.html#barrier-objects) *.*

**Exercice 2 :**

*L’objectif de cet exercice est de réutiliser l’ensemble des notions vues dans le module*

**Nous reprenons dans ce sujet l’exercice sur la collecte de données de sondes de températures du TP8.**

**Rappel du fonctionnement :**

Un programme appelé « superviseur » collecte des données de températures à partir de plusieurs capteurs. Le nombre de capteurs est fourni en argument à l’appel du programme « superviseur ».

Les capteurs de températures génèrent un nombre aléatoire toutes les secondes.

Les données collectées sont stockées dans une matrice de taille N \* 10 où N est le nombre de capteurs. La matrice sert à stocker les données collectées en 10 secondes. Les données correspondant à chaque capteur sont stockées sur une ligne correspondante de la matrice, qui sera gérée comme une *liste circulaire*.

Toutes les 10 secondes le superviseur cherche la valeur maximale reçue par chaque capteur et l’écrit à l’écran, puis cherche la température maximale sur l’ensemble des capteurs.

**Dans ce TP**, nous souhaitons implémenter le comportement des capteurs par des processus fils du superviseur (un capteur = un fils).

La matrice de température sera stockée dans une région de mémoire partagée.

**Q1**. Ecrire un programme implémentant le comportement décrit. Le processus père doit créer N fils où N est le nombre de capteurs reçus en paramètre. Chaque fils génère un nombre aléatoire entre 0 et 32. Le père affiche les maximums périodiquement toutes les 10 secondes.

*Attention : le stockage de la matrice dans la mémoire partagée implique de gérer manuellement sa représentation dans un tableau unidimensionnel (vous ne pourrez plus utiliser numpy).*

**Q2**. Modifier le programme pour que le père tue ses fils et supprime la région de mémoire partagée lors de la réception du signal CTRL^C