TD #8 – Threads

Préambule

Objectif : Utilisation du mécanisme de threads, pour le partage de tâches

Création de thread

En utilisant l'API de gestion de threads, implémentez les solutions aux énoncés suivants :

- Affichage de "Hello World!" ;
- Affichage d'un entier aléatoire généré par le processus principal ;
- Affichage d'un entier aléatoire généré par le thread qui sera aussi affiché par le processus principal ;
- Affichage de la moyenne d'un tableau de n entiers générés aléatoirement par le processus principal.

Mécanisme de réduction pour le calcul

L'objectif de l'exercice est d'offrir à l'utilisateur une bibliothèque de fonctions et structures permettant le traitement d'un *grand* tableau dans un environnement multi-threadé. L'exercice se découpe en plusieurs parties.

Programme principal

L'exécution du programme se fait par la ligne de commande suivante :

```
$ ./reduction.py m n opcode
```

L'argument `m` définit le nombre de threads générés par le programme principal, `n` la taille du tableau et `opcode` l'opération à réaliser sur le tableau. Le programme doit dans un premier temps créer le tableau et générer les entiers le composant (entre 1 et 100), puis créer les threads. Il affiche, une fois l'exécution des threads terminée, le résultat obtenu **et uniquement** ce résultat.

Opcode

L'opcode ou code de l'opération est un identifiant permettant de distinguer quelle opération on souhaite effectuer. Nous allons considérer ici 4 symbôles pouvant être donner en entrée du programme, chacun correspondant à l'opération donnée dans la table ci(dessous) :

Symbole	Opération
`+`	Somme
`/`	Moyenne
`M`	Maximum
`m`	Minimum

Arguments de la fonction de thread

Les arguments à la fonction de thread sont composés de :

- Le tableau d'entier (dans sa totalité) ;
- Les indices de début et de fin de traitement (partie du tableau que le thread doit traiter).

Il faut également penser au stockage du résultat local à chaque thread. L'une des manières de faire est de donner à chaque thread un tableau de taille `m`, où le thread `i` remplirait la case `i` avec la valeur locale qu'il a calculée.

Exécution du thread et post-traitement

Il ne reste plus qu'à lancer les threads, récupérer les résultats et, si nécessaire opérer un dernier traitement sur les résultats locaux aux threads pour obtenir un résultat global.