Variables critiques et exclusion mutuelle

Nous supposons que votre plate-forme de développement supporte les threads C11 et les variables et opérations atomiques. Pour cela nous vous suggérons d'utiliser la lib C alternative "musl" avec une version de gcc moderne :

```
musl-gcc -static -Wall -O3 -std=c11 -o toto toto.c
```

Pour avoir accès aux opérations atomiques et aux threads C11 il faut les includes dans votre programme :

```
#include <stdatomic.h>
#include <threads.h>
```

Le but de cet exercice est de créer 4 exécutables et de les comparer.

Gardez une copie de votre source pour chacun des exercices !

Exercice 1

Écrivez un programme composé d'un processus `main` et de n thread ayant accès à une variable partagée de type `size_t`, où n est fournie en ligne de commande. Chaque thread incrémente un million de fois ce compteur. Une fois les threads terminés, affichez la valeur du compteur.

Faites des variantes de votre programme qui utilisent différents écritures pour l'incrément, p.ex `count++` ou `count = count + 1`, compilez votre programmes avec différentes options d'optimisation, notamment `-O0` et `-O3`, et exécutez le programme avec un nombre de thread varié (p.ex 1, 2, 4 et 8).

Que constatez-vous ?

Exercice 2

Pour résoudre le problème vu dans la première question, protégez la section critique avec un `mtx_t`.

Exercice 3

Pour résoudre le problème vu dans la première question, protégez la section critique avec un `atomic_flag`.

Exercice 4

Pour résoudre le problème vu dans la première question, réalisez la variable critique avec un `_Atomic(size_t)` au lieu d'un simple `size_t`.

Exercice 5

Exécutez les trois différents programmes correctes avec 8 thread chacun et mésurez les temps d'exécution avec `/usr/bin/time`.