HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Prof. Yann Thoma

Laboratoire de Programmation Concurrente

semestre printemps 2015 - 2016

Prise en main des threads Qt

Temps à disposition : 4 périodes (travail débutant en semaine 2)

1 Objectifs pédagogiques

— Se familiariser avec un environnement de programmation ;

— Réaliser son premier programme concurrent en C++ avec la bibliothèque Qt ;

— Optimiser un calcul mathématique.

2 Cahier des charges

Nous désirons implémenter un programme nous affichant si un nombre passé sur la ligne de commande est premier ou non. Bien entendu, nous sommes intéressés par une implémentation multi-threadée. [1]

Partie 1

La première partie de cet exercice est de développer une version séquentielle de l'algorithme.

Pour ce faire, écrivez un programme qui prend en argument le nombre à tester et qui affiche *prime number !* si le nombre est premier et *NOT a prime number !* dans le cas contraire.

Pour vérifier qu'un nombre entier n est premier ou non, il faut simplement vérifier s'il peut être divisé par un des nombres compris entre 2 et $\sqrt{n}$.

A noter que le nombre premier peut être potentiellement grand, donc stockez-le dans un entier long non signé (`uint64_t`). Attention : un `uint64_t` peut contenir au maximum 18 chiffres !

Partie 2

Dans cette deuxième partie nous voulons améliorer les performances de notre implémentation en développant une version multi-threadée de l'algorithme. Notre programme prendra alors en premier argument le nombre à tester et le nombre de threads à utiliser en deuxième argument.

A noter que dans votre implémentation aucun des threads ne devra utiliser la fonction exit pour terminer le programme. Pensez-donc à utiliser une politique d'annulation afin d'optimiser l'utilisation du processeur (pas de calcul pour rien).

1. Ce labo est inspiré par une proposition de Florent Gluck.

Partie 3

Comparez la vitesse d'exécution entre vos deux implémentations, notamment en changeant le nombre de threads.

Le polycopié du cours vous donne les informations nécessaires au calcul du temps. Placez ce qui est nécessaire dans votre programme pour évaluer le temps nécessaire et afficher le temps total.

Questions

1. Mesurez le gain de temps produit par votre version multi-threadée en faisant varier le nombre de threads. Que remarquez-vous ?
2. Le gain de vitesse est-il linéaire avec le nombre de threads ?

Informations utiles

— Vous pouvez tester votre programme avec les nombres premiers suivants : 433494437, 2971215073, 68720001023, 4398050705407, 70368760954879, 18014398241046527, 99194853094755497

— La fonction `std::stoull` permet de convertir une chaîne de caractères en un **`unsigned long long`**

— La fonction `sqrt` de la libraire math.h renvoie la racine carrée d'un double

3 Travail à rendre

— Les modalités du rendu se trouvent dans les consignes qui vous ont été distribuées.

— La description de l'implémentation, ses différentes étapes, la manière dont vous avez vérifié son fonctionnement et toute autre information pertinente doivent figurer dans les programmes rendus. Aucun rapport n'est demandé.

— Inspirez-vous du barème de correction pour connaître là où il faut mettre votre effort.

— Vous pouvez travailler en équipe de deux personnes au plus.

4 Barème de correction

Conception	25%
Exécution et fonctionnement	20%
Codage	15%
Documentation et en-têtes des fonctions	30%
Commentaires au niveau du code	5%
Robustesse	5%