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

Yann Thoma
Florian Vaussard
Olivier Cuisenaire

Laboratoire de Programmation Concurrente
semestre automne 2024

Bogosort

Temps à disposition: 4 périodes

1 Objectifs pédagogiques

- Se familiariser avec la gestion des threads
- Stopper élégamment des threads.

2 Récupération des sources

La récupération du code source s'effectue à l'aide de la commande suivante:

```
retrieve_lab pco24 lab02
```

3 Cahier des charges

Développer un programme capable de trier une séquence de chiffres générée aléatoirement avec l'algorithme Bogosort déterministe. Une application vous est fournie, mais est incomplète. Il en manque la partie multi-threadée.

Depuis la classe `threadmanager.cpp`, dans la fonction `startSorting` vous devez implémenter le code nécessaire pour démarrer le nombre de threads demandé, attendre le résultat et retourner la séquence triée sous forme de `QVector`. Pour le code des threads et donc l'implémentation parallélisée du Bogosort déterministe, il est demandé de placer votre code dans les fichiers `mythread.h` et `mythread.cpp`. Vos modifications devront être réalisées à l'aide de la librairie `PcoSynchro` avec des threads `PcoThread`. Essayez d'avoir une solution la plus rapide possible. Quand un thread trouve la solution, il est nécessaire de stopper l'ensemble des threads.

La barre de progression est contrôlée par la fonction

```
void ThreadManager::incrementPercentComputed(double percentComputed)
```

A vous de faire en sorte de pouvoir utiliser cette fonction depuis vos threads afin d'avoir une progression pertinente.

Au niveau des étapes de développement il est suggéré de commencer par la création de threads, l'implémentation d'une fonction `bogosort` dans les fichiers `mythread.h` et `mythread.cpp`, et du passage des paramètres nécessaires à ces threads.

⚠Réfléchissez bien à la manière de répartir le travail entre les différents threads.

Une fois que votre solution est fonctionnelle, réfléchissez à une manière de mettre en place une terminaison des threads permettant d'obtenir le résultat sans devoir attendre toutes les exécutions.

⚠Attention à la manière de partager de l'information entre les threads, et notamment de ne pas

accéder à des emplacements mémoires non alloués ou à des threads non existants.

4 Au cas où

Lors de la compilation de votre code il se pourrait que vous vous trouviez devant des messages plutôt hermétiques, comme ceux-ci:

```
In file included from ../solution_code/src/mythread.h:5,
from ../solution_code/src/threadmanager.cpp:4:
In instantiation of 'PcoThread::PcoThread(Fn&&, Args&& ...) [with Fn = void (*)(PasswordParam*, ThreadManager*); Args = {PasswordParam*, ThreadManager*, int}]':
required from 'typename std::_MakeUniq<_Tp>::__single_object std::make_unique(_Args&& ...) [with _Tp = PcoThread; _Args = {void (*)(PasswordParam*, ThreadManager*), PasswordParam
required from here
no matching function for call to 'invoke(void (* const&)(PasswordParam*, ThreadManager*), PasswordParam* const&, ThreadManager* const&, const int&)'
In file included from /usr/include/c++/9/pstl/glue_algorithm_defs.h:13,
from /usr/include/c++/9/algorithm:71,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qglobal.h:142,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qnamespace.h:43,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qobjectdefs.h:48,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qobject.h:46,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/QObject:1,
from ../solution_code/src/threadmanager.h:16,
from ../solution_code/src/threadmanager.cpp:3:
candidate: 'template<class _Callable, class ... _Args> std::invoke_result_t<_Callable, _Args ...> std::invoke(_Callable&&, _Args&& ...)'
note:   template argument deduction/substitution failed:
In file included from /usr/include/x86_64-linux-gnu/qt5/QtCore/qglobal.h:45,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qnamespace.h:43,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qobjectdefs.h:48,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/qobject.h:46,
from /usr/include/x86_64-linux-gnu/qt5/QtCore/QObject:1,
from ../solution_code/src/threadmanager.h:16,
from ../solution_code/src/threadmanager.cpp:3:
required by substitution of 'template<class _Callable, class ... _Args> std::invoke_result_t<_Callable, _Args ...> std::invoke(_Callable&&, _Args&& ...) [with _Callable = void (* const&)(Passwor
required from 'PcoThread::PcoThread(Fn&&, Args&& ...) [with Fn = void (*)(PasswordParam*, ThreadManager*); Args = {PasswordParam*, ThreadManager*, int}]'
required from 'typename std::_MakeUniq<_Tp>::__single_object std::make_unique(_Args&& ...) [with _Tp = PcoThread; _Args = {void (*)(PasswordParam*, ThreadManager*), PasswordParam
required from here
no type named 'type' in 'struct std::invoke_result<void (* const&)(PasswordParam*, ThreadManager*), PasswordParam* const&, ThreadManager* const&, const int&>'
In file included from /usr/include/c++/9/memory:80,
from /usr/include/c++/9/thread:39,
from /usr/local/include/pcosynchro/pcothread.h:23,
from ../solution_code/src/mythread.h:5,
from ../solution_code/src/threadmanager.cpp:4:
```

Il s'agit ici simplement du fait que la création de `PcoThread` pose problème car les arguments passés au constructeur ne correspondent pas aux arguments demandés par la fonction exécutée par le thread. Si vous tombez sur ce type de messages pensez donc à vérifier la manière dont vous créez les threads.

5 Travail à rendre

- Ne pas créer de nouveau fichier. Modifiez et utilisez judicieusement les fichiers `mythread.h` et `mythread.cpp`, ainsi que la fonction `startSorting()`.
- 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 un fichier nommé `rapport.pdf`.
- Inspirez-vous du barème de correction pour savoir là où il faut mettre votre effort.
- Vous devez travailler en équipe de deux personnes.

- L'archive à rendre doit être générée avec le script de rendu $\texttt{pco}_r endu.sh$

6 Barème de correction

Conception	25%
Exécution et fonctionnement	5%
Codage	10%
Documentation et analyse	60%