HEIG VD HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD

Yann Thoma
Florian Vaussard

Laboratoire de Programmation Concurrente

semestre automne 2023

Cracking md5

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 pco23 lab02
```

3 Cahier des charges

Nous sommes intéressés à développer un programme capable de cracker un hash md5 pour récupérer un mot de passe. Une application vous est gracieusement fournie, mais souffre de quelques carences en termes de performances. En effet, elle n'est pas multi-threadée pour les calculs à faire.

Le code correspondant au coeur de calcul se trouve dans la classe `threadmanager.cpp`, dans la fonction `startHacking()` pour le moment. A vous de modifier ce code de manière à lancer le nombre de threads demandé, d'attendre le résultat et de retourner le mot de passe trouvé. 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. Il n'est par exemple pas nécessaire de laisser tourner tous les threads de calcul si l'un d'eux a trouvé la réponse. La fonction `startHacking()` devra être modifiée afin de lancer les threads au lieu de faire le travail. Pour le code des threads il est suggéré d'utiliser les fichiers `mythread.h` et `mythread.cpp` pour y placer votre code.

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, 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.

Pour faire vos tests, le site web suivant vous permet de générer des hash md5 :

`https://emn178.github.io/online-tools/md5.html`

Si vous voulez utiliser un sel, alors le sel devra être placé avant le mot de passe sur ce site web. Par exemple, si votre sel est *xy* et votre mot de passe *mute*, alors placez *xymute* dans la saisie web.

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 `startHacking()`.
- 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 a rendre doit être générée avec la script de rendu

6 Barème de correction

Conception	40%
Exécution et fonctionnement	5%
Codage	20%
Documentation et analyse	25%
Commentaires au niveau du code	10%