Exercices de programmation multi-tache avec l'API Pthread

Guillermo Andrade B.

Ingénieur de recherche IRISA/INRIA Rennes

Guillermo.Andrade@irisa.fr

- Pour ces exercices, nous utilisons le portage de pthread pour windows 32 http://sourceware.org/pthreads-win32/
- Vous disposez d'une répertoire TP_pthread avec le fichier main.cpp qui est le point de départ des exercices.
- Les sorties consoles utilisent les flux **std::cout** la librairie STL (standard en C++). Nous vous préoccupez pas, la syntaxe est relativement simple à comprendre. Elle permettrons d'illustrer les exercices.

1 Exercice de démarrage

- Compiler et lancer l'exécutable avec la commande :

  g++ main.cpp -lpthread -O3 -o tp_pthread
- Que pouvons nous constater sur la sortie de la console?

2 Multiples Threads

- Modifiez le fichier main.cpp pour lancer un nombre quelconque de pthreads. Ce nombre devra être défini par une constante (ici bas le nombre de threads est 3):

  ```
  const long int numberThreads=3;
  ```
- Chaque thread secondaire devra afficher dans le message son numéro de création **id**:

  std::cout << "Hello World from thread "<< id<<" created by the main thread!"<< std::endl;
- Utilisez le paramètre passé au thread à sa création pour transmettre le numéro id.
- Lancez plusieurs exécutions avec un nombre de threads différents.
- Que constatez vous sur la sortie de la console?

3 Protection de la sortie par des Mutex

- Pour protéger les sorties consoles utilisez un Mutex que vous verrouillerez avant les appels à **std::cout** et que vous déverrouillerez après ces appels.
- Vérifiez que les phrases ne sont plus coupées quelque soit le nombre de threads lancés.

4 Ping et Pong en utilisant les conditions et les mutex (exercice long)

- Dans cet exercice, vous modifierez le programme précèdent pour lancer deux

threads secondaires A et B. Ces deux threads simulerons deux joueurs de tennis qui se renvoient la balle à tour de rôle.

- La position de la balle est décrite par une variable globale X de type float.
- La vitesse de la balle est décrite par une variable V dont la valeur absolue est égale à 2 m/s
- l'intervalle de temps entre deux animations est dT= 1 s.
- pour chaque itération, la nouvelle valeur de X se calcule X= V * dT + X
- Chaque thread secondaire doit changer le sens de la vitesse d'une balle imaginaire qui se déplace dans un axe.
  - Le Thread A change le sens de la balle (V = -V) lorsque celle-ci attends -10 m.
  - Le thread B change le sens de la balle (V = -V) lorsque celle-ci attends +10 m.
- Chaque thread secondaire anime la balle à tour de rôle.
  - Si X <0 le thread A anime la balle
  - Si X >=0 le thread B anime la balle
  - Le thread qui « n'a pas la balle » doit attendre jusqu'à l'arrivée de la balle dans son terrain.
- Le thread principal va être chargé d'afficher la position de la balle à chaque intervalle de temps dT.

- Utilisez les mutex et les conditions associés pour coordonner l'ensemble de threads. Le but est de maintenir la charge des processeurs au minimum.
- Pour contrôler l'animation, utilisez la fonction **sleep(sec)** qui permet de mettre en pause le thread pour un nombre donnée de secondes **sec**. Pour inclure cette fonction ajoutez ceci en début dans le haut de votre code:

```
#include <unistd.h>
```