TP 1 : utilisation de `pthread`

Systèmes et supports d'exécution — 2005–2006

Alexandre DENIS
denis@labri.fr
http ://dept-info.labri.fr/~denis/

L'objectif de ce TP est d'effectuer vos premiers pas en `pthread` et d'étudier des schémas « classiques » de synchronisation. Vous effectuerez également vos premières mesures de performances en contexte *multi-thread*.

Utilisation de `pthread`

La programmation se fera en langage `C`. Pour utiliser la bibliothèque `pthread`, votre code doit contenir un :

```
#include <pthread.h>
```

et l'édition de lien doit se faire avec un `-lpthread`, c'est-à-dire :

```
% gcc -o toto toto.o -lpthread
```

En cas de doute sur la syntaxe d'un appel `pthread`, n'hésitez pas à **utiliser la page de `man`**. Pour retrouver un appel `pthread` dont vous auriez oublié le nom :

```
% info -a pthread
```

Mesure de performances

Pour mesurer des temps en `C`, nous vous proposons une bibliothèque plus précise que les appels systèmes standard. Vous trouverez cette bibliothèque, ainsi que sa documentation, sur la page web du cours accessible à partir de :
`http ://dept-info.labri.fr/~denis/`

Travail à faire

1. Premiers pas en `pthread`

- Écrivez un programme qui crée deux *threads*, chaque *thread* étant constitué d'une boucle affichant un message.
- Que constatez vous à l'affichage ? Comment les messages sont-ils entrelacés ?

2. Utilisation de *mutex*

- Écrivez un programme qui crée deux *threads*, chaque *thread* étant constitué d'une boucle incrémentant un compteur global 10^{10} fois — attention, tous les compteurs doivent être des `long long` et non des `int`.
- Qu'observez-vous à l'exécution ? Le comportement est-il déterministe ?
- Ajouter le verrouillage nécessaire pour que le résultat soit correct.
- Que constatez-vous avec cette nouvelle version ? La vitesse d'exécution est-elle identique à la version sans verrou ?
- À partir du temps global d'exécution (mesuré à l'aide de la commande `time`), évaluez le coût introduit par l'utilisation du *mutex*.

3. Synchronisation entre *threads*, principe de la « *boîte aux lettres* »

Dans cette partie, nous concevons un programme qui incrémente un compteur de 1 à 1 000 000, l'incrémentation normale étant faite par un *thread*, la propagation des retenues étant faite par un autre *thread*.

Soient deux *threads*, `tid1` et `tid2` et un compteur `i`. Si `i % 10 != 0` alors le *thread* `tid1` incrémente `i` ; sinon, il attend que l'autre *thread* le fasse. Le *thread* `tid2` attend que `i % 10 == 0`, et incrémente alors `i` et affiche sa valeur à l'écran.

- Écrire un tel programme et observer son comportement.
- indication : les *threads* doivent s'attendre mutuellement. Il faut donc utiliser *deux* conditions.

4. Mesure des performances de `pthread`

Vous allez imaginer, écrire, et faire tourner un programme capable de mesurer le temps que prennent les opérations suivantes :

- exécution complète d'un *thread* qui ne fait rien ;
- prendre et relâcher un *mutex* ;
- délai entre le moment où un *thread* fait un `pthread_cond_signal` et le moment où le *thread* qui attendait sur un `pthread_cond_wait` est effectivement débloqué.