Mini EP9: Thread Pool

Alfredo Goldman, Elisa Silva e Luciana Marques
MAC 0219-5742 – Programação Concorrente e Paralela 2021

Due date: July 10th, 2021

1. Introduction

Thread pool is a technique to reduce the overhead in applications that need to create many threads through the concept of tasks and a pool of threads that are reused. The thread pool works over a queue that keeps the tasks to be executed, and the threads consume the tasks from the queue. The threads from the pool keep working until they receive a special "end" task.

In this mini EP you are going to finish an implementation of a program that demonstrate how thread pool works.

We are not going to implement at the moment the support for complex tasks that may pause its execution for a later return (yield).

2. Tasks

Download from eDisciplinas the file **"**threadpool.c.". You must read it and complete the implementation of the thread pool.

You may modify the program as you wish and probably will use **mutexes** from the pthreads to code the solution.

Follows an example of the output of a completed program modified to generate 9 tasks and run in 3 threads:

```
Adding 0th task to sleep 1
Adding 1th task to sleep 4
Adding 2th task to sleep 5
Adding 3th task to sleep 8
Adding 4th task to sleep 7
Adding 5th task to sleep 2
Adding 6th task to sleep 6
Adding 7th task to sleep 1
Adding 8th task to sleep 4
Thread 0 Running task sleep 1s
Thread 2 Running task sleep 5s
Thread 1 Running task sleep 4s
Thread 0 Running task sleep 8s
Thread 1 Running task sleep 7s
Thread 2 Running task sleep 2s
Thread 2 Running task sleep 6s
Thread 0 Running task sleep 1s
Thread 0 Running task sleep 4s
Thread 1 exiting
Thread 2 exiting
Thread 0 exiting
```

3. Submission

Add in the beginning of the program your name and USP number, and send the optimized program file to eDisciplinas.

The due date is July 10th, 2021.

4. Score Criteria

The mini EPs utilize a binary score criteria (1 or 0). To get 1 you must submit the file as specified to eDisciplinas. Code quality will not be scored, but do your best.

It is worth reinforcing paragraph II of article 23 of the **USP Code of Ethics**:

> Article 23 - Members of the student body and other students of the University are prohibited from:
> [...]
> II. make use of means and devices that can defraud the performance evaluation, yours or of others, in academic, cultural, artistic, sports and social activities, within the scope of the University, and cover up the possible use of these means.

Plagiarized Mini EPs will receive a score of 0.

Any questions, send a message at the forum or send an email to elisa@silva.moe, lucianadacostamarques@gmail.com, or gold@ime.usp.br with *[miniEP9]* in the subject. Have fun!

5. Extra

A tip to implement a function that only blocks under certain conditions, it to use conditional unlocks. So, consider a function A that has locked a mutex, if it needs to block the next call to A until B is called, it does not unlock the mutex and when B is called, it does the unlock of the mutex of A, unblock eventually the future call of A.

Search on the web about thread pool and mutexes. Modify parts of the program such as number of generated tasks or the maximum size of the queue to help in the development of the program.