Assignment-5 (Section-A)

SimpleMultithreader: Using Multithreading with Ease

Due by 11:59pm on 22th November 2025
(Total **7%** weightage)
Instructor: Vivek Kumar

No extensions will be provided. Any submission after the deadline will not be evaluated. If you see an ambiguity or inconsistency in a question, please seek clarification from the teaching staff.

Plagiarism: **This is a pair programming-based assignment that you must do with the group member that you have already chosen. No change to the group is allowed. You are not allowed to discuss the approach/solution outside your group.** You should never misrepresent some other group's work as your own. In case any plagiarism case is detected, it will be dealt as per the new plagiarism policy of IIITD **and will be applied to each member in the group.** Even if you are a single member group, there will not be any relaxations in marking scheme/deadlines, and the same rubric will be followed for each group.

Open-sourcing of this assignment solution is not allowed, even after the course gets over.

General Instructions

a) Hardware requirements:
   a) You will require a machine having any Operating System that supports Unix APIs. You should not use MacOS for OS assignments. You can either have a dual boot system having any Linux OS (e.g., Ubuntu), or install a WSL.
b) Software requirements are:
   a) C compiler and GNU make.
   b) You **must** do version controlling of all your code using github. You should only use a **PRIVATE** repository. If you are found to be using a **PUBLIC** access repository, then it will be considered plagiarism. NOTE that TAs will check your github repository during the demo.

Assignment Details

You would recall from Lecture 21 how much programming effort someone has to spend if they want to parallelize an algorithm using Pthreads. For example, the parallel version of the simple array sum discussed in Lecture 21 had around 3x lines of code than its corresponding sequential implementation.

In this assignment, you will have to help the Pthread programmers by abstracting away some of the essential programming efforts into a header-file-only implementation ("**simple-multithreader.h**") that the programmer can include and use in their program. We have provided two C++ sample programs with this assignment that use the SimpleMultithreader. The SimpleMultithreader provides C++11 lambda expressions-based APIs. You can read about C++11 lambda expressions from these easy-to-understand links (only look for C++11-supported lambda):

https://riptutorial.com/cplusplus/example/1854/what-is-a-lambda-expression-
https://blogs.embarcadero.com/lambda-expressions-for-beginners/

A sample declaration and usage of C++11 lambda expressions are also demonstrated inside the simple-multithreader.h file provided along with this assignment. Makefile is also provided herewith to help you in the compilation.

NOTE that this assignment requires C++ implementation, but you don't need to use any features of the C++ other than C++11 lambda function. You can simply use C programming, but inside a C++ file. We have provided the makefile also to ease your work.

1. Signature of Methods Supported by SimpleMultithreader

Following are the signature of the two methods that the programmer can use for exposing the Pthread based parallelism supported by SimpleMultithreader.

```
// parallel_for accepts a C++11 lambda function and runs the loop body (lambda) in
// parallel by using 'numThreads' number of Pthreads to be created by the simple-multithreader
void parallel_for(int low, int high, std::function<void(int)> &&lambda, int numThreads);

// This version of parallel_for is for parallelizing two-dimensional for-loops, i.e., an outter for-i loop and
// an inner for-j loop. Loop properties, i.e. low, high are mentioned below for both outter
// and inner for-loops. The suffixes "1" and "2" represents outter and inner loop properties respectively.
void parallel_for(int low1, int high1,  int low2, int high2,
                                        std::function<void(int, int)>  &&lambda, int numThreads);
```

2. Implementation

a. SimpleMultithreader **must not use** any concept of task/thread pool. SimpleMultithreader should simply create Pthreads whenever parallel_for APIs are invoked in the user program. You should implement SimpleMultithreader in C++ inside the header file provided herewith. **You are not allowed to use C++11 threading APIs (std::thread) or parallel programming APIs (e.g. std:async, etc.)**
b. SimpleMultithreader runtime execution **must** have the exact number of threads specified by the programmer including the main thread of execution.
c. Every call to SimpleMultithreader interfaces (parallel_for) will create a new set of Pthreads and they will terminate as soon as the scope of that interfaces has ended.
d. Your code should be modular and must avoid code repetitions.
e. We will evaluate your implementation of SimpleMultithreader using the two examples provided herewith without any changes. You should not do any changes to these examples as for evaluation the instructor will use his own copy of these examples.
f. SimpleMultithreader must also print the total execution time for each call of a parallel_for.

3. Requirements

a. You should strictly follow the instructions provided above.
b. Proper error checking must be done at all places. It's up to you to decide what are those necessary checks.
c. Proper documentation should be done in your coding.
d. Your assignment submission should consist of two parts:
    a. A zip file containing your source files as mentioned above. Name the zip file as "group-ID.zip", where "ID" is your group ID specified in the spreadsheet shared by the TF.
    b. A design document **inside the above "zip"** file detailing the contribution of each member in the group, detailing your **SimpleMultithreader** implementation, and the link to your **private** github repository where your assignment is saved.
e. There should be **ONLY ONE** submission per group.
f. In case your group member is not responding to your messages or is not contributing to the assignment then please get in touch with the teaching staff immediately.