CS/SE 4348: Operating Systems Concepts
Section 0U2
Programming Project 2

Instructor: Neeraj Mittal

Assigned on: Monday, July 1, 2024
Due date: Tuesday, July 16, 2024 (at midnight)

This is an individual assignment. You are expected to write the code independently and submit your own work. However, you can freely discuss the implementation ideas with other students in the class. Copying or using work not your own will result in disciplinary action and the suspected incident will be referred to the Office of Community Standards and Conduct for investigation!

1 Project Description

Implement n-thread mutual exclusion using the following three algorithms discussed in the class:

(a) *Tournament-Tree (TT)* based, where each node in the tree is an instance of Peterson's algorithm,

(b) *Test-And-Set (TAS)* based, and

(c) *Fetch-And-Increment (FAI)* based.

The most elegant (and, in my opinion, also easiest) way to implement a TT-based lock is by using (a) an array (or vector) of Peterson's locks *numbered* appropriately and (b) recursion for both `acquire` and `release` methods. I will highly recommend writing a pseudocode for the TT-based lock before writing a single line of code.

Your program should take two command-line arguments. The first argument specifies the algorithm type: 0 for TT based algorithm, 1 for TAS based algorithm and 2 for FAI based algorithm. The second argument specifies the value of n. Note that n need not be a power of two. You will need to use the atomic library available in C++11 and later versions (`https://en.cppreference.com/w/cpp/header/atomic`) for this project. When implementing Peterson's algorithm, declare all your shared variables as `atomic`. Further, both TAS and FAI instructions are also available in the atomic library.

You can write your program in C or C++. Name your program as `my-lock`. *Ensure that your program runs on one of the department machines cs1.utdallas.edu, cs2.utdallas.edu or giant.utdallas.edu.* Any significant deviation from the description without prior approval may result in substantial penalty.

2 Grading Criteria

As such, projects will be graded with these criteria in mind:

- Solutions must adequately address the problem at hand. Specifically:
  - The solution represents a good-faith attempt to actually address the requirements for the assignment.
  - The program complies and executes.
  - The program runs correctly.
- The solution constitutes a high quality product expected of a professional. Specifically:
  - The program is easy to read and to understand, that is, it is well commented. In addition, method and variable names are meaningful, all potentially confusing/complex code is well documented.
  - The general design of the program is clear and reasonable.
  - All procedure and function headers include comments explaining what the method is supposed to do (not how it does it) and the purpose of each formal parameter. Be as precise and careful as you can be.
  - The program is robust and handles important errors and exceptions properly.

3 Submission Information

You have to submit your project through eLearning. Along with all the source files, submit the following: (i) a Makefile to compile the program and (ii) a README file that contains the names of all the group members and instructions for running the compiled program. *Points will be deducted if you fail to submit either a Makefile or README file.*