CS/CE/TE 6378: Advanced Operating Systems
Section 003
Programming Project 2

Instructor: Neeraj Mittal

Assigned on: Monday, February 24, 2014
Due date: Friday, March 28, 2014 (at midnight)

This is a group project. *Code sharing among groups is strictly prohibited and will result in disciplinary action being taken.* Each group is expected to demonstrate the operation of this project to the instructor or the TA.

You can do this project in C, C++ or Java. Since the project involves socket programming, you can only use machines netXX.utdallas.edu, where XX $\in$ {01, 02, .., 45}, for running the program. Although you may develop the project on any platform, the demonstration has to be on netXX machines; otherwise you will be assessed a penalty of 20%.

1 Project Description

Implement a mutual exclusion service. Your service should provide two function calls to the application: cs-enter and cs-leave. The first function call cs-enter allows an application to request permission to start executing its critical section. The function call is blocking and returns only when the invoking application can execute its critical section. The second function call cs-leave allows an application to inform the service that it has finished executing its critical section.

Conduct literature search and select a distributed mutual exclusion algorithm of your choice to implement the mutual exclusion service. The algorithm should be different from the ones covered in the class. For your convenience, a list of pre-selected distributed mutual exclusion algorithms that you can use for this project will be provided to you. You are not required to choose an algorithm from this list. But, if you choose one from outside the list, then you have to obtain our approval.

Implementation Details: Design your program so that each process or node consists of two separate modules–one module that implements the application (requests and executes critical sections) and one module that implements the mutual exclusion algorithm (coordinates critical section executions of all processes so that they do not overlap). Intuitively, the two modules interact using cs-enter and cs-leave function calls. Each module in turn may be implemented using one or more threads. *It should be possible to swap your application module with our own application module and your program should still compile and run correctly!*

Testing: Design a mechanism to *test* the correctness of your implementation. Your testing mechanism should ascertain that at most one process is in its critical section at any time. It should be as *automated* as possible and should require minimal human intervention. *For example, visual inspection of log files to verify the correctness of the execution will not be acceptable.* You will be graded on how accurate and automated your testing mechanism is.

2 Submission Information

You will have to submit your project using eLearning. Submit all the source files necessary to compile the program and run it. Also, submit a README file that contains instructions to compile and run your program.