Operating Systems-2
CS 3510 Spring 2019
Lab Exam Report

INSTRUCTOR: DR. SATHYA PERI

Report By:
Vijay Tadikamalla
CS17BTECH11040

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Task

The goal of this assignment is to implement a program to simulate generalization of the mutual exclusion problem

Appoarch

We implement the algorithms using the standard algorithms and then compare their performances over several tests.

Design of Program

We have a testCS() which will be executed on n threads in our program.

Critical Section & Remainder Section
We simulate these using an exponential random distribution for run times and call sleep for that duration.

And finally, we keep a track and log of all points when :
An Entry Request occurs : a task wishes to enter the CS
An Entry occurs : a task has just entered its CS
An Exit occurs. : a task has just left its CS
All of these are then logged to the output. These are then plotted for a graphical comparison.

Approach and Implementation

1. To achieve our above mentioned goal make a *testCS* function which takes thread_index as a parameter. We will pass this function along with the thread index to the each thread and calculate the average waiting time and max waiting time for all three mutual exclusion algorithm.

```
vector<thread> t; // Array of n threads
for(int i=0;i<n;i++) t.push_back(thread(testCS,i));
```

```
for(int i=0;i<n;i++) t[i].join();
```

2. Functions and data-types of the chrono library (and other libraries) like
    a. **std::chrono::system_clock, std::chrono::system_clock::now()**
    b. **std::chrono::time_point**
    c. **struct tm - Time structure**
    d. **struct tm *localtime(const time_t *timer)**

    were used to calculate the average waiting time and max waiting time.
3. **int usleep(useconds_t *usec*)** function was used to suspend the execution of the thread for microsecond intervals
4. **template <class RealType = double> class exponential_distribution**: This is a random number distribution that produces floating-point values according to an exponential distribution, which is described by the following probability density function:

    $$p(x|\lambda) = \lambda e^{-\lambda x} \quad , \; x > 0$$

5. We make two exponential distributions and pass the value of 1/λ1, 1/λ2 in the constructor. Later this can be used to obtain random numbers t1 and t2 with values that are exponentially distributed with an average of λ1, λ2 seconds.

```
distribution1 = new exponential_distribution<double>(1/lt1);
distribution2 = new exponential_distribution<double>(1/lt2);
```

6. A uniform_int_distribution is used for generating session number
7. ME algorithm can be generated by simply commenting the utility functions

Input Parameters

N: No. of threads vary from 20 to 100 in steps of 10.
K = 20: No. of CS request by each thread
S = 5
λ1=1, λ2=1