Assignment 8

Operating System Lab (**CS342**)

Department of CSE, IIT Patna

Date:- 30-Mar-2021 **Time:**- 6 hours

Instructions:

1. All the assignments should be completed and uploaded by 9:00 pm. Marks will be deducted for the submissions made after 9:00 pm.
2. Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
3. Proper indentation and appropriate comments (if necessary) are mandatory.
4. You should zip all the required files and name the zip file as ***roll_no*.zip**, eg. **1701cs11.zip.**
5. Upload your assignment (**the zip file**) in the following link: https://www.dropbox.com/request/ZcJJCoAHLcyOLBupHNwy

Questions:

1. Write a program of creating two threads (pthread1 and pthread2) where each thread call a particular function. Apply a **mutex lock** (acquiring a lock and releasing a lock) on the function show that if a thread access the function using mutex lock the other thread cannot access the function.

    Sample output:

    Lock acquired by pthread1
    pthread1 is accessing function f()
    pthread2 cannot access function f() lock acquired by pthread1
    Lock released by pthread1
    Lock acquired by pthread2.
    pthread2 is accessing function f()

2. Write a program to find the sum of all the primes from 0 to 10000. Use 4 threads, each thread will iterate over ¼ of the range and will add the prime numbers to a global variable. The global variable should be shared between threads.

3. In this assignment, you implement the ***dining philosophers problem***. Here each philosopher grabs the two forks one by one – first the left fork, and then after some waiting the right fork. The parent process checks at regular intervals whether a deadlock has occurred. If so, it chooses a philosopher randomly and releases the fork (***the left one actually***) grabbed by him. Maintain a ***resource graph*** using shared memory. The parent process periodically checks for a deadlock (cycle) in the shared resource graph. Use ***semaphores*** for synchronization and mutual exclusion.

In both the programs, print suitable diagnostic messages, like the following:

Philosopher 3 starts thinking
Philosopher 2 starts eating
Philosopher 0 grabs fork 0 (left)
Philosopher 4 ends eating and releases forks 4 (left) and 0 (right) Parent detects deadlock, going to initiate recovery
Parent preempts Philosopher 1

Also for each step, print the ***allocation matrix*** and ***request matrix*** for each process.