IIT Bhubaneswar
School of Electrical Sciences
Operating Systems Lab (0-0-3)
Spring 2017
Lab Schedule: Thu (8:00-10:55)

Instructor: Debi Prosad Dogra (dpdogra@iitbbs.ac.in)
Teaching Assistant: Santosh K. K. (sk47@iitbbs.ac.in)

Laboratory 4 (**Submission Deadline: 01/03/2017 midnight**)

Points: 20 **(Only submission)**

Objectives: The object of this assignment is to gain experience on threads and process synchronization.

Practice during Laboratory Hours:
Understand the concept of pthreads, mutex, and semaphore and learn how to use them in programs. Get acquainted with various functions/APIs and systems calls related to these features in POSIX standard.

Take Home Assignment for Submission:

Hint for assignment: Use the below code structure to calculate time taken by various segments of a large algorithm. Assume the "**read input data**" reads the data from file, "**count the number of 3s in the input data**" does the main task (discussed later), and "**output result**" displays the output. Use man page to understand the system calls used in this pseudo-code.

```
#include <sys/time.h>
struct timeval startread, startcalc, finish, readtime, calctime, overalltime;
int main() {
        gettimeofday(&startread, NULL);
        //read input data
        gettimeofday(&startcalc, NULL);
        //count the number of 3s in the input data
        gettimeofday(&finish, NULL)
        timersub(&startcalc, &startread, &readtime);
        timersub(&finish, &startcalc, &calctime);
        timersub(&finish, &startread, &overalltime);
        //output result
}
```

We are interested in the time that elapses between the start and end of a computation. When you pass the struct timeval variable as argument to the gettimeofday() system call, the Linux kernel records the time elapsed since January 1, 1970, in this variable. Such a measurement is called a time-stamp. You take time-stamps before reading the input-data, after reading the input data, and at the end of counting. By taking two time-stamps and subtracting the later one from the earlier one, you can determine the time that has elapsed between the two time-stamps. The timersub() system call does just that: it takes two

timestamps (argument 1 and 2), and returns the difference in the third argument. You find all the details of the gettimeofday() system call in the corresponding man page. You can compute the execution times for reading the input data, counting, and reading + counting as given in the pseudo-code.
We assume that you cannot parallelize the time to read data from the file (sequential access), and that you can parallelize counting 3.
Problem: In this example, you need to count the occurrences of the number "3" in input data. Input data is provided in a text file according to a format, where the first line contains a single number that specifies the number of data items in the file. Every subsequent line contains one number (data item). You can assume that numbers are in the range 1 <= N <= 100.

Example:
4
2
3
33
6
The above file contains 4 data and there is one occurrence of "3" in the whole file. Remember "33" is not "3" appearing two times! Therefore, it must not be counted.

Write a concurrent program using pthreads and mutex locks to count the occurrences of number "3" in the given input file. Assume the user will provide number of threads as input through command line. Vary the number of threads from 1 to 16 and record time taken by various sections of the code as depicted in the pseudo-code. Using Amdahl's Law on the execution times for I/O and counting, plot the speedup and efficiency with respect to number of threads. Use **gnuplot** to create the graphs.
Another hint:
Reading of data from the file can be done first before you create the threads to count the number of times "3" is occurring. Just copy the data in an array! Then, apply your concurrent logic to process the array in parallel.

Submission Guidelines: Source files, Readme.txt, and graphs (pdf or png) should be zipped together in a file named "**RollNo_Lab4.zip**" and send it to the TA with the following subject line: "**OS_lab4_Assignment**".

Additional Reading: Read through **Chapter 3** in the **xv6** textbook. Start getting conversant with xv6 OS. Next assignment on threads and synchronization is going to be tough!