National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

(CL-220)

Course Instructor	Mr. Aamir Rahim
Lab Instructor(s)	Zumirrah Khalid Amir
Section	CS 4D
Semester	Spring 2021

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives:
In this lab, students will practice process synchronization using semaphores

Submission Instructions:

1. Submit your cpp file on google classroom named as your roll number (18_0000.cpp).
2. Do not zip your file.
3. **If you do not name your file as your roll number, submit a zipped file or submit a wrong file; then no marks will be given.**
4. We will check your code for plagiarism, so avoid it. Otherwise, there will be a heavy penalty.

Question 1: Barrier Problem

Implement a program in C/C++ which is passed as parameter an integer number n. The program will create n threads. There will be a variable x shared among all these threads. These threads will do the following:

1. Increment the value of x by 1.
2. After incrementing the value of x by 1, print the value of x.

Before creating n threads, the program will also create a special thread that will not do the above task. Instead, when the n threads have finished their execution, it will print ("All threads have finished their execution"). Make use of semaphores to synchronize the access to x. Also use a semaphore to prevent the special thread from printing the required statement until the n threads have finished. The values of x will be printed in order. Do not use semaphores in the main function; use them in the threads created.

Question 2: Calculating the maximum number from a list of integers by exploiting parallelism:

Suppose we have an array of integers: 1, 10, 9, 100, 23, 4, 11, 12, 3, and 1. The size of this array is 10. The maximum number in this list is 100. Now, if we have a large number of elements, then it will take a lot of time to find the maximum number. We can utilize parallelism by dividing the task of finding the maximum among different processes. Suppose we divide the task of finding the maximum from the above list to two processes. We pass the first half (1, 10, 9, 100, 23) of list to process p1 and the second half (4, 11, 12, 3, 1) of the list to process p2. P1 will return 100 as maximum and P2 will return 11 as maximum. No we can check which of the numbers returned from both processes is bigger which is 100 since 100 > 23. So 100 is the maximum number in the whole list.

Assume that we want to find the maximum number from a list of numbers. The numbers are not in a single file, but they are placed in several files. Now write a program that is passed the names of files containing integers as command line arguments. The parent process will create as many processes as the number of files. Each process will be passed the name of a file using a pipe. All processes will then work in parallel to find the maximum number from the files passed

to them as argument. Each process will return the maximum number from the file (that was passed to it to via pipe) to the parent process via a pipe. The parent process will then find the maximum number from the numbers returned by child processes, and prints the maximum on screen.

Suppose we have three files:

file1.txt	file2.txt	file3.txt
1, 10, 11, 54	54, 6, 7, 10	23, 100, 45, 9, 1, -5, 7, 75

1. P1 will be passed filename “file1.txt” by parent using a pipe. P1 will read file1.txt and return 54 to parent using a pipe.
2. P2 will be passed filename “file2.txt” by parent using a pipe. P2 will read file2.txt and return 54 to parent using a pipe.
3. P3 will be passed filename “file3.txt” by parent using a pipe. P3 will read file3.txt and return 100 to parent using a pipe.
4. Parent Process has received 54, 54, and 100 from P1, P2, and P3, respectively. Parent will now find and print the maximum from these numbers which is 100.