National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

(CL-220)

Course Instructor	Mr. Mubashar Hussain
Lab Instructor (s)	Haiqa Saman, Rasaal Ahmad
Section	BCS-4G & BCS-4F
Semester	Spring 2024

Department of Computer Science

FAST-NU, Lahore, Pakistan

Objectives:

- In this lab, students will practice thread and process synchronization using semaphores.

Lab Questions

Question 1:

There are exactly 3 threads that generate string a, b, and c in an arbitrary order. In the absence of any synchronization mechanism, there will be no order in the generation of a, b, and c. Synchronize threads using semaphore in such a way that your printed string will be aaacbaaacbaaacb……

//thread 1	//thread 2	//thread 3

While(1)
{
   cout << ‘a’;
}
	While(1)
	{
	      Cout << ‘b’;
	}
		While(1)
		{
		      Cout << ‘c’;
		}

Question 2:

Two Threads i.e., Producer and Consumer are sharing a buffer of size 100. The producer generates a random number from 1 to 6, stores that number in the buffer, and prints the number. The consumer must read values from the buffer added by the producer, perform the summation operation, and display the result. The producer will stop after adding 100 values. Both producer and consumer threads run simultaneously; hence, synchronize.

You’re not allowed to use cout/printf statements.

The output should look Like

```
Number 2      //printed by producer.
Number 5
Number 4
Number 2
Number 6
Number 2
Number 5
Number 1
Number 4
Number 2
Number 3
Number 4
Number 3
Number 5
Sum 2     //printed by consumer.
Sum 7
Number 2
Sum 11
Sum 13
```

Question 3:

Suppose there are two processes. Each process reads a different file having a list of integers. Both processes read the integers, calculate their sum, and send the sum and the count of integers to a server process via shared memory. The server then finds the total average by following formulae:

Total Sum= sum of integers sent by p1 + sum of integers sent by p2

Total Count= count sent by p1 + count sent by p2

Average= Total Sum/ Total Count

After calculating the average, the server sends the average to both processes. Both processes then print the sum on their respective terminal. (You need to synchronize the processes using semaphores on shared memory)

Shared Memory Portions Required:

1. The will be a single shared memory portion on which both processes will place their (sum, count) pair. One process will put the pair on the 0th index and the other will put the pair on the 1st index.
2. A shared memory portion for the currently available index number where a process can put the pair.