National University of Computer and Emerging Sciences

Laboratory Manual

for

Operating Systems Lab

Department of Computer Science

FAST-NU, Lahore, Pakistan

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

Question 1.

Write C++/C code for a program that takes as command line argument a source file name. The program acts as a producer and reads each time 20 character from file and writes the characters to a shared memory buffer created by the producer. The shared memory buffer can store at most 20 characters. Now there is another program which acts as a consumer, attaches the shared memory to its address space, and reads those 20 characters from shared memory, prints the characters on the screen, and waits for the user to press enter key. The producer can only write the next 20 character of file to shared memory after the consumer has read the previous 20 characters from the shared memory. So you need to synchronize the two processes using semaphores. You are not allowed to synchronize the processes any other way except using semaphores. When the file has been complete written to the shared memory, the producer writes a $ to shared memory which indicates to the consumer that file has been finished. After that all resources must be freed. (Assume the file's size is always multiple of 20, such as 20, 40, 60, etc.)

For example: Suppose there is a file of 40 characters.

- Producer writes 20 characters, and waits for the consumer to read these 20 characters.
- Consumer reads 20 characters and prints on the screen.
- Producer now writes the next 20 characters and waits for the consumer to read these 20 characters.
- The consumer reads the 20 characters and prints the data on the screen
- The end-of-file has been reached, so the producer writes $ to the shared memory. Then the producer detaches all shared memory portions from its address space and deletes the shared memory portions and exits.
- The consumer reads $, which indicates that the file's data has completely been read. So the consumer detaches all shared memory from its address space and exits.

Postlab (No need to submit it):

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

//child 1
while(1)
{
cout << 'a'<<endl;
}
	//child 2
	while(1)
	{
	cout << 'b'<<endl;
	}
		//child 3
		while(1)
		{
		cout << 'c'<<endl;
		}