CS-232: Operating Systems

Assignment 01

Duration: 1 Weeks (Please submit it by 11th of March before 1159 pm)

This assignment is a practice in inter-process communication and synchronization. You are supposed to implement the producer-consumer problem. You know the principle: *producer* and *consumer* are two processes (can be threads). The *producer* produces some information that is consumed (read) by the *consumer*. In this example we'll suppose that the *producer* will produce some integer values that it will send to the *consumer*. The *consumer* will receive and consume those values. The thing to take care of is that the values read by the *consumer* should exactly be the same as generated by the *producer* i.e. if the *producer* produced 100 values, the *consumer* should receive the same 100 values.

(You may face synchronization problems. Think about race conditions and using locks/semaphores to access critical sections.)

TO DO:

Part a:

You should create two different processes: *producer* and *consumer*. The *producer* should produce 1000 different integers (generate them randomly or read from a file) and communicate them to the *consumer*. The *consumer* should receive these values and store them in a pre-allocated array of integers. Once the *consumer* has received 1000 integers, it should open a file and write all those values in that file. It should then close the file and exit.

The *producer* should also store the values it generates in an array and after it has communicated all the values to the *consumer*, it should also write them in a file before exiting.

Once both programs have exited, you should compare the files generated by the *producer* and the *consumer* to see if they are exactly the same. The diff command will be of use here: *diff producer.txt consumer.txt*

In this part you should use Message Queues as the communication mechanism between the two processes. The size of the message queue should be 10 i.e. it can hold only a maximum of 10 messages at any given time.

Part b:

Implement the same *producer-consumer* mechanism as in part a but this time you are supposed to use Shared Memory as the communication mechanism between the processes. Again the size of the shared memory should be such that it be able to hold a maximum of 10 integers at any given time.

Part c:

You implement the same producer-consumer problem as in part a and b, but this time *producer* and *consumer* are two threads, not two processes as in the previous cases.

We'll let you figure out how to communicate between the producer and consumer threads.

The only restriction be that whatever communication mechanism you use, it should be able to hold only a maximum of 10 integers at any given moment.

We've included a toy producer/consumer program using code from Silberschatz 9th edition Section 6.1 for shared memory between processes to get you started.

IMPORTANT: Get started at the earliest so that you may know the potential problems earlier !!!

Submission guidelines:

1. You should submit a single tar-zipped file whose name should be your ID.
2. When extracted it should extract to a single directory whose name should be your ID.
3. In this extracted directory, you should have three sub directories named part_a, part_b, part_c.
4. The sub directories should contain only files for the source code of each part and a makefile to build them.
5. Write clean code, comment it, follow good coding practices like, giving sensible names to variables, de-allocate all resources after their use, etc.

Make sure that you DO NOT create any memory leaks or dangling pointers.