CSC139: Operating System Principles
Fall 2023
First Assignment
Multiprocessing and Shared Memory
Due Wednesday, February 22, 2023

In this assignment, you will write producer and consumer programs that communicate through a bounded buffer in shared memory. The code that you will need to write will be based on the code given in class for forking a child process, creating a shared memory block and implementing a bounded buffer accessed by a producer and a consumer. Write your code in the given producer_template.c and consumer_template.c source files, and then rename them to producer.c and consumer.c. Follow the instructions provided in these files. Note that the low-level functions for reading from and writing to the shared memory block are already provided to you. Please use the provided functions; do not write your own implementation of an already provided function.

As you can see in the given producer_template.c source file, the producer process is the parent process that creates (forks) a child process and loads the consumer executable into it. The consumer executable will be generated by compiling consumer.c (after you write the required code in it). After writing all the required code in both consumer.c and producer.c, use the following commands to compile the two programs and build the two executables:
gcc producer.c -lrt -o producer
gcc consumer.c -lrt -o consumer

The producer program (parent) takes the following three command-line arguments (all integers):

1. The size of the bounded buffer (bufSize). This is the number of items that can fit in the bounded buffer. This cannot exceed **512**. Your program should check if the value of this-command line argument is within the valid range (**2** to **512**); if not, it should print an appropriate error message and terminate.
2. The number of items to produce/consume (itemCnt). Your program should work correctly for any itemCnt that is greater than 0. However, in an interesting test case, itemCnt will be much larger than bufSize. **If your program works only if itemCnt is less than bufSize, you won't get any credit, as you won't be solving the bounded-buffer problem.**
3. A seed for the random number generator (randSeed). This will allow us to test the program using different data.

The producer program creates a shared memory block with a fixed size of **4K**. Let the name of the shared memory block be OS_HW1_yourName, where "yourName" should be your actual full name. The purpose of this is to avoid interfering with other students' work if you happen to be running your programs on the same shared server. The producer then uses this block to communicate with the consumer. The block consists of a header followed by the bounded buffer with the given bufSize. The header contains the following four integers in order:

1. bufSize
2. itemCnt
3. in: the index of the next item to be produced
4. out: the index of the next item to be consumed

The main in the producer program first calls the function InitShm() to create the shared memory block and initialize its header with the above four integer values. It then forks a child process and loads the consumer executable into it. The parent process calls the function Producer() to actually generate the items and write them into the bounded buffer. The producer must generate itemCnt random integers in the range **10** to **2500**. Simply call the provided GetRand() function to do that. **The key point is that when the bounded buffer is full, the producer must wait until the consumer has consumed at least one item**.

The consumer program, in turn, performs the following main steps:

1. Opening and mapping the shared memory block created by the producer. The shared memory block name must match the name of the block created by the producer: OS_HW1_yourName.
2. Reading the four integer values from the header.
3. Reading all the items written by the producer to the shared memory buffer (their count is itemCnt). The key point is that when the bounded buffer is empty, the consumer must wait until the producer has produced at least one more item.

The producer/consumer code that you will need to write will be very similar to the bounded-buffer producer/consumer pseudo-code given in class, but make sure that you do the following:

1. Whenever you read any of the shared variables (in and out), you must read the latest value from the shared buffer using the provided GetIn() and GetOut() functions.
2. Whenever you update any of the shared variables (in and out), you must update their values in the shared buffer using the provided SetIn() and SetOut() functions.
3. Reading from and writing into the shared buffer should be done using the provided ReadAtBufIndex() and WriteAtBufIndex() functions.
4. Use the provided GetRand() function to generate a random number in the above-specified range.

Submission

Submit your source files on Canvas in the Assignment 1 item. **Your code must compile and work correctly in our ECS computing system**. Name your files producer.c and consumer.c; don't use any other names. In each file, please include a header that has the following information:

Your name

Your section number

The OSs on which you have tested your program (Linux, Mac, etc). These must include Linux and your code must work correctly in our ECS computing environment.