Programming Project 1

1. Introduction

In this project, you will write a program to simulate producer-consumer problem using bounded buffer whose size is 10. (this means the buffer only can store 10 messages.) For this, there will be one program creating two threads (one for producer and the other for consumer) which act as different processes. After 15 seconds of running, the program (both producer and consumer threads) should be terminated gracefully without any error.

2. Producer Thread

Producer thread generates integers (0~14) which are treated as a message. The thread first checks if the buffer is full or not. At this moment, make sure the producer thread always waits random period of time (10~100 milliseconds) before checking the buffer. This is to create randomness in the program execution. If the buffer is full, it waits another random period of time (10~100 milliseconds). If the buffer is not full, then it places an integer (0~14) in sequence into the buffer. "In sequence", herein, means first the producer places 0 in the buffer. Next time, it produces 1, 2, 3, … and so on until it produces 14 and places it in the buffer. Once the producer places 14 in the buffer, it produces 0 again.

Every time producer thread places a message (and integer) in the buffer, it makes a log in the file (say, producer.txt) in the following format (where current time is in millisecond):

Producer
"current time", Placing 0 in the buffer location 0.
"current time", Placing 1 in the buffer location 1.
"current time", Placing 2 in the buffer location 2.
…
"current time", Placing 9 in the buffer location 9.
"current time", Placing 10 in the buffer location 0.
"current time", Placing 11 in the buffer location 1.
…
"current time", Placing 14 in the buffer location 4.
"current time", Placing 0 in the buffer location 5.
"current time", Placing 1 in the buffer location 6.
"current time", Placing 2 in the buffer location 7.
…

3. Consumer Thread

Consumer thread consumes integers (messages), 0~14, placed in the buffer by the producer thread. The consumer thread first checks if the buffer is empty or not. At this moment, make sure the consumer thread always waits random period of time (10~100 milliseconds) before checking the buffer. This is to create randomness in the program execution. If the buffer is empty, it waits another random period of time (10~100 milliseconds). If the buffer is not empty, then it takes an integer (message), which is placed in the buffer at the earliest time, from the buffer. (based on the FIFO policy)

Every time consumer thread consumes a message (and integer) from the buffer, it makes a log in the file (say, consumer.txt) in the following format (where current time is in millisecond):

Consumer
"current time", Consuming 0 from the buffer location 0.
"current time", Consuming 1 from the buffer location 1.
"current time", Consuming 2 from the buffer location 2.
…
"current time", Consuming 9 from the buffer location 9.
"current time", Consuming 10 from the buffer location 0.
"current time", Consuming 11 from the buffer location 1.
…
"current time", Consuming 14 from the buffer location 4.
"current time", Consuming 0 from the buffer location 5.
"current time", Consuming 1 from the buffer location 6.
"current time", Consuming 2 from the buffer location 7.
…

4. Report

You are to provide a report named "report.txt" that should include:

a. Problem definition and proposed solution (how your program is implemented).
b. What are the results you got (in plain English, discuss the output of your system, do not copy and paste the program output)
c. All the bugs or problems known, any missing items and limitations of your implementation IF ANY. (honesty deserves additional points)
d. Any additional sections you see necessary

Please note that your reports (your program as well) MUST consist of your own sentences, if you have to copy anything from anywhere you have to quote it and provide reference. Also a perfect program does not necessarily deserve full points if it is not complemented with a good report. A good report is a brief one that helps its reader understand the system thoroughly from the problem definition through the limitations.

5. Submission Guidelines

a. Only submit source code files. (NO object or project files)
b. Please compress your files to ".zip" format.
c. Include your 1~3 page(s) project report.
d. You must submit the zip file after logging into your blackboard account.

6. Grading Rubric

a. Correct Implementation/Execution/Graceful Termination 75%
b. Report 20%
c. Stylish Code/Readability/Comments/Structure and Readability of Output 5%