Programming Assignment 3

**CSC 4320/6320 - Operating Systems**

In the case of program is completed by Python, please make corresponding adjustments to the content related to the C language (such as file suffix, compile code)

**Problem Statement:**

Read through **Project 3 (page 253-255)** in the textbook, the **Producer–Consumer Problem,** and finish it using Pthreads only (ignore the Windows API part). Your project must meet the following requirements for full credit:

1. Instead of generating random numbers and write to the buffer, producer threads will write incremental integers into the buffer. For example, if the initial value of the number is 10, then the next number should be 11, and the next will be 12....
2. Each producer and consumer thread has an index that are passed as a parameter when the thread is created. For both producer and consumer threads, the index starts at 0, and increases by 1 for each new thread.
3. When a producer (consumer) thread finishes producing (consuming) a number, it needs to print a message, e.g., “Producer xhu\_P0 produced 10” or “Consumer xhu\_C2 consumed 10”, where xhu\_P0 and xhu\_C2 are the “name” of the producer and consumer. The “name” of a producer has the following format: #firstInitial#lastname\_P#index, where #firstInitial is the initial of your first name, #lastname is your last name, #index is the index of the thread. For example, xhu\_P0 is the name for the producer thread that has index 0 (based on my name Xiaolin Hu). The “name” of a consumer has the following format: #firstInitial#lastname\_C#index. See sample screenshots below as examples.
4. To ensure that the messages are printed in the correct order, e.g., a number will be first produced before it is consumed, it is necessary to print the messages within the critical sections of the producer and consumer threads. Note: this is different from the code outline shown in Figure 5.26, where the messages are printed outside the *insert\_item( )* and *remove\_item( )*.
5. Create producer and consumer threads with 3 scenarios and show your results in screenshots similar as the sample screenshots shown below. The three scenarios are: 1) more producers than consumers; 2) more consumers than producers; 3) same number of producers and consumers. **Please refer to the screenshots shown below for sample outputs under different scenarios. (Your outputs may be different)**

The main () function will be passed **four** parameters on the command line. The four parameters are:

1. How long will the main thread sleep before terminating
2. The number of producer threads
3. The number of consumer threads
4. The initial value of the number (for producers)

Your program should be executed like this:

./*buffer <sleep time> <producer threads> <consumer threads> <start number>.* Here *buffer* is your executable file which is followed by four parameters, *sleep time, the number of producer threads, the number of consumer threads, and the initial number*. Eg. ./buffer 10 2 2 10

**Some Notes:**

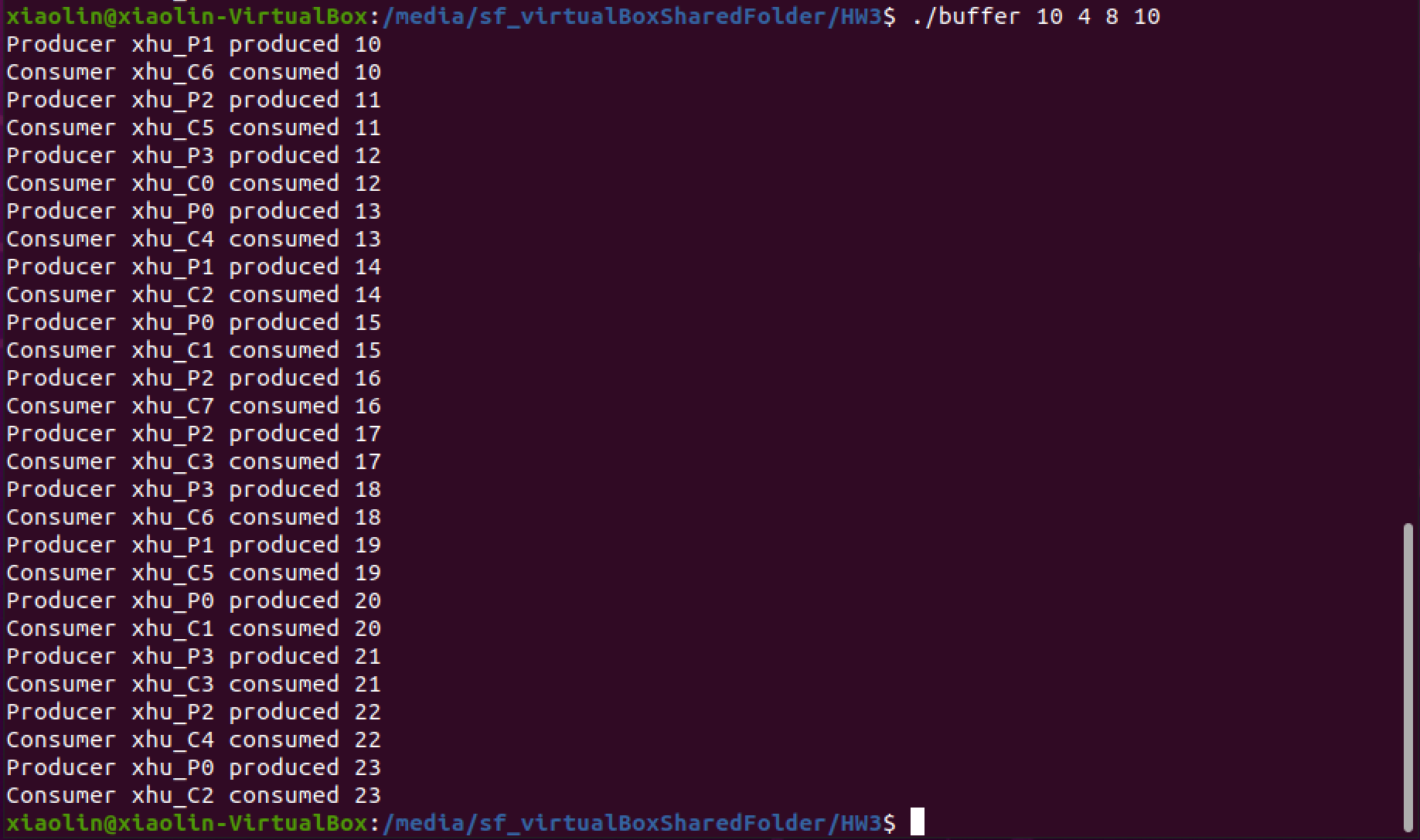
1. To help you to get started, a skeleton of the buffer.c is provided in iCollege. Feel free to use or modify this skeleton based on your needs.
2. Using **gcc –pthread –o buffer buffer.c** to compile, and then an executable file *buffer* would be generated. (Suppose buffer.c is your code file)
3. In your report, you need to include screenshots of the three scenarios (see sample screenshots below).
4. Work individually. Programs should exhibit a modular or object-oriented design. Poor design will not earn full credit.

**What to submit (submit trough iCollege)**

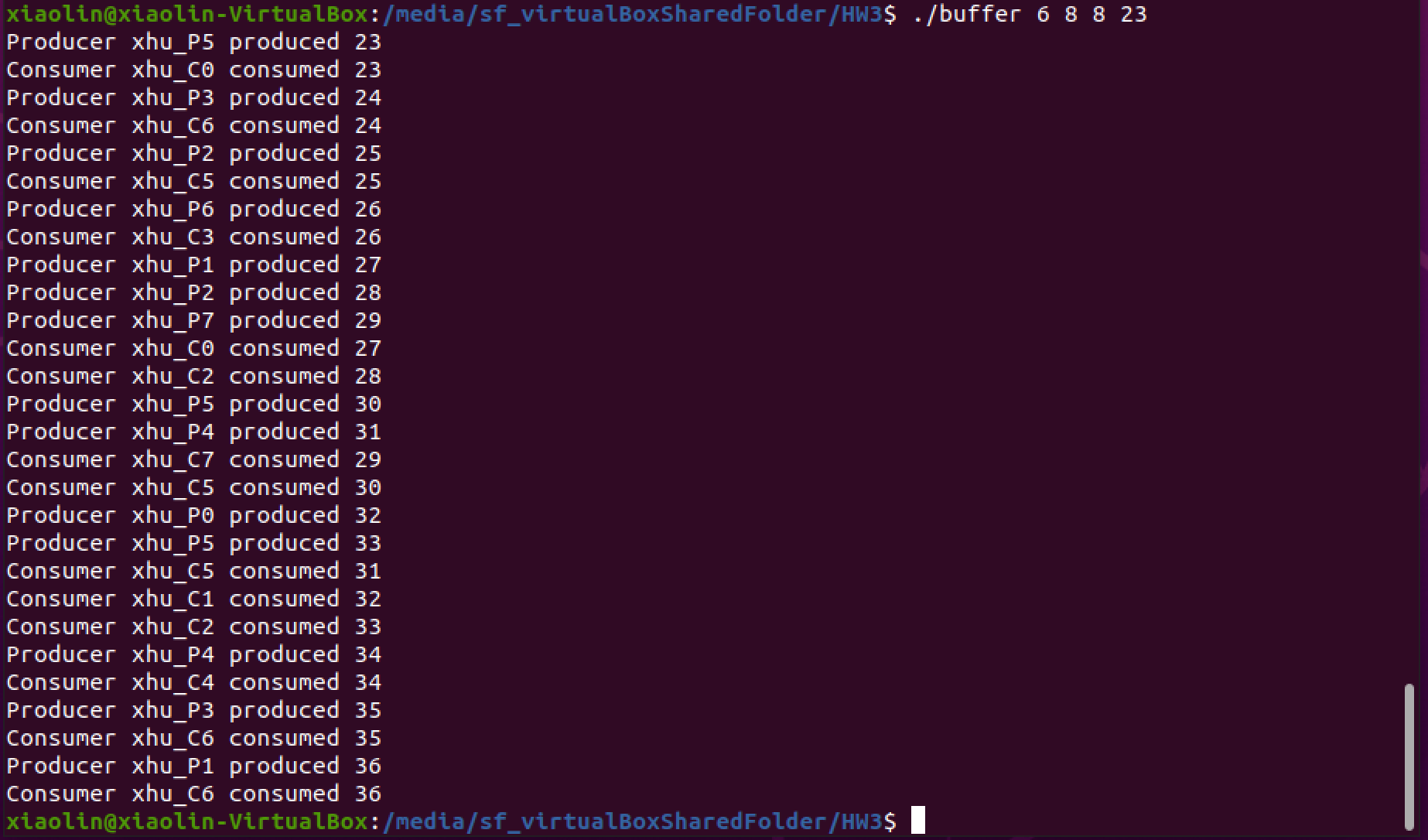
1. Upload **a source code file** of *buffer.c* and **a project report** (named as “*HW3.pdf*” or “*HW3.doc*”) to iCollege. The project report should include: 1) the source code;2) a screenshot of the output.
2. Submit your files separately; do NOT submit a zip file.

**Sample screenshots:**

**Scenario 1:** **number of producers is less than number of consumers. In this scenario, a produced number will be immediately consumed. In other words, the buffer is empty most of the time.**



**Scenario 2:** **Number of producers is equal to number of consumers. In this scenario, the produced numbers and consumed numbers are more or less balanced.**



**Scenario 3:** **Number of producers is larger than number of consumers. In this scenario, the producers generate numbers faster than being consumed. As a result, the buffer is full most of the time.**

