Introduction

In this assignment you will be solving the classic bounded buffer producer-consumer problem with one producer and multiple consumer threads. You have been provided with the following files:

- `buffer.c` - The file where you will be writing your code
- `buffer.h` - A header file with some useful definitions
- `buffer_driver.c` - The driver file that sets up the multithreading simulation
- `shortlist` - a short trace
- `longlist` - a longer trace
- `Makefile` - A makefile to compile your program

The program takes the number of consumers as argument (defaulting to 1) and a sequence of numbers from stdin. More information on the workings can be found in the form of comments on top of the `buffer_driver.c` file.

The producer thread reads the sequence of numbers and feeds that to the consumer. Consumers pick up the number, do some ”work” with the number, and then go back for another number.

Requirements

You are required to follow the guidelines below in order for your solution to be considered correct.

- You are responsible for making a synchronized buffer that can hold TEN integers. This buffer should follow a first in first out order.
- The buffer must be backed by a linked list.
- Your implementation must not spin-wait. There are several possible strategies. An excellent strategy with pthreads is to use a mutex and condition variables, i.e. use `pthread_cond_wait()` to wait when the buffer is empty or full.
- No two consumers can be processing the same integer at a given time. If a consumer extracts a number from the buffer that another consumer is currently processing, it must wait (not spin-wait!) until that consumer is done. Think about what additional data structures and locks you might use to do this.

NOTE: If you are running Linux in a virtual machine, make sure to enable multiple cores. If you do not enable multiple cores, running your code on our machine may produce deadlocks that won’t show up on your computer.

Implementation

As you may have seen, the given `buffer.c` file has some incomplete functions. For this project, you are asked to complete the following functions.

- `void buffer_init(void)` - This function is called by `main()` in `buffer_driver.c` before the producer and consumer threads are started. Use this function to initialize your buffer, mutex, condition variables, etc.
- `void buffer_insert(int number)` - This function, called by the producer thread inserts a number into the next available slot in the buffer. If no slots are empty, the thread should wait (not spin-wait!) for an empty slot to become available.
  - **Note:** You may assume that numbers normally passed into this function are positive numbers in the range 1-10 (inclusive).
  - **Note:** The exception to the above rule is when the producer is signaling the consumer to ”shut down”, which is done by inserting 0 into the buffer.
- `int buffer_extract(void)` - This function, called by the consumer threads removes and returns the number from the next available slot. If no numbers are available, the thread should wait (not spin-wait!) for a number to become available.
  - **Note:** Multiple consumer threads may call buffer extract simultaneously.

Testing

Compile your code by typing `make`. You can run it by doing the following

```
$ ./bounded_buffer [Number of Consumers] < [Input Filename]
```

Note: The '<' is used to pipe the contents of the file as input to the program.

- Try measuring the execution time using the `/bin/time` command in Linux. When running with `longlist`, doubling the number of consumers should roughly halve the execution time. What is the minimum possible execution time?

Deliverables

Use `make submit` to generate a tar.gz file, which you should submit via T-Square. Please download your submission once you have submitted it to make sure that it compiles. **Any submission that does not compile will get 0 credit.**