CSCI 340 Operating Systems Project

By: Shefali Emmanuel Date: December 7, 2019

I compiled the program by stating 'gcc filename.c -o filename -lpthread' followed by './filename'. I found many resources on how to link the pthread library with the program, but I preferred this solution as it was the shortest. I have written the commands for each part below their headers.

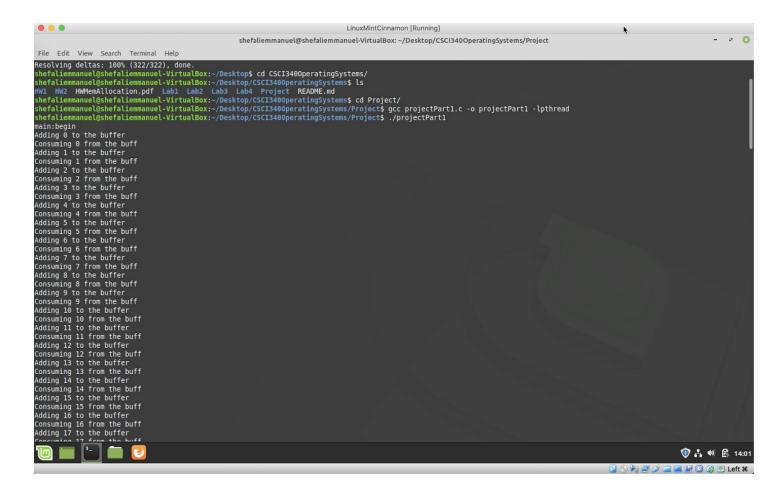
Here is a link to my main reference:

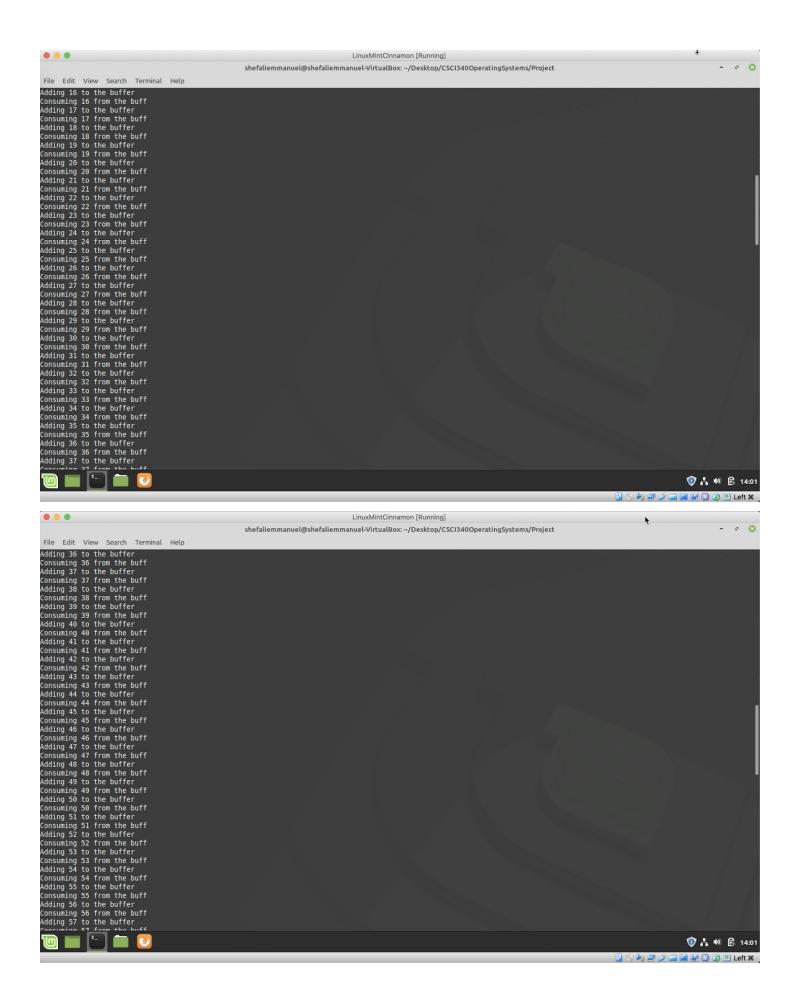
https://www.includehelp.com/c-programming-questions/compiling-program-with-pthread-library-linux.aspx

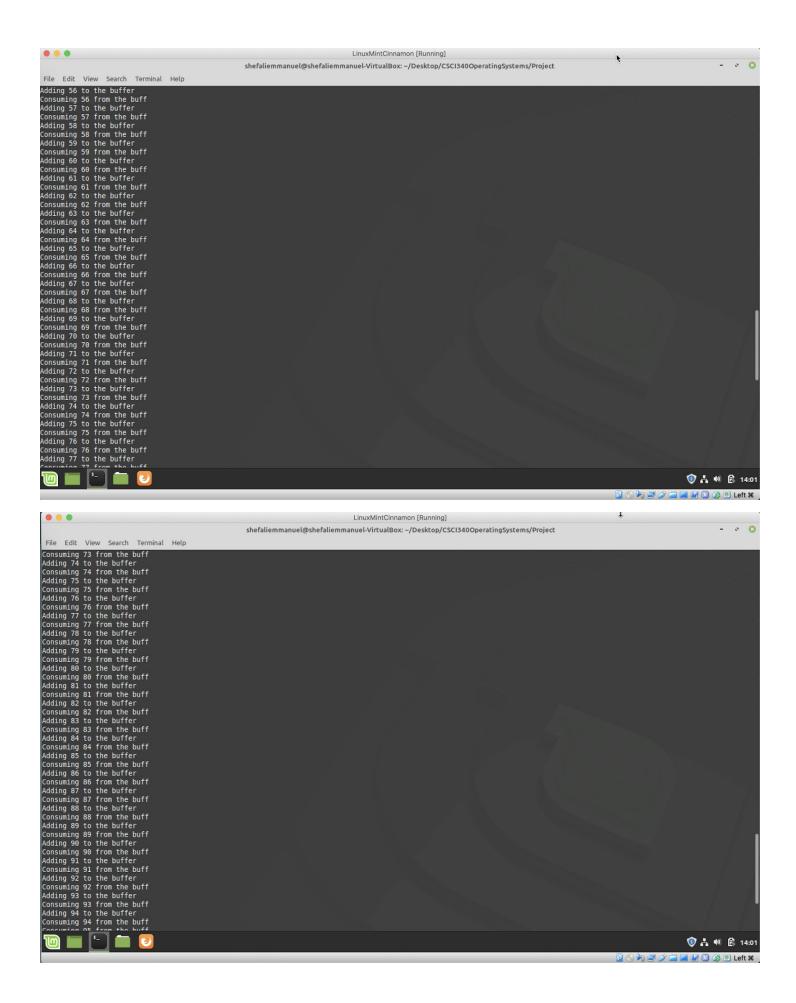
I also referenced various parts of the book and github.

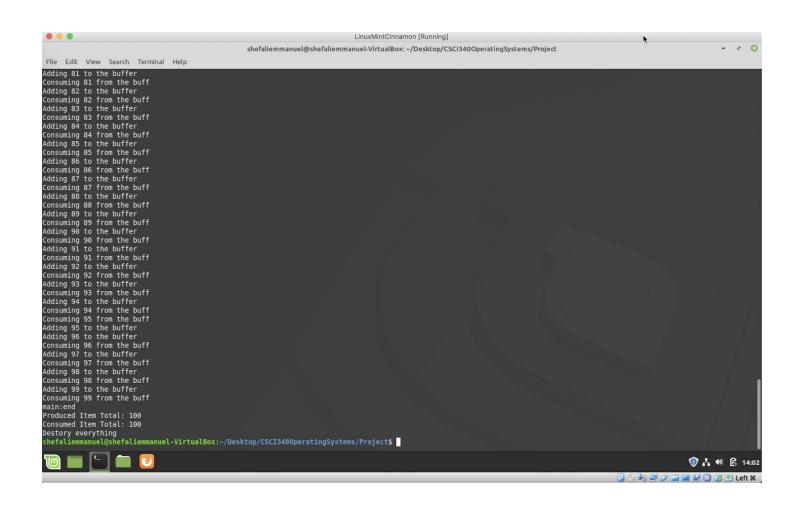
Part 1:

gcc projectPart1.c -o projectPart1 -lpthread ./projectPart1



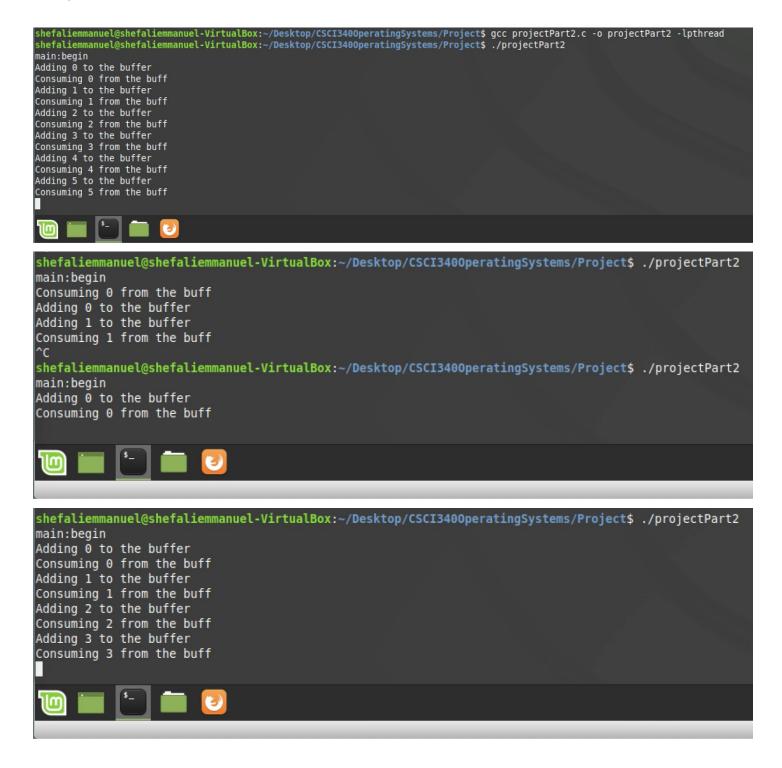






Part 2:

gcc projectPart2.c -o projectPart2 -lpthread ./projectPart2

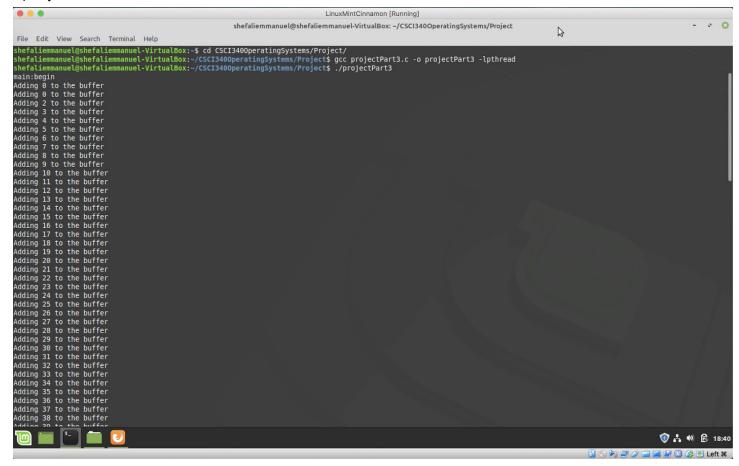


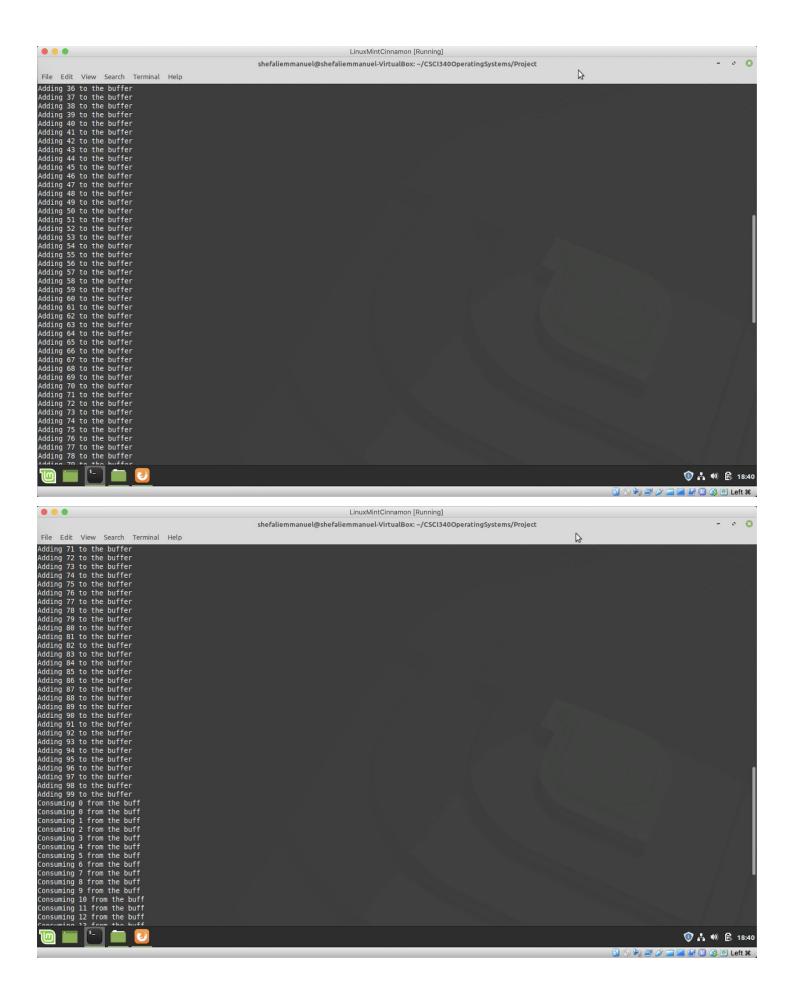
```
shefaliemmanuel@shefaliemmanuel-VirtualBox:~/Desktop/CSCI340OperatingSystems/Project$ ./projectPart2
main:begin
^C
shefaliemmanuel@shefaliemmanuel-VirtualBox:~/Desktop/CSCI340OperatingSystems/Project$ ./projectPart2
main:begin
Adding 0 to the buffer
Consuming 0 from the buff
Adding 1 to the buffer
Consuming 1 from the buff
Adding 2 to the buffer
Consuming 2 from the buff
Adding 3 to the buffer
Consuming 3 from the buff
Adding 4 to the buffer
Consuming 4 from the buff
Adding 5 to the buffer
Consuming 5 from the buff
Adding 6 to the buffer
Consuming 6 from the buff
Adding 7 to the buffer
Consuming 7 from the buff
Adding 8 to the buffer
Consuming 8 from the buff
Adding 9 to the buffer
Consuming 9 from the buff
Adding 10 to the buffer
Consuming 10 from the buff
Adding 11 to the buffer
Consuming 11 from the buff
```

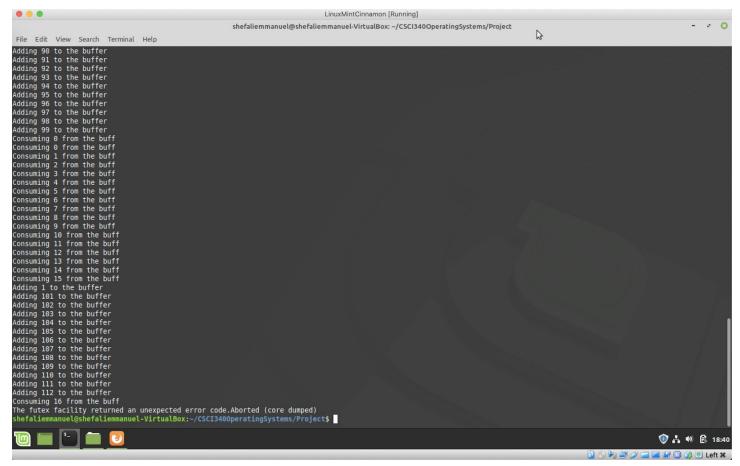
After running the program multiple times, it was clear to me that the output was not the same. This version 2 of the project is indeterministic. This version will produce a deadlock. A producer can accept a mutex, but then get blocked when all buffer slots are being utilized. Since it is holding the mutex, the consumer can not rid itself of the buffer. I did not expect the program to reach a different breaking point upon each execution. Sometimes it would not even start the first thread creation... other times it would create multiple!

Part 3:

gcc projectPart3.c -o projectPart3 -lpthread ./projectPart3







These results are indeterministic due to a race condition that is going on.

I created 2 procedures followed by a consumer. In the race condition mentioned above, the first procedure runs to its fullest extent. Next the consumer gets allocated some memory which is quickly taken away by the start of the second producer who starts promptly after the consumer has run for a little bit. Each time I ran this executable, the consumer ends at a different number.