 # Name: Nicholas Fotinakes

 # Date: 2022-04-12

 # Title: Lab 6 - Lab Report: Notes, Questions, & Test Results

 # Description: This pdf contains notes, answers, and test results for Lab 6

Questions

Explain what happens when you run the threadSync.c program?

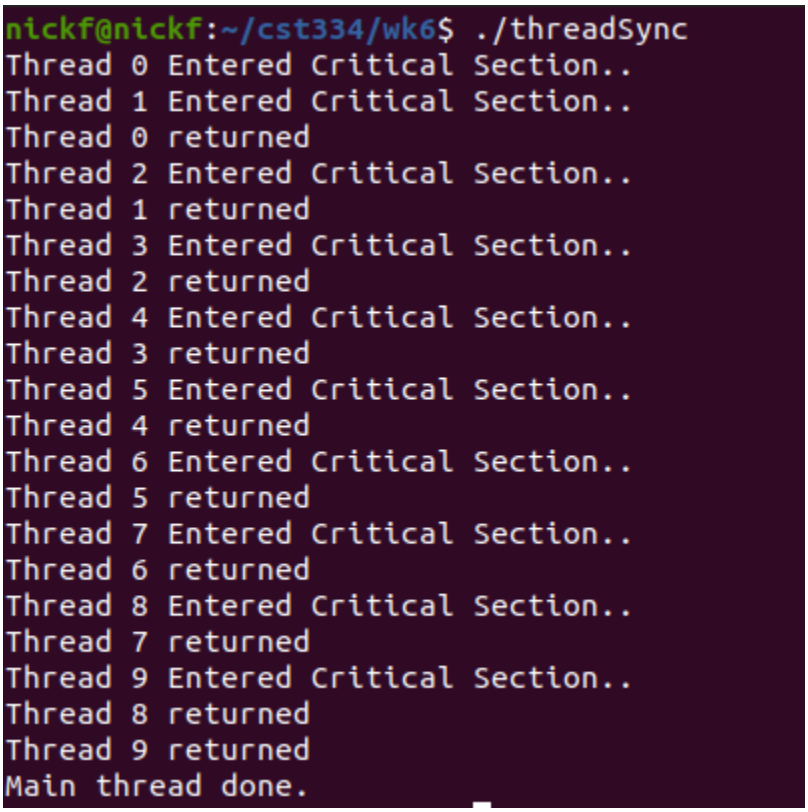
The threadSync.c program makes use of a binary semaphore called mutex to act as lock to protect a critical section of code and provide thread synchronization. The program uses a for loop to create 10 threads as we have seen before in previous labs. When each thread is created, it calls the go function. The go function uses the binary semaphore to protect the printf statement and sleep call, only allowing one thread to execute this at a time. When the first thread reaches the semaphore, it will decrement the initial set value of 1 to 0, meaning the next thread created and calling go will now wait until the sem_post is called to raise the semaphore value back to 1. This provides synchronization of threads so that each thread waits it's turn to execute the critical section of code within the go function before returning and printing the return statement and joining. NOTE: I found using %lu in the print statement for size_t can throw a warning for some systems. It may be safer to use %zu but I left the code as is because the prompt did not mention changing anything.

Notes

For Step 2 to solve the Producer/Consumer problem my program alphabet.c uses a binary semaphore similar to threadSync.c to provide thread synchronization in addition with the use of two signaling semaphores to control a bounded buffer, to keep track of when a buffer of n size is full or empty. By using these semaphores, a producer thread and consumer thread can synchronize and alternate back and forth, producing alphabet letters and consuming them based on the status of the buffer. We initially set the full semaphore to 0 and empty semaphore to the size of the buffer, so when the producer and consumer threads are created, the producer will see the buffer is empty and start producing while the consumer thread sees that the buffer is not full and will wait until the producer thread fills the buffer. The mutex semaphore provides a lock when each thread needs to perform the critical section of code of putting or getting a letter in the buffer. The use of semaphores allows this "handshake" or synchronization of threads to solve the problem. In my implementation I decided to use for loops in my producer/consumer set to loop the size of the alphabet (26) and to simply increment a char starting at 'A' to easily produce each letter. NOTE: Professor Lamble informed me that as long as our semaphores are placed correctly, using printf is okay even though printf is not technically thread safe.

Test Results:

threadSync.c:



Step 2 - Buffer Size 1:

```
nickf@nickf:~/cst334/wk6$ ./alphabet
Buffer size: 1   Buffer status: *
Producer letter: A       BUFFER STATUS: A
Consumer letter: A       BUFFER STATUS: *
Producer letter: B       BUFFER STATUS: B
Consumer letter: B       BUFFER STATUS: *
Producer letter: C       BUFFER STATUS: C
Consumer letter: C       BUFFER STATUS: *
Producer letter: D       BUFFER STATUS: D
Consumer letter: D       BUFFER STATUS: *
Producer letter: E       BUFFER STATUS: E
Consumer letter: E       BUFFER STATUS: *
Producer letter: F       BUFFER STATUS: F
Consumer letter: F       BUFFER STATUS: *
Producer letter: G       BUFFER STATUS: G
Consumer letter: G       BUFFER STATUS: *
Producer letter: H       BUFFER STATUS: H
Consumer letter: H       BUFFER STATUS: *
Producer letter: I       BUFFER STATUS: I
Consumer letter: I       BUFFER STATUS: *
Producer letter: J       BUFFER STATUS: J
Consumer letter: J       BUFFER STATUS: *
Producer letter: K       BUFFER STATUS: K
Consumer letter: K       BUFFER STATUS: *
Producer letter: L       BUFFER STATUS: L
Consumer letter: L       BUFFER STATUS: *
Producer letter: M       BUFFER STATUS: M
Consumer letter: M       BUFFER STATUS: *
Producer letter: N       BUFFER STATUS: N
Consumer letter: N       BUFFER STATUS: *
Producer letter: O       BUFFER STATUS: O
Consumer letter: O       BUFFER STATUS: *
Producer letter: P       BUFFER STATUS: P
Consumer letter: P       BUFFER STATUS: *
Producer letter: Q       BUFFER STATUS: Q
Consumer letter: Q       BUFFER STATUS: *
Producer letter: R       BUFFER STATUS: R
Consumer letter: R       BUFFER STATUS: *
Producer letter: S       BUFFER STATUS: S
Consumer letter: S       BUFFER STATUS: *
Producer letter: T       BUFFER STATUS: T
Consumer letter: T       BUFFER STATUS: *
Producer letter: U       BUFFER STATUS: U
Consumer letter: U       BUFFER STATUS: *
Producer letter: V       BUFFER STATUS: V
Consumer letter: V       BUFFER STATUS: *
Producer letter: W       BUFFER STATUS: W
Consumer letter: W       BUFFER STATUS: *
Producer letter: X       BUFFER STATUS: X
Consumer letter: X       BUFFER STATUS: *
Producer letter: Y       BUFFER STATUS: Y
Consumer letter: Y       BUFFER STATUS: *
Producer letter: Z       BUFFER STATUS: Z
Consumer letter: Z       BUFFER STATUS: *
Main thread done.
```

Buffer Size 8:

```
nickf@nickf:~/cst334/wk6$ ./alphabet
Buffer size: 8   Buffer status: ********
Producer letter: A       BUFFER STATUS: A*******
Producer letter: B       BUFFER STATUS: AB******
Producer letter: C       BUFFER STATUS: ABC*****
Producer letter: D       BUFFER STATUS: ABCD****
Producer letter: E       BUFFER STATUS: ABCDE***
Producer letter: F       BUFFER STATUS: ABCDEF**
Producer letter: G       BUFFER STATUS: ABCDEFG*
Producer letter: H       BUFFER STATUS: ABCDEFGH
Consumer letter: A       BUFFER STATUS: *BCDEFGH
Consumer letter: B       BUFFER STATUS: **CDEFGH
Consumer letter: C       BUFFER STATUS: ***DEFGH
Consumer letter: D       BUFFER STATUS: ****EFGH
Consumer letter: E       BUFFER STATUS: *****FGH
Consumer letter: F       BUFFER STATUS: ******GH
Consumer letter: G       BUFFER STATUS: *******H
Consumer letter: H       BUFFER STATUS: ********
Producer letter: I       BUFFER STATUS: I*******
Producer letter: J       BUFFER STATUS: IJ******
Producer letter: K       BUFFER STATUS: IJK*****
Producer letter: L       BUFFER STATUS: IJKL****
Producer letter: M       BUFFER STATUS: IJKLM***
Producer letter: N       BUFFER STATUS: IJKLMN**
Producer letter: O       BUFFER STATUS: IJKLMNO*
Producer letter: P       BUFFER STATUS: IJKLMNOP
Consumer letter: I       BUFFER STATUS: *JKLMNOP
Consumer letter: J       BUFFER STATUS: **KLMNOP
Consumer letter: K       BUFFER STATUS: ***LMNOP
Consumer letter: L       BUFFER STATUS: ****MNOP
Consumer letter: M       BUFFER STATUS: *****NOP
Consumer letter: N       BUFFER STATUS: ******OP
Consumer letter: O       BUFFER STATUS: *******P
Consumer letter: P       BUFFER STATUS: ********
Producer letter: Q       BUFFER STATUS: Q*******
Producer letter: R       BUFFER STATUS: QR******
Producer letter: S       BUFFER STATUS: QRS*****
Producer letter: T       BUFFER STATUS: QRST****
Producer letter: U       BUFFER STATUS: QRSTU***
Producer letter: V       BUFFER STATUS: QRSTUV**
Producer letter: W       BUFFER STATUS: QRSTUVW*
Producer letter: X       BUFFER STATUS: QRSTUVWX
Consumer letter: Q       BUFFER STATUS: *RSTUVWX
Consumer letter: R       BUFFER STATUS: **STUVWX
Consumer letter: S       BUFFER STATUS: ***TUVWX
Consumer letter: T       BUFFER STATUS: ****UVWX
Consumer letter: U       BUFFER STATUS: *****VWX
Consumer letter: V       BUFFER STATUS: ******WX
Consumer letter: W       BUFFER STATUS: *******X
Consumer letter: X       BUFFER STATUS: ********
Producer letter: Y       BUFFER STATUS: Y*******
Producer letter: Z       BUFFER STATUS: YZ******
Consumer letter: Y       BUFFER STATUS: *Z******
Consumer letter: Z       BUFFER STATUS: ********
Main thread done.
```