CSE320 Assignment7 Review Document

2 May, 2024
113949335
Hojun Kwak

The utmost goal for the assignment was to understand the role of semaphores and how to effectively utilize the feature to control memory while multithreading. Since multithreading shares the same resources(memory), it is crucial to ensure that the shared resource is accessed by one thread at a time. For part a, I inserted the print statement on each for loop for both customer and producer thread so that I could track the memory access for both threads. For part b, I changed the actions done by the reader in the given code to be done by the writer. This way, I was able to resolve the reader-writer problem in a writer favorable way. Writer writes once it is ready to write and the reader waits for the writer even if it is waiting.

Things that I've learned:

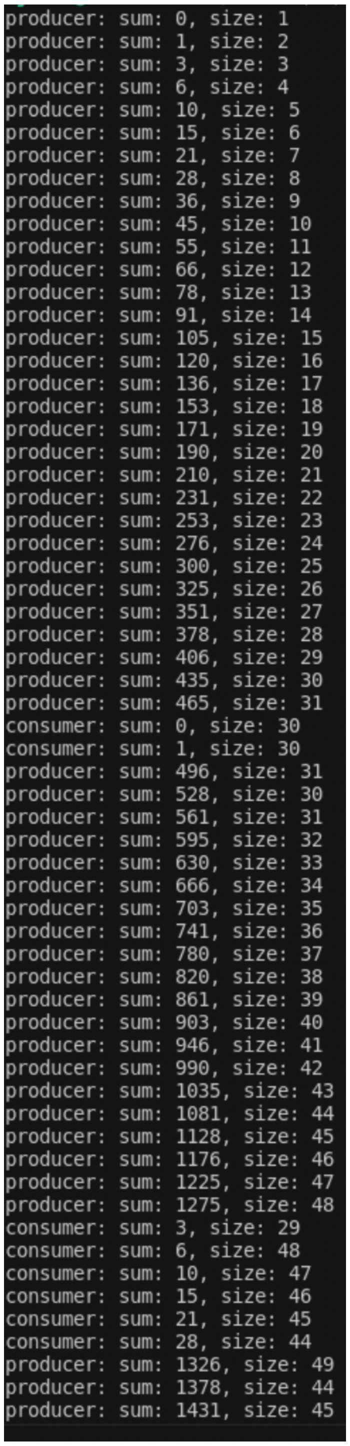
- Mutual exclusion is useful in virtual situations such as producer and consumer where the memory is treated like products and buffers are like stores
- There is a feature called sem_wait, that waits for the other thread to finish accessing the memory
- Sem_post releases memory for other threads to access

Some troubles I've faces:

- Understanding the interconnectivity between consumer and producer was not easy at first
- It seemed strange that the reader kept printing the same result for multiple times, but this came clear as I understood that the reader's role was to print out what has been written so far by the writer

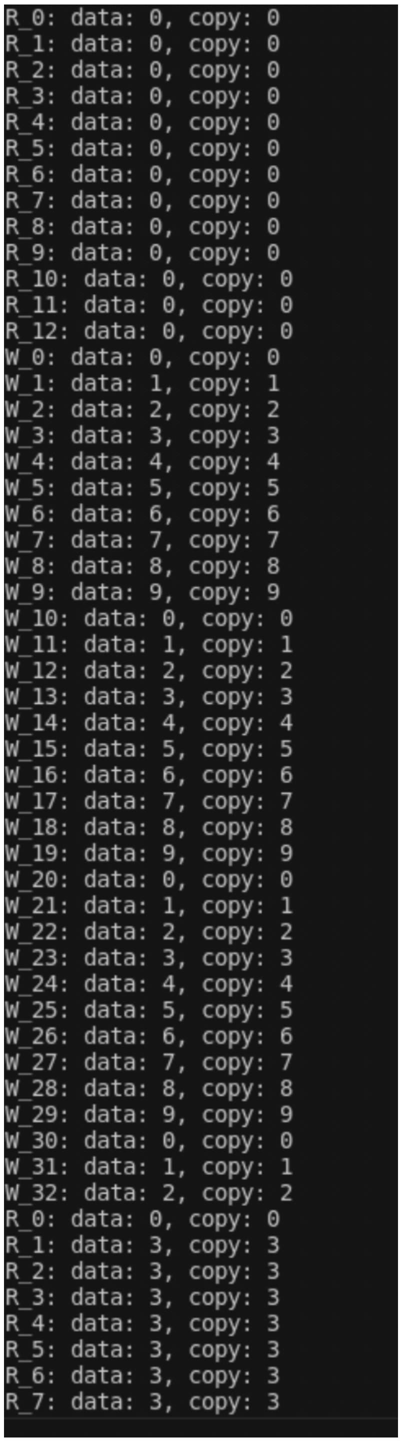
Code execution results:

Part 1:



```
producer: sum: 0, size: 1
producer: sum: 1, size: 2
producer: sum: 3, size: 3
producer: sum: 6, size: 4
producer: sum: 10, size: 5
producer: sum: 15, size: 6
producer: sum: 21, size: 7
producer: sum: 28, size: 8
producer: sum: 36, size: 9
producer: sum: 45, size: 10
producer: sum: 55, size: 11
producer: sum: 66, size: 12
producer: sum: 78, size: 13
producer: sum: 91, size: 14
producer: sum: 105, size: 15
producer: sum: 120, size: 16
producer: sum: 136, size: 17
producer: sum: 153, size: 18
producer: sum: 171, size: 19
producer: sum: 190, size: 20
producer: sum: 210, size: 21
producer: sum: 231, size: 22
producer: sum: 253, size: 23
producer: sum: 276, size: 24
producer: sum: 300, size: 25
producer: sum: 325, size: 26
producer: sum: 351, size: 27
producer: sum: 378, size: 28
producer: sum: 406, size: 29
producer: sum: 435, size: 30
producer: sum: 465, size: 31
consumer: sum: 0, size: 30
consumer: sum: 1, size: 30
producer: sum: 496, size: 31
producer: sum: 528, size: 30
producer: sum: 561, size: 31
producer: sum: 595, size: 32
producer: sum: 630, size: 33
producer: sum: 666, size: 34
producer: sum: 703, size: 35
producer: sum: 741, size: 36
producer: sum: 780, size: 37
producer: sum: 820, size: 38
producer: sum: 861, size: 39
producer: sum: 903, size: 40
producer: sum: 946, size: 41
producer: sum: 990, size: 42
producer: sum: 1035, size: 43
producer: sum: 1081, size: 44
producer: sum: 1128, size: 45
producer: sum: 1176, size: 46
producer: sum: 1225, size: 47
producer: sum: 1275, size: 48
consumer: sum: 3, size: 29
consumer: sum: 6, size: 48
consumer: sum: 10, size: 47
consumer: sum: 15, size: 46
consumer: sum: 21, size: 45
consumer: sum: 28, size: 44
producer: sum: 1326, size: 49
producer: sum: 1378, size: 44
producer: sum: 1431, size: 45
```

Part 2:



```
R_0: data: 0, copy: 0
R_1: data: 0, copy: 0
R_2: data: 0, copy: 0
R_3: data: 0, copy: 0
R_4: data: 0, copy: 0
R_5: data: 0, copy: 0
R_6: data: 0, copy: 0
R_7: data: 0, copy: 0
R_8: data: 0, copy: 0
R_9: data: 0, copy: 0
R_10: data: 0, copy: 0
R_11: data: 0, copy: 0
R_12: data: 0, copy: 0
W_0: data: 0, copy: 0
W_1: data: 1, copy: 1
W_2: data: 2, copy: 2
W_3: data: 3, copy: 3
W_4: data: 4, copy: 4
W_5: data: 5, copy: 5
W_6: data: 6, copy: 6
W_7: data: 7, copy: 7
W_8: data: 8, copy: 8
W_9: data: 9, copy: 9
W_10: data: 0, copy: 0
W_11: data: 1, copy: 1
W_12: data: 2, copy: 2
W_13: data: 3, copy: 3
W_14: data: 4, copy: 4
W_15: data: 5, copy: 5
W_16: data: 6, copy: 6
W_17: data: 7, copy: 7
W_18: data: 8, copy: 8
W_19: data: 9, copy: 9
W_20: data: 0, copy: 0
W_21: data: 1, copy: 1
W_22: data: 2, copy: 2
W_23: data: 3, copy: 3
W_24: data: 4, copy: 4
W_25: data: 5, copy: 5
W_26: data: 6, copy: 6
W_27: data: 7, copy: 7
W_28: data: 8, copy: 8
W_29: data: 9, copy: 9
W_30: data: 0, copy: 0
W_31: data: 1, copy: 1
W_32: data: 2, copy: 2
R_0: data: 0, copy: 0
R_1: data: 3, copy: 3
R_2: data: 3, copy: 3
R_3: data: 3, copy: 3
R_4: data: 3, copy: 3
R_5: data: 3, copy: 3
R_6: data: 3, copy: 3
R_7: data: 3, copy: 3
```