**Introduction to Operating Systems**

**COP 4600-002**

Name and ID \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Worksheet # 8**

Q1. Suppose that processes P1, P2, and P3 shown below running concurrently. S1 and S2 are among the statements that P1 will eventually execute, S3 and S4 are among the statements that P2 will eventually execute and S5 and S6 are among the statements that P3 will eventually execute.

You need to use semaphores to guarantee that statement S3 is executed AFTER statement S6 has been executed – also that statement S1 will be executed BEFORE statement S4 – also that statement S5 is executed BEFORE statement S2.

Show, within the structure of the processes below, how you would use semaphores to coordinate these three processes. Include semaphore names and to which values you would initialize the semaphores.

Which semaphores are you using and their initial values:

Answer:

Process P1 Process P2 Process P3

S1 S3 S5

S2 S4 S6

After the semaphores have been appropriately placed, is it possible to determine which of the 6 statements above will be executed first?

If YES – which one? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

And which of the 6 statements above will be executed last?

If YES – which one \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q2. A producer and a consumer threads are sharing a buffer as shown below. The buffer is initialized with “blank” characters in each position. Semaphores X, Y, and Z are used to protect the buffer and are initializing properly. After the threads have been running for a while, the buffer appears as shown below and the threads are executing at the location shown by the arrows below. If the last character placed in the buffer was ‘a’ and the value of semaphore Y is 4, answer the question below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | a | 8 | d | f | 1 | 9 | s | m | c | w | 7 | b |

**Consumer** **Producer**

Repeat Repeat

Do other calculations do other calculations

P(X) P(Y)

P(Z) P(Z)

Read item from buffer Place item in buffer

V(Z) V(Z)

Do other calculations

V(Y) do other calculations

V(X)

Until false Until false

1. What was the value of semaphore X when it was initialized? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. What is the value of semaphore X now? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
3. What is the value of the consumer buffer index now? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
4. What is the value of the producer buffer index now? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Which was the last character read by the consumer? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
6. Which specific character that are still in the buffer HAVE BEEN READ by the consumer

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Q3. The CPU and I/O times for 3 processes are shown below. Assume that PI gets to the ready queue just before P2 and P3 after P2. The scheduling algorithm used by the OS is Round Robin with a time slice of 4 time units. Assume that I/Os for the processes are different so that there is no I/O queue. Assume also that an interrupt from a completed I/O for process "X" will place process "X" in the ready queue BEHIND the process that was just interrupted from CPU.

1. Using the first empty graph, describe how the CPU will be assigned to each process and for how long.
2. Use the second empty graph to show ALL the states that P1 goes through and the amount of time it has remained in that state until it has halted.

P1

CPU

I/O

8

7

3

Halt

P2

CPU

I/O

3

5

2

halt

P3

CPU

I/O

6

4

2

halt

Busy CPU

Idle CPU