

Q-23

Problem - 1

Assume.

binarySemaphore n represents filled slot in buffer
binarySemaphore s represents mutex lock behaviour

Problem - 1

Initially binarySemaphore $s = 0$.

If this is the case, then Producer & consumer won't be able to progress further from `semWait(s)` line. as s is 0. So both will keep on waiting at that line resulting in deadlock.

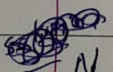
Solution

So to solve this the binarySemaphore s should be initialized with 1 (equivalent to open lock) so one of them can consume the semaphore.

Problem - 2

Even if now binarySemaphore $s = 1$.

Now if consumer runs first and execute `semWait(s)`, then s is made 0, and then consumer will wait on `semWait(n)` as nothing is there in buffer to read.



Now suppose switch happen and producer begin, then at `semWait(s)` it will stuck as s is made 0 by consumer and this will result in deadlock.

Solution

To avoid the consumer to take lock on buffer (with s) even there is no data, we have to flip the lines. `semWait(s)`, `semWait(n)`, so that if data is there in buffer

Date: _____

which is represented by n , then only lock on $\&$ can be taken by consumer.

```
void consumer () {
```

```
    while(true) {
```

```
        semWait(m);
```

```
        semWait(s);
```

```
        ;
```

```
    }
```

//switched both lines.

Problem-3

If we have to have use of all N spaces of buffers, then it is not possible with binary semaphore.

Because let's say after one iteration by producer, 1 value is put in buffer and binary semaphore n is made 1.

Now next iteration if producer tries to write, it can not increment n as it is already 1. (\therefore binary semaphore).

Solution

So to represent usage of all N spaces of buffer the binary semaphore n (representing full slot) should be replaced with semaphore taking non-negative value N or ∞ . So that producer can increment semaphore n to N at ∞ . To N max,

Problem-4

If above all things are solved then also, there is issue of buffer overflow by producer as there is no restriction of putting values in buffer on producer. So producer can even produce even if buffer is full ($n = N$).

Solution

So to solve it, we need to introduce semaphore $emptySlot = N$ which represents N empty space in buffer. Every time Producer plan to write it decreament $emptySlot$ semaphore. Consumer on consuming signal space available by incrementing this $emptySlot$ semaphore.