ECSE 427 - Assignment 3

John Wu - #260612056

April 11, 2018

1 Theoretical Questions

1.1 When we already use 2 semaphores in the producer-consumer problem why is there a need for mutex?

A semaphore and mutex are not the same thing and are used for different purposes. A semaphore is a signaling mechanism while a mutex is a locking mechanism. In the producer-consumer problem, one semaphore is used to block the producer when the buffer is full and one semaphore is used to block the consumer when the semaphore is empty. The mutex is then needed to lock the buffer itself to ensure exclusive access to it.

1.2 Is it possible that a consumer with lowest priority suffers from starvation in the 2 semaphore and 1 mutex setup for producer-consumer? Explain the situation.

It is possible that consumer with lowest priority suffers from starvation. This happens when there are no more producers and the higher priority consumers have already emptied the buffer. The solution to solving this problem is on a case by case basis. In the previous assignment, this problem was solved by destroying the consumer (taxi) threads after a certain time limit (the point in time where there were no more planes arriving).

1.3 Though binary semaphores avoid starvation and mutexes dont, why is it recommended to use mutex in producer-consumer to secure the critical section?

Binary semaphores are used for synchronization while mutexes are used for controlling mutual access. Binary semaphores allow external entities to interrupt the owner of the resource and ask for control during wait states in its routine. However, when dealing with a critical section, we want the owner of the resource to have full control and exclusive access to complete its full routine before releasing it. For this reason, mutexes should always be used for any critical section.

1.4 Why do producer-consumer need to have 2 semaphores namely full and empty. What complications may arise if we use only one semaphore for full and rely on computed complementary value for empty.

The ”full” semaphore is used to keep track of when the buffer is full and block the producer from accessing it when this is the case. The ”empty” semaphore is used to keep track of when the buffer is empty and block the consumer from accessing it when this is the case. In the previous assignment, only one semaphore was needed because as soon as the buffer (taxi line) was full, all the producers (passengers) went to take the bus instead of waiting for the buffer to empty. If this wasn't the case then the producers would wait for the only semaphore to send a signal indicating that the buffer has space. At the same time, the arriving consumers would also be waiting for the same signal, but would have a lower priority over the producers that arrived first. Therefore, even though the consumer should have access to the resource because the buffer is not empty, it's not able to because the higher priority producers must access first. Since the consumers are the only ones who can empty the buffer, this essentially creates a deadlock where all incoming producers and consumers are ultimately blocked.

2 Disk Scheduling Algorithm

The disk scheduling algorithm can run by compiling the attached file using *gcc 260612056_part2.c*.

3 Bankers Algorithm

3.1 Question 1

The normal Bankers algorithm can be run by compiling the attached file using *gcc -pthread resource_request_simulator.c*

3.2 Question 2

The faulty Bankers algorithm can be run by compiling the attached file using *gcc -pthread faulty_resource_request_simulator*