OS Project#1

Name: Chang Liu Email: chang_liu@student.uml.edu Phone: 978-942-9725

Question#1:

1) When I run the code with 100 times, the rate of deadlock is 20%; when I run the code with 50 33times, the rate of deadlock is 27%; when I run the code with 30 times, the rate is 33%, so in the first situation (1 producer, 5 consumers, 50 slots), the rate of deadlock is between 20% - 30%, nearly around 30%, as large data test could avoid the coincidence of runtime.

Table 1 runtime times and its corresponding deadlock rate

Runtime times	Deadlock rate
30	33%
50	27%
100	20%

2) Now that we get the result of fixed configuration, we now change the depth setting. That is changing the slot size. This distribution is a **linear** distribution, I have tested a large set of data, varying the slot number from 20 to 100, and the deadlock probability is as follows:

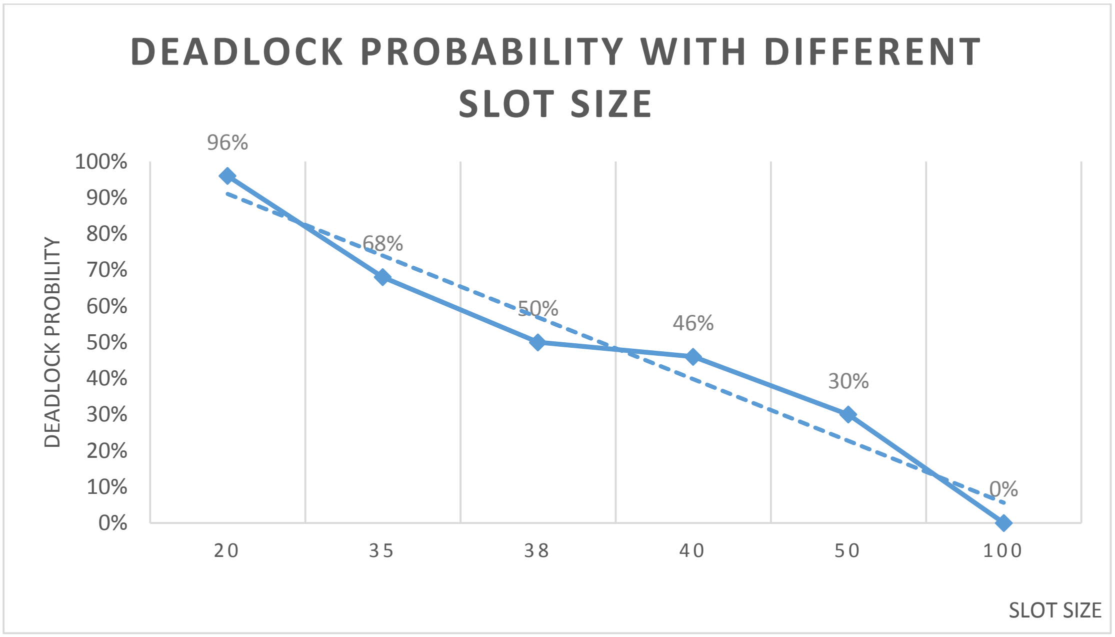

Figure 1. probability of deadlock with different slot size

As the factors of affecting the times of deadlock is very big, I have to test many times and calculate the average of the data, as the following shows:

Deadlock probability	Slot size
20	96%
35	68%
38	50%
40	46%
50	30%
100	0%

In order to get the precise of the slot size when the deadlock probability is 50%, we use the test data, from 1) we could know that when slot size is 50, it is around 30%; after testing fewer times which is 30, we could get to know that the fewer slot size, the higher probability. So after calculating the deadlock rate, we could locate the size from 35 to 40, which make the deadlock rate is 50%, and the overall distribution is linear. **My best result data is 38 when the deadlock rate is 50% precisely**, but sometimes it maybe 40 or 35 around, depending on the environment.

My observation is that with larger slot size, there is less possibility to deadlock, as the larger the slot size is, the less conflict it will make to collaborate and wait. Image there is less than 20 slots, then with 1 producer and 5 consumers, there will be more chances to fill up the slot, and there is more chance to wait for the empty slot, then the multiple consumer thread have to wait, generating the deadlock.

When deadlock occurs, it will show some error information as the error p, v operation identifier removed, as the output result shows in the output file, but in total the distribution is in linear as the deadlock is directly connected with the slot size.

Question#2:

The second deadlock probability distribution is as follows (Figure2 shows):

From the figure we could conclude that the distribution is not a linear distribution, for there is a sharp change in the trend of the line. Unlike the first figure, we could only set the consumer number to integers, and from 2 to 3 and 6 to 7, the line's leanness is apparently differently from the before, we could hardly ignore this exception. And from 7 to 10, the line is almost in horizon.

My conclusion is:

1) When consumer size is from 3 to 6, the distribution is nearly linear.
2) After a specific number, the deadlock rate doesn't change.

3) In some sequence, the distribution is partly linear.

My analysis is:

This time the variable is consumer number, however the deadlock rate means the consumer and producer’s relation about how to release and take lock. In previous example, the larger the size, the less possibility of filling the slot, then deadlock rate is low. However, this time consumer size is not directly affected by consumer size, even though as it increase the deadlock rate is becoming high, because more consumer take chances to get resources.

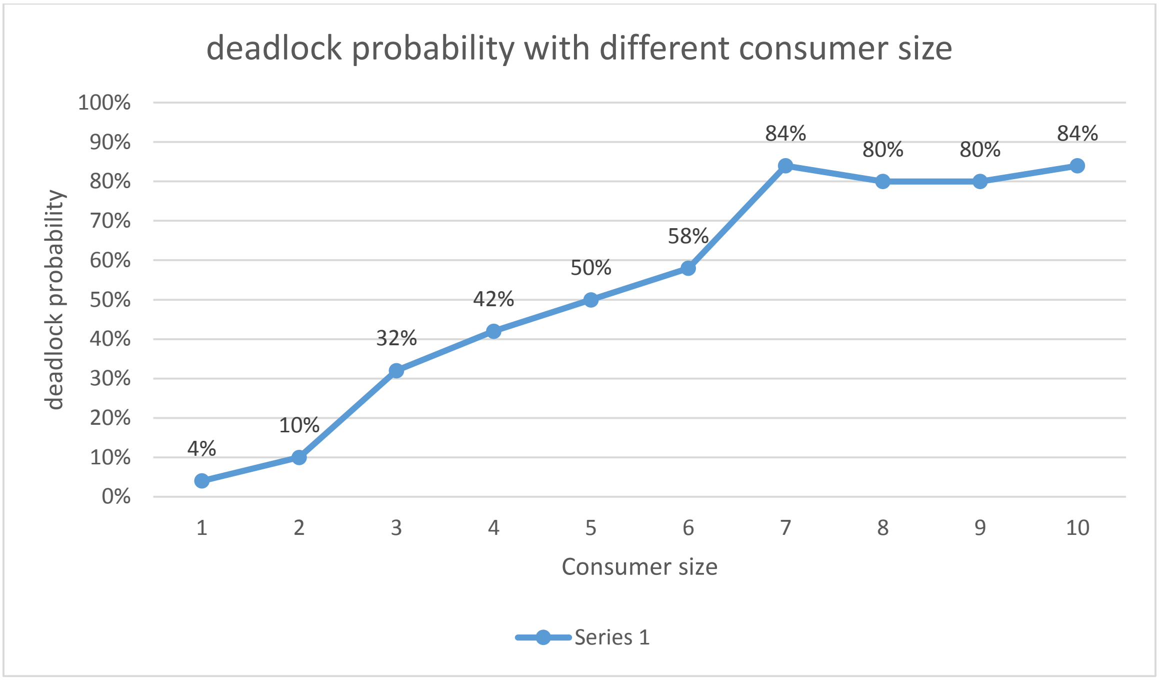

Figure 2. Deadlock probability with different consumer size

Conclusion#3:

From this project, I learned about the POSIX semaphore and shared memory API on Linux, I also mastered the basic skills in multiprocessing and data sharing under Linux environment, In some situations the data is corrupted, I learned that the sequence of acquiring lock and releasing lock is very important, and deadlock occurs when producer and consumer both wait the other side to release resource but both of them are unavailable. The more resources there are (which in this experiment is slot size), the less chances of getting deadlocked. And with more process competing the resources, as the second test shows, the more chances of getting deadlock. But these two situations are different as their influence on the deadlock rate.