Amit Kumar Choudhary (UMS # 01248671)
amit_choudhary@student.uml.edu

Operating System(91.515) Assignment #1

Objective

The Objective of the work is to implement a producer consumer problem with one producer and multiple consumer using semaphores and shared memory.

Design & Implementation

- **Producer**

The design of the project is based on shared memory between producer and consumer. It is shared because we wanted to analyze how does the system behave when there is just one memory resource and view deadlocks. We have declared in pointers **in_ptr** to store the position of next available slot for producer to fill for each flavor and out pointers **out_ptr** for storing the next available donut for customers for each flavor. The variable **serial** is used to store the count of donut produced for each flavor.

A donut_ring structure, is created in the shared memory region with definition as below:-

```
struct   donut_ring{
   int   flavor  [NUMFLAVORS]  [NUMSLOTS];
   int   outptr  [NUMFLAVORS];
```

The flavor array stores the donut count for each flavor and the outptr stores the next available donut for consumer. The logic of shared memory is implement in following block

```
f((shmid = shmget[1](MEMKEY, sizeof(struct donut_ring), 0)) == -1){
                perror("shared get failed: ");

        /* Now its Start Attaching shared memory */
        if((shared_ring=shmat[2](shmid, NULL, 0)) == (void *)-1){
                perror("shared attach failed: ");

        for(i=0; i<NUMSEMIDS; i++) {
                if ((semid[i] = semget[3](SEMKEY+i, NUMFLAVORS, 0)) == -1){
                perror("semaphore allocation failed: ");
```

After initial setups, random number is generated using nrand48 between 0 to 3, to let producer

[1] shmget to get shared memory id denoted by MEMKEY and of size struct donut_ring.

[2] shmat to attach shared memory to shared_ring struture using same memory id as used for creation of shared memory.

[3] semget is used to create as many semaphores as NUMFLAVORS with SEMKEY id.

know which flavor of donut to produce. The producer keeps producing continuously unless it is killed by the calling program. During each produce, the producer does a "p" operation on PROD semaphore for that particular flavor and decreases the count by one (**p(semid[PROD],j)**). The value of serial counter for that flavor is stored in donut_ring structure and the in_ptr is increased by one in ring buffer fashion to point to next location available for filling. In case, there are no empty slots to fill, the producer will wait unless there's one created by consumer process. Last, it does a "v" operation on consumer semaphore to denote that "1 more donut of flavor j is available" via **v(semid[CONSUMER],j)**.

- **Consumer**

On Consumer side, similar operations for memory happen. This time, instead of creating new memory location, the structure and semaphores attach themselves to memeory producer has created. Next, a random no generator generates between 0 to 3, as flavor of donut to be consumed by the consumer. The consumer does a "p" operation on semaphore outptr (**p(semid[OUTPTR],j)**) for particular flavor to lock it while it is consuming that flavor and then does a p operation on CUSTOMER semaphore via **p(semid[CONSUMER],j)**. The next available donut position is updated followed by "v" operation on outptr semaphore (**v(semid[OUTPTR],j)**). The PROD semaphore is processed next as "v" operation to denote additional one more slot for a donut of flavor "j". If the consumer fails to complete its operation and waits on semaphores, the deadlock will happen and consumer process will be terminated once producer gets killed as consumer required resources will not be available anymore.

<u>Result</u>

1. Case I : For 5 Consumer, 1 Producer, 10 Dozen & 10 Looks but varying Slot Size.

Sr. No.	**Test Case**	**Result**	**No. of Dead Lock**	**Errors**
1.	**No. of Slots = 50**	10 loops and 3 deadlocks	3	p operation for CONSUMER failed : Identifier removed
	No. of Slots = 55	10 loops and 0 deadlocks	0	No Error
2	**No. of Slots = 10**	10 loops and 10 deadlocks	10	p operation for CONSUMER failed : Identifier removed
3	**No. of Slots = 30**	10 loops and 8 deadlocks	8	p operation for CONSUMER failed : Identifier removed
4	**No. of Slots = 40**	10 loops and 4	4	p operation for

		deadlocks		CONSUMER failed : Identifier removed
5	**No. of Slots = 35**	10 loops and 9 deadlocks	6	p operation for CONSUMER failed : Identifier removed
6	**No. of Slots = 38**	10 loops and 5 deadlocks	5	p operation for CONSUMER failed : Identifier removed

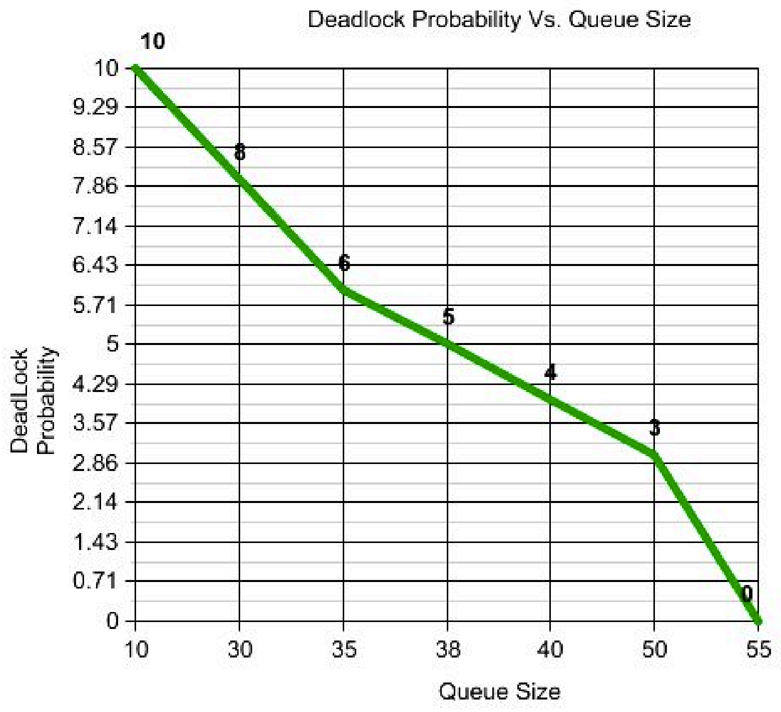

Figure 1.

The result clearly shows that as queue size increases, dead lock probability decreases for Case I. It is because, as no. of slots increases, contention among various consumers for same donut flavor will decreases even when producer is not producing that flavor donut for long time. Slot size of 38 gives 50% deadlock probability.

2. Case II. For 50% deadlock queue size, change no of customers from 1 to 10.
   In Case I, test case 6, we have found that 50% deadlock queue size is 38. Hence Constant factors are Slot size = 38, Flavor = 4, No of Loop = 10, No of Dozen = 10.

Sr. No	**Test Case**	**Result**	**Deadlock**	**Error**
1	**No. of Consumer = 1**	10 loops and 0 deadlocks	0	No Error
2	**No. of Consumer = 2**	10 loops and 0 deadlocks	0	No Error

3	**No. of Consumer = 3**	10 loops and 1 deadlocks	1	p operation for CONSUMER failed : Identifier removed
4	**No. of Consumer = 4**	10 loops and 2 deadlocks	2	p operation for CONSUMER failed : Identifier removed
5	**No. of Consumer = 5**	10 loops and 2 deadlocks	2	p operation for CONSUMER failed : Identifier removed
6	**No. of Consumer = 6**	10 loops and 3 deadlocks	3	p operation for CONSUMER failed : Identifier removed
7	**No. of Consumer = 7**	10 loops and 4 deadlocks	4	p operation for CONSUMER failed : Identifier removed
8	**No. of Consumer = 8**	10 loops and 5 deadlocks	5	p operation for CONSUMER failed : Identifier removed
9	**No. of Consumer = 9**	10 loops and 6 deadlocks	6	p operation for CONSUMER failed : Identifier removed
10	**No. of Consumer = 10**	10 loops and 5 deadlocks	5	p operation for CONSUMER failed : Identifier removed

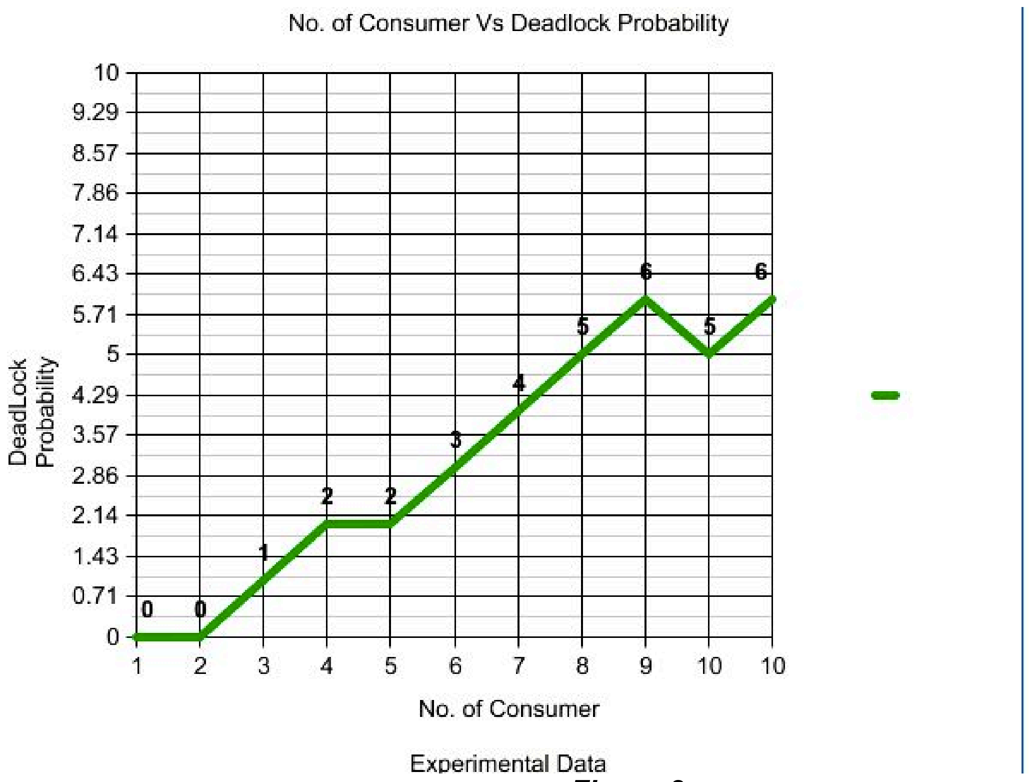

Figure 2.

For Case II, the deadlock probability increases as increase in the no of customers. It is very logical because, as customer increases, there will be more contention for a particular donut flavor. When the no of customer was one or two, we can see that there was zero deadlock which confirms our argument and as no of customer increases, dead lock probability increases. Also, as can be see from the Figure 2, we get two different values for x=10. It shows the non-ideal condition of our system where it gives different results for same input.

Files

```
-rwxr-xr-x 1 achoudha grad  387 2011-09-29 04:31 makefile
-rwxr-xr-x 1 achoudha grad 1126 2011-10-08 01:08 myutilities.c
-rwxr-xr-x 1 achoudha grad 2300 2011-10-09 00:37 myconsumer.c
-rw-r--r-- 1 achoudha grad 3531 2011-10-09 03:15 myproducer.c
-rwxr-xr-x 1 achoudha grad  537 2011-10-09 04:07 mydonuts.h
-rwxr-xr-x 1 achoudha grad 1721 2011-10-09 05:18 runprogram.sh
-rw-r--r-- 1 achoudha grad 1111 2011-10-09 06:27 output.txt
```

Success Rate & Challenges

My overall **success rate** will be **around 80%.** I faced multiple challenges when starting with this project. Since it was the first time I was reading Operating System as academic curriculum, I spent initial few weeks studying basics to build the foundation. After becoming comfortable with the semaphore concepts, I spent time in understanding the various blocks of the project and read about underlying system programming concepts required for each block.

After building enough confidence, I started this assignment and below are the technical/

conceptual challenges I faced and sought help of Professor Moloney in few cases.

1. I thought ring buffer to be a linked list type only to realize later that it is not.
2. I created shared memory for consumers too which got clarified.
3. I thought that the producer is only created and killed once for the entire loop but the doubt got cleared after talking with Professor.
4. I did p-operation for Out-Pointer initially on Producer side which was wrong.
5. I didnt had much success with getting output file as desired. I tried using array, awk array and finally got it using awk with just necessary details.

Conclusion

The overall conclusion of the project is, semaphore is a important concepts in resource management in cases where many require same resource but it has its own challenges. As the participants increases, deadlock increases and similarly for changes in other parameters. Also, after making numerous mistakes I can say I have understood the concepts well.