# *Operating Systems - EECE.5730*

***Instructor:*** *Prof. Dalila Megherbi*

*Assignment – 2*

*Due by 10-13-16*

*By,*

- *Naga Ganesh Kurapati*

# 1) Objective:

The purpose of this Lab is to initialize producer and buyer threads. Then producer produce item and puts it on the queue. Consumer thread consume the item. The synchronization is achieved using one mutex and semaphores.

# 2) Background:

Mutex are used to synchronize the threads accessing the critical sections on the program. It has can be either 1 or 0. 1 if thread has lock to access the critical section where as 0 then it has to wait for the lock. Semaphore can allow multiple threads to access the critical section. It can have more than two values unlike mutex. If it is 0, thread has to wait for the semaphore, if it is >0 thread can take the semaphore to access the critical section. Also thread can post the value to the semaphore after it done with the critical section.

# 3) Algorithms/Functions used:

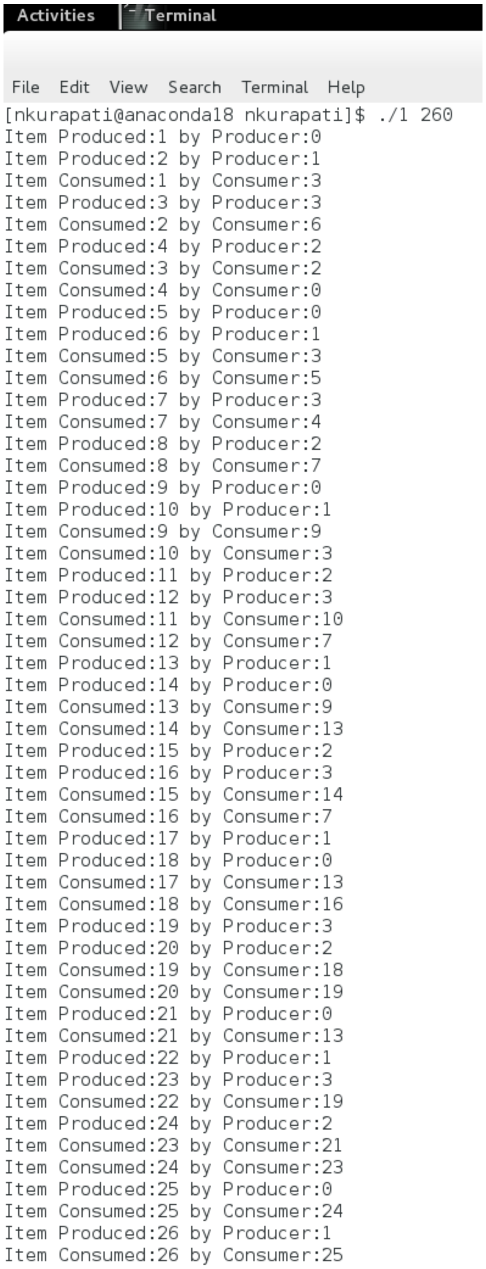
pthread_mutex_init() - to initialize mutex
sem_init() – to initialize semaphore
malloc() – to allocate memory
pthread_create() – to create pthreads
pthread_join() – to wait for threads to complete and join
pthread_mutex_destroy() – destroy mutex
sem_destroy() – destroy semaphore
sem_wait() – wait on semaphore
sem_post () – post the semaphore
pthread_mutex_lock() – mutex lock
pthread_mutex_unlock() -mutex unlock

**Userdefined functions:**

Produce() – to produce an item and place it on the queue
Consume() – to consume an item from the queue

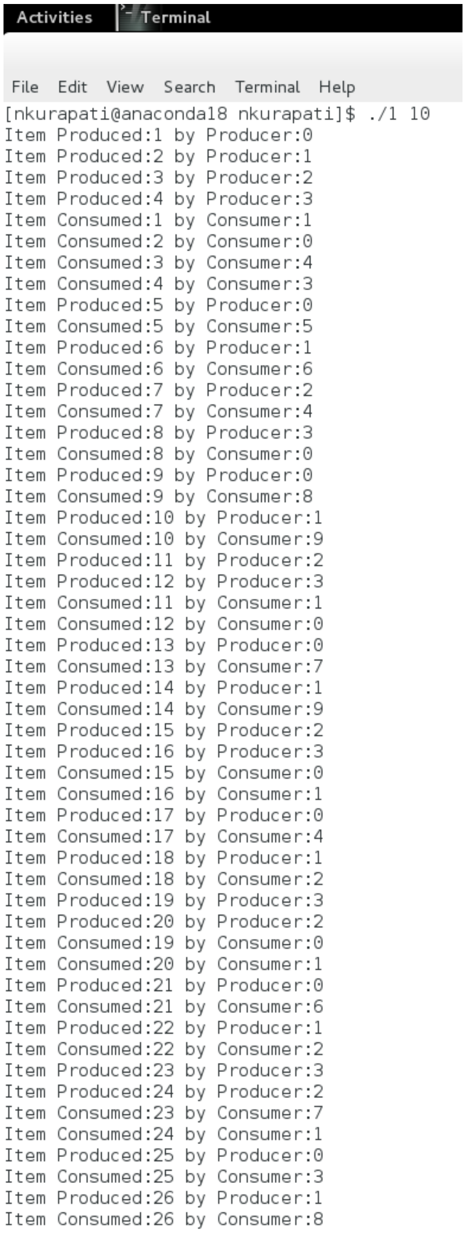
# 4) Results:

For program 1.c, two iteration are provided below with 260 buyers and 10 buyers
For program 2.c, two iteration are provided below with 6 buyers and 10 buyers



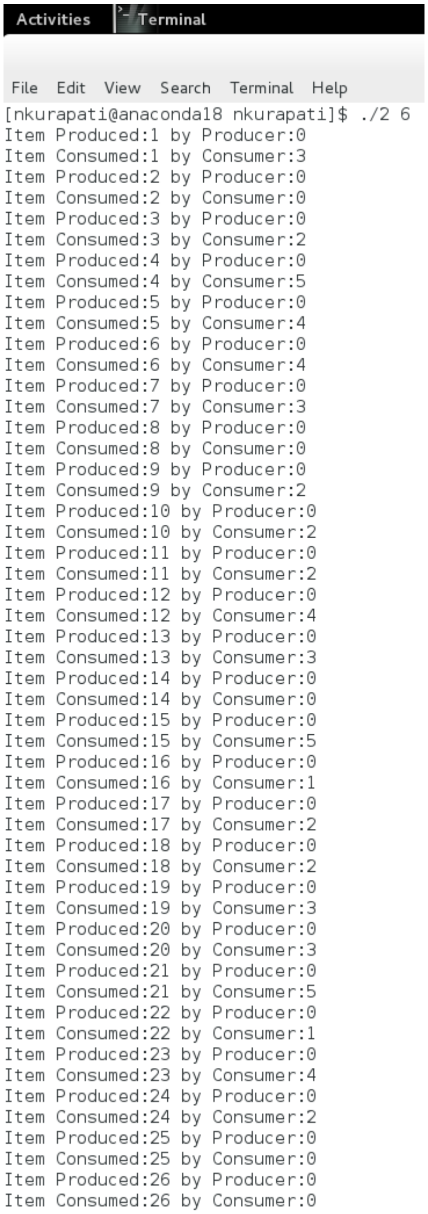
```
[nkurapati@anaconda18 nkurapati]$ ./1 260
Item Produced:1 by Producer:0
Item Produced:2 by Producer:1
Item Consumed:1 by Consumer:3
Item Produced:3 by Producer:3
Item Consumed:2 by Consumer:6
Item Produced:4 by Producer:2
Item Consumed:3 by Consumer:2
Item Consumed:4 by Consumer:0
Item Produced:5 by Producer:0
Item Produced:6 by Producer:1
Item Consumed:5 by Consumer:3
Item Consumed:6 by Consumer:5
Item Produced:7 by Producer:3
Item Consumed:7 by Consumer:4
Item Produced:8 by Producer:2
Item Consumed:8 by Consumer:7
Item Produced:9 by Producer:0
Item Produced:10 by Producer:1
Item Consumed:9 by Consumer:9
Item Consumed:10 by Consumer:3
Item Produced:11 by Producer:2
Item Produced:12 by Producer:3
Item Consumed:11 by Consumer:10
Item Consumed:12 by Consumer:7
Item Produced:13 by Producer:1
Item Produced:14 by Producer:0
Item Consumed:13 by Consumer:9
Item Consumed:14 by Consumer:13
Item Produced:15 by Producer:2
Item Produced:16 by Producer:3
Item Consumed:15 by Consumer:14
Item Consumed:16 by Consumer:7
Item Produced:17 by Producer:1
Item Produced:18 by Producer:0
Item Consumed:17 by Consumer:13
Item Consumed:18 by Consumer:16
Item Produced:19 by Producer:3
Item Produced:20 by Producer:2
Item Consumed:19 by Consumer:18
Item Consumed:20 by Consumer:19
Item Produced:21 by Producer:0
Item Consumed:21 by Consumer:13
Item Produced:22 by Producer:1
Item Produced:23 by Producer:3
Item Consumed:22 by Consumer:19
Item Produced:24 by Producer:2
Item Consumed:23 by Consumer:21
Item Consumed:24 by Consumer:23
Item Produced:25 by Producer:0
Item Consumed:25 by Consumer:24
Item Produced:26 by Producer:1
Item Consumed:26 by Consumer:25
```

Fig1: Output of program 1 with 260 buyers



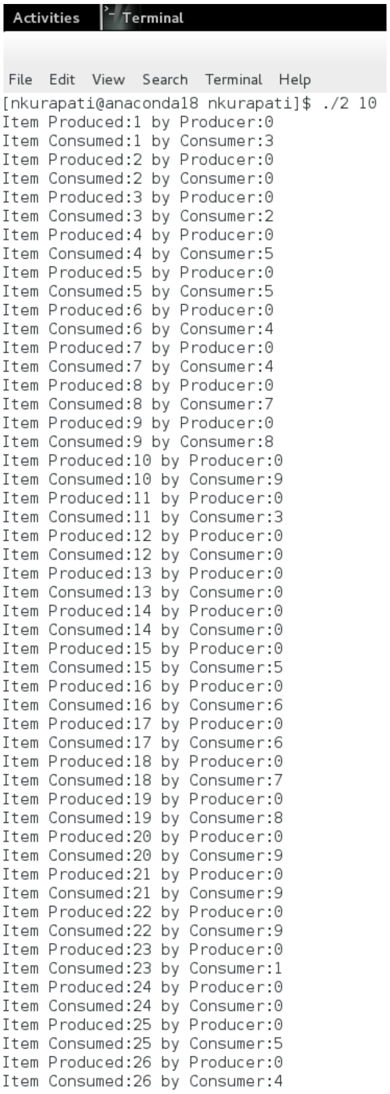
```
[nkurapati@anaconda18 nkurapati]$ ./1 10
Item Produced:1 by Producer:0
Item Produced:2 by Producer:1
Item Produced:3 by Producer:2
Item Produced:4 by Producer:3
Item Consumed:1 by Consumer:1
Item Consumed:2 by Consumer:0
Item Consumed:3 by Consumer:4
Item Consumed:4 by Consumer:3
Item Produced:5 by Producer:0
Item Consumed:5 by Consumer:5
Item Produced:6 by Producer:1
Item Consumed:6 by Consumer:6
Item Produced:7 by Producer:2
Item Consumed:7 by Consumer:4
Item Produced:8 by Producer:3
Item Consumed:8 by Consumer:0
Item Produced:9 by Producer:0
Item Consumed:9 by Consumer:8
Item Produced:10 by Producer:1
Item Consumed:10 by Consumer:9
Item Produced:11 by Producer:2
Item Produced:12 by Producer:3
Item Consumed:11 by Consumer:1
Item Consumed:12 by Consumer:0
Item Produced:13 by Producer:0
Item Consumed:13 by Consumer:7
Item Produced:14 by Producer:1
Item Consumed:14 by Consumer:9
Item Produced:15 by Producer:2
Item Produced:16 by Producer:3
Item Consumed:15 by Consumer:0
Item Consumed:16 by Consumer:1
Item Produced:17 by Producer:0
Item Consumed:17 by Consumer:4
Item Produced:18 by Producer:1
Item Consumed:18 by Consumer:2
Item Produced:19 by Producer:3
Item Produced:20 by Producer:2
Item Consumed:19 by Consumer:0
Item Consumed:20 by Consumer:1
Item Produced:21 by Producer:0
Item Consumed:21 by Consumer:6
Item Produced:22 by Producer:1
Item Consumed:22 by Consumer:2
Item Produced:23 by Producer:3
Item Produced:24 by Producer:2
Item Consumed:23 by Consumer:7
Item Consumed:24 by Consumer:1
Item Produced:25 by Producer:0
Item Consumed:25 by Consumer:3
Item Produced:26 by Producer:1
Item Consumed:26 by Consumer:8
```

Fig.2: Output of Program 1 with 10 buyers



```
[nkurapati@anaconda18 nkurapati]$ ./2 6
Item Produced:1 by Producer:0
Item Consumed:1 by Consumer:3
Item Produced:2 by Producer:0
Item Consumed:2 by Consumer:0
Item Produced:3 by Producer:0
Item Consumed:3 by Consumer:2
Item Produced:4 by Producer:0
Item Consumed:4 by Consumer:5
Item Produced:5 by Producer:0
Item Consumed:5 by Consumer:4
Item Produced:6 by Producer:0
Item Consumed:6 by Consumer:4
Item Produced:7 by Producer:0
Item Consumed:7 by Consumer:3
Item Produced:8 by Producer:0
Item Consumed:8 by Consumer:0
Item Produced:9 by Producer:0
Item Consumed:9 by Consumer:2
Item Produced:10 by Producer:0
Item Consumed:10 by Consumer:2
Item Produced:11 by Producer:0
Item Consumed:11 by Consumer:2
Item Produced:12 by Producer:0
Item Consumed:12 by Consumer:4
Item Produced:13 by Producer:0
Item Consumed:13 by Consumer:3
Item Produced:14 by Producer:0
Item Consumed:14 by Consumer:0
Item Produced:15 by Producer:0
Item Consumed:15 by Consumer:5
Item Produced:16 by Producer:0
Item Consumed:16 by Consumer:1
Item Produced:17 by Producer:0
Item Consumed:17 by Consumer:2
Item Produced:18 by Producer:0
Item Consumed:18 by Consumer:2
Item Produced:19 by Producer:0
Item Consumed:19 by Consumer:3
Item Produced:20 by Producer:0
Item Consumed:20 by Consumer:3
Item Produced:21 by Producer:0
Item Consumed:21 by Consumer:5
Item Produced:22 by Producer:0
Item Consumed:22 by Consumer:1
Item Produced:23 by Producer:0
Item Consumed:23 by Consumer:4
Item Produced:24 by Producer:0
Item Consumed:24 by Consumer:2
Item Produced:25 by Producer:0
Item Consumed:25 by Consumer:0
Item Produced:26 by Producer:0
Item Consumed:26 by Consumer:0
```

Fig.3: Output of Program 2 with 6 buyers



```
[nkurapati@anaconda18 nkurapati]$ ./2 10
Item Produced:1 by Producer:0
Item Consumed:1 by Consumer:3
Item Produced:2 by Producer:0
Item Consumed:2 by Consumer:0
Item Produced:3 by Producer:0
Item Consumed:3 by Consumer:2
Item Produced:4 by Producer:0
Item Consumed:4 by Consumer:5
Item Produced:5 by Producer:0
Item Consumed:5 by Consumer:5
Item Produced:6 by Producer:0
Item Consumed:6 by Consumer:4
Item Produced:7 by Producer:0
Item Consumed:7 by Consumer:4
Item Produced:8 by Producer:0
Item Consumed:8 by Consumer:7
Item Produced:9 by Producer:0
Item Consumed:9 by Consumer:8
Item Produced:10 by Producer:0
Item Consumed:10 by Consumer:9
Item Produced:11 by Producer:0
Item Consumed:11 by Consumer:3
Item Produced:12 by Producer:0
Item Consumed:12 by Consumer:0
Item Produced:13 by Producer:0
Item Consumed:13 by Consumer:0
Item Produced:14 by Producer:0
Item Consumed:14 by Consumer:0
Item Produced:15 by Producer:0
Item Consumed:15 by Consumer:5
Item Produced:16 by Producer:0
Item Consumed:16 by Consumer:6
Item Produced:17 by Producer:0
Item Consumed:17 by Consumer:6
Item Produced:18 by Producer:0
Item Consumed:18 by Consumer:7
Item Produced:19 by Producer:0
Item Consumed:19 by Consumer:8
Item Produced:20 by Producer:0
Item Consumed:20 by Consumer:9
Item Produced:21 by Producer:0
Item Consumed:21 by Consumer:9
Item Produced:22 by Producer:0
Item Consumed:22 by Consumer:9
Item Produced:23 by Producer:0
Item Consumed:23 by Consumer:1
Item Produced:24 by Producer:0
Item Consumed:24 by Consumer:0
Item Produced:25 by Producer:0
Item Consumed:25 by Consumer:5
Item Produced:26 by Producer:0
Item Consumed:26 by Consumer:4
```

Fig.4: Output of Program 2 with 10 buyers

## 5) Observations:

In Program 2, since we have only one buyer, we had a chance to reduce one semaphore. So, in program we used to one mutex and two semaphore to design the problem where as in program 2 we used only one mutex and one semaphore.

## 6) Conclusions:

Successfully created the producer and buyer threads according to the problem designed. Producer produce at least one item and buyer consumes at least one item.

## 7) Source Code:

```
/*-------------1.c-----------------*/
    #include<stdio.h>
    #include<stdlib.h>
    #include<unistd.h>
    #include<pthread.h>
    #include<semaphore.h>
    #include<signal.h>

    //Define the size of the queue
    #define BufferSize 10

    //Defintion of Queue
    typedef struct {
      int in;   //Number of items on queue
      int out; //Number of itmes consumed
      int buffer[BufferSize]; //buffer to place items produced
    }item;

    //Intialize queue
    item itemBuf = {0,0,{0}};
    //Item unique Id
    unsigned int itemId = 1;

    //Mutex & semaphore
    pthread_mutex_t mutex;
    sem_t full, empty;

    //Producer function
    void *Produce(void *id)
    {
      while(1)
        {
          //wait on empty place on buffer to place an item
          sem_wait(&empty);
          //wait for the critical area access
          pthread_mutex_lock(&mutex);
          //check if queue is full
          if(!(((itemBuf.in + 1) % BufferSize ) == itemBuf.out))
            {
              //Plcae the item on buffer
              itemBuf.buffer[itemBuf.in] = itemId++;

    printf("ItemProduced:%dbyProducer:%d\n",itemBuf.buffer[itemBuf.in],(long*)id);
              //Increment the 'in' index
              itemBuf.in = (itemBuf.in + 1) % BufferSize;
            }
          else
            {
              printf("Queue is full when accessed by Producer:%d \n",(long*)id);
            }
          //release the critical area access
          pthread_mutex_unlock(&mutex);
          //Post the produced item
          sem_post(&full);
          sleep(1);
        }
    }

    //Consumer function
    void *Consume(void *id)
    {
      while(1)
        {
          //wait on filled place on buffer to consume an item
          sem_wait(&full);
          //wait for the critical area access
          pthread_mutex_lock(&mutex);
          //Check if buffer is empty
          if(!(itemBuf.in == itemBuf.out))
            {

    printf("ItemConsumed:%dbyConsumer:%d\n",itemBuf.buffer[itemBuf.out],(long*)id);
              //Consume an item on buffer
              itemBuf.buffer[itemBuf.out] = 0;
              //Increment the 'out' index
              itemBuf.out = (itemBuf.out + 1) % BufferSize ;
            }
          else
            {
              printf("Queue is empty when accessed by Consumer:%d \n",(long*)id);
            }
          //release the critical area access
          pthread_mutex_unlock(&mutex);
          //Post the empty place
          sem_post(&empty);
          sleep(1);
        }
    }


    int main(int argc, char *argv[])
    {
      //To get the number of buyers from cmd line
      unsigned int buyersNo = atoi(argv[1]);

      //Define producer number
      unsigned int providersNo = 4;
      pthread_t p[providersNo];
      pthread_t* b;

      //Intialize mutex and two semaphores
      int e1 = pthread_mutex_init(&mutex, NULL);
      int e2 = sem_init(&full, 0, 0);
      int e3 = sem_init(&empty, 0, BufferSize);

      //Notify if failed
      if(e1!=0||e2!=0||e3!=0)
        printf("Intialization Error");

      b = malloc(buyersNo*sizeof(pthread_t));

      long i;
      //Create producer threads
      for(i=0;i<providersNo;i++)
        pthread_create(&p[i], NULL, Produce, (void*) i);

      //Create buyers threads
```

```
      for(i=0;i<buyersNo;i++)
        pthread_create(&b[i], NULL, Consume, (void*) i);

      //Wait for Producer and consumer threads to finish
      for(i=0;i<providersNo;i++)
        pthread_join(p[i], NULL);
      for(i=0;i<buyersNo;i++)
        pthread_join(b[i], NULL);

      //Destroy mutex and semaphores
      pthread_mutex_destroy(&mutex);
      sem_destroy(&full);
      sem_destroy(&empty);
      return 0;
}
  /*-------------2.c-----------------*/
    #include<stdio.h>
    #include<stdlib.h>
    #include<unistd.h>
    #include<pthread.h>
    #include<semaphore.h>
    #include<signal.h>

    //Define the size of the queue
    #define BufferSize 10

    //Defintion of Queue
    typedef struct {
      int in;   //Number of items on queue
      int out; //Number of itmes consumed
      int buffer[BufferSize]; //buffer to place items produced
    }item;

    //Intialize queue
    item itemBuf = {0,0,{0}};
    //Item unique Id
    unsigned int itemId = 1;

    //Mutex & semaphore
    pthread_mutex_t mutex;
```

```
sem_t full;

//Producer function
void *Produce(void *id)
{
  while(1)
    {
      //wait for the critical area access
      pthread_mutex_lock(&mutex);
      //check if queue is full
      if(!(((itemBuf.in + 1) % BufferSize ) == itemBuf.out))
        {
          //Plcae the item on buffer
          itemBuf.buffer[itemBuf.in] = itemId++;

printf("ItemProduced:%dbyProducer:%d\n",itemBuf.buffer[itemBuf.in],(long*)id);
          //Increment the 'in' index
          itemBuf.in = (itemBuf.in + 1) % BufferSize;
        }
      else
        {
          printf("Queue is full when accessed by Producer:%d \n",(long*)id);
        }
      //release the critical area access
      pthread_mutex_unlock(&mutex);
      //Post the produced item
      sem_post(&full);
      sleep(1);
    }
}

//Consumer function
void *Consume(void *id)
{
  while(1)
    {
      //wait on filled place on buffer to consume an item
      sem_wait(&full);
      //wait for the critical area access
      pthread_mutex_lock(&mutex);
      //Check if buffer is empty
      if(!(itemBuf.in == itemBuf.out))
        {

printf("ItemConsumed:%dbyConsumer:%d\n",itemBuf.buffer[itemBuf.out],(long*)id);
          //Consume an item on buffer
          itemBuf.buffer[itemBuf.out] = 0;
          //Increment the 'out' index
          itemBuf.out = (itemBuf.out + 1) % BufferSize ;
        }
      else
        {
          printf("Queue is empty when accessed by Consumer:%d \n",(long*)id);
        }
      //release the critical area access
      pthread_mutex_unlock(&mutex);
      sleep(1);
    }
}


int main(int argc, char *argv[])
{
  //To get the number of buyers from cmd line
  unsigned int buyersNo = atoi(argv[1]);

  //Define producer number
  unsigned int providersNo = 1;
  pthread_t p[providersNo];
  pthread_t* b;

  //Intialize mutex and two semaphores
  int e1 = pthread_mutex_init(&mutex, NULL);
  int e2 = sem_init(&full, 0, 0);

  //Notify if failed
  if(e1!=0||e2!=0)
    printf("Intialization Error");

  b = malloc(buyersNo*sizeof(pthread_t));
```

```
        long i;
        //Create producer threads
        for(i=0;i<providersNo;i++)
          pthread_create(&p[i], NULL, Produce, (void*) i);

        //Create buyers threads
        for(i=0;i<buyersNo;i++)
          pthread_create(&b[i], NULL, Consume, (void*) i);

        //Wait for Producer and consumer threads to finish
        for(i=0;i<providersNo;i++)
          pthread_join(p[i], NULL);
        for(i=0;i<buyersNo;i++)
          pthread_join(b[i], NULL);

        //Destroy mutex and semaphores
        pthread_mutex_destroy(&mutex);
        sem_destroy(&full);
        return 0;
}
```