SRN: PES1UG19EC326

NAME: Tanya Chanchalani

WEEK: 8&9

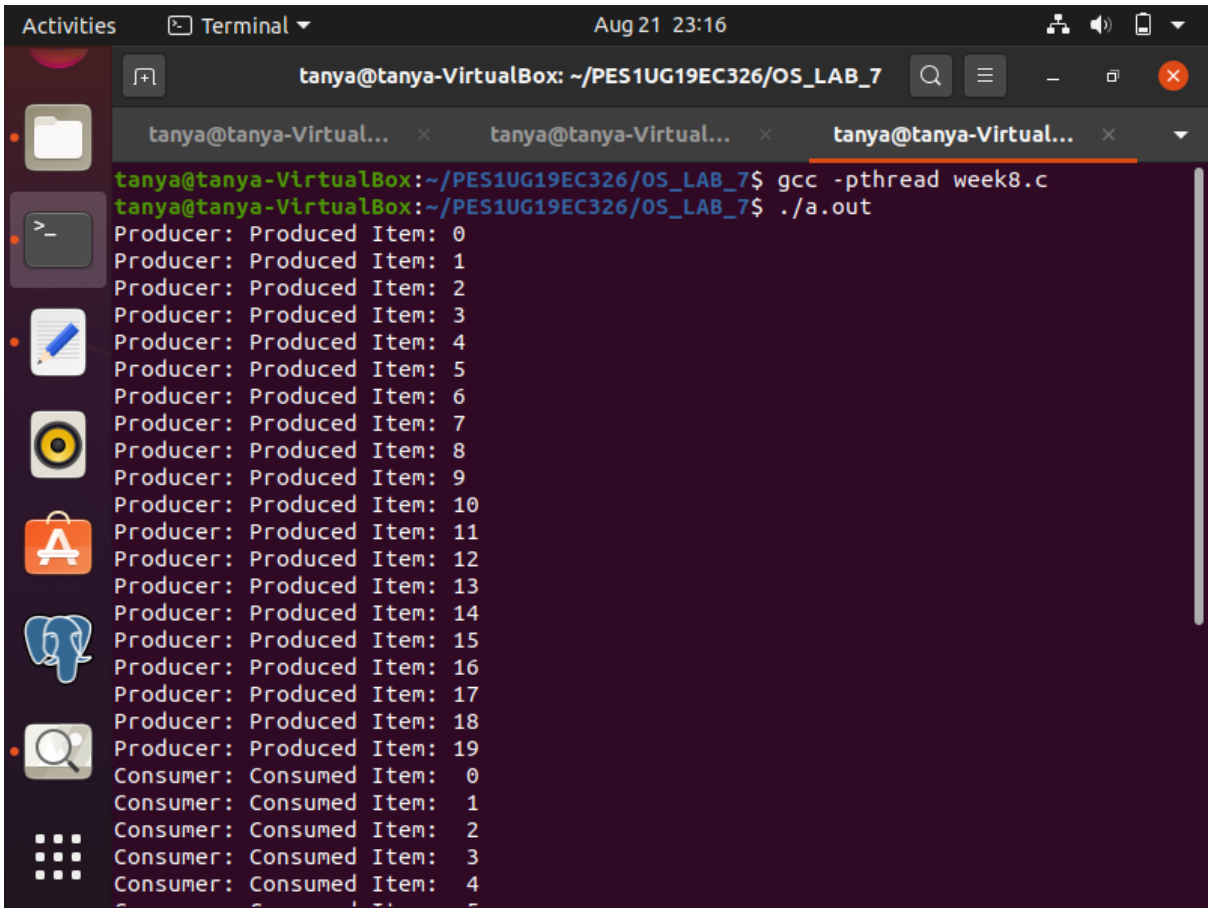
SUBJECT: OS

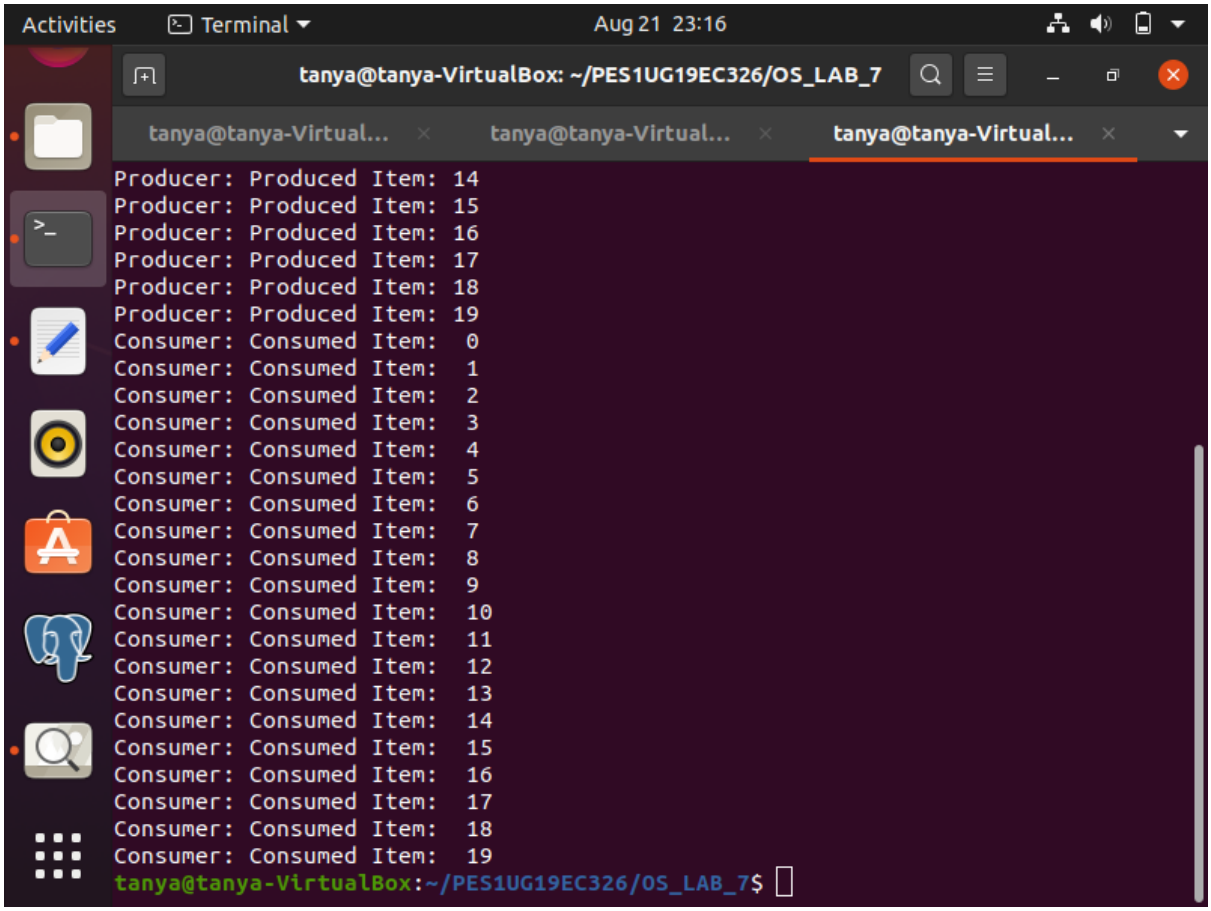
**// using binary semaphore**

```
#include<unistd.h>
#include<stdlib.h>
#include<stdio.h>
#include<pthread.h>
#define BUFF_SIZ 20
int s = 1;
int full = 0;
int empt = BUFF_SIZ;
int arr[BUFF_SIZ];
int value = 0;
void wait(int *s)
{
 while(*s <= 0);
 *s = *s - 1;
}
void signal(int* s)
{
 *s = *s + 1;
}
void *producer(void* param)
{
```

```
 for(int i=0;i<BUFF_SIZ;i++)
 {
 int new_item = value;
 value++;
 wait(&empt);
 wait(&s);
 printf("Producer:");
 printf(" Produced Item: %d\n",new_item);
 arr[i] = new_item;
 signal(&s);
 signal(&full);
 }
 pthread_exit(0);
}
void *consumer(void* param)
{
 for(int i=0;i<BUFF_SIZ;i++)
 {
 wait(&full);
 wait(&s);
 printf("Consumer:");
 printf(" Consumed Item:  %d\n", arr[i]);
 signal(&s);
 signal(&empt);
 }
 pthread_exit(0);
}
int main()
{
 pthread_t tid_p,tid_c;
 pthread_attr_t attr1,attr2;
```

```
pthread_attr_init(&attr1);
pthread_attr_init(&attr2);
pthread_create(&tid_p,&attr1,producer,NULL);
pthread_create(&tid_c,&attr2,consumer,NULL);
pthread_join(tid_p,NULL);
pthread_join(tid_c,NULL);
return 0;
}
```



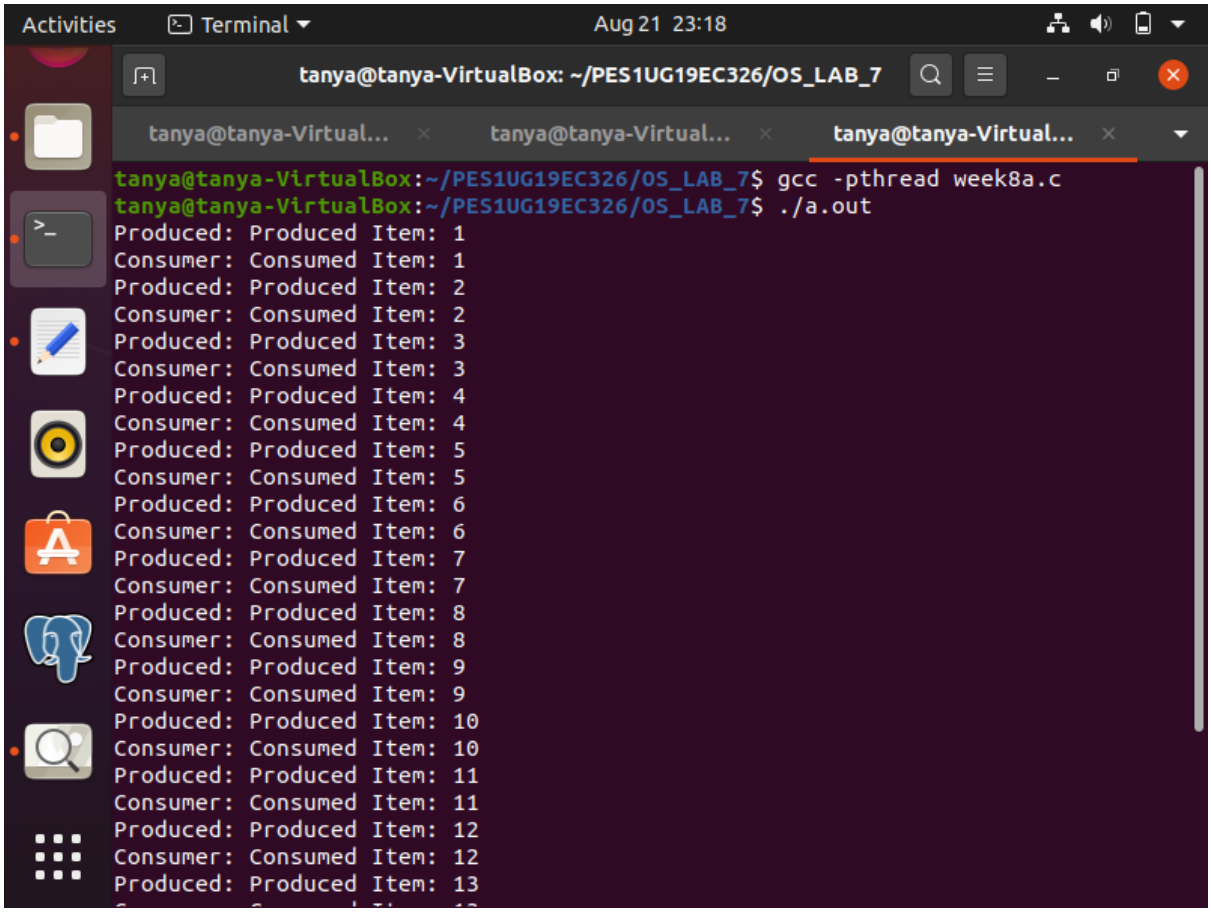



**//using semaphore -avoid critical section problems in a concurrent system**

```
#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>//Boolean data type

#include <pthread.h>//allows a program to control multiple different flows of work that overlap in time

#include <unistd.h>//header file that provides access to the POSIX operating system API

#include <semaphore.h>//define the sem_t type

#define BUFFER_SIZE 75 // total number of slots

int *buf;

int start = 0, end = 0;

sem_t mutex;// enforce mutual exclusion to shared data

sem_t empty;// keep track of the number of empty spots

sem_t full; // keep track of the number of full spots
```

```
void *consumer();
void *producer();

void *producer()
{
   int item = 0;
   while (true)
   {
      sem_wait(&empty);//lock
      sem_wait(&mutex);
      item++;
      printf("Produced: Produced Item: %d\n", item);
      buf[start] = item;
      sem_post(&mutex);//unlock
      sem_post(&full);
      start = (start + 1) % BUFFER_SIZE;//using circular queue
   }
}
void *consumer()
{
   while (true)
   {
      sem_wait(&full);
      sem_wait(&mutex);
      int consumed = buf[end];
      printf("Consumer: Consumed Item: %d\n", consumed);
      sleep(1);
      end = (end + 1) % BUFFER_SIZE;
      sem_post(&mutex);
      sem_post(&empty);
   }
```

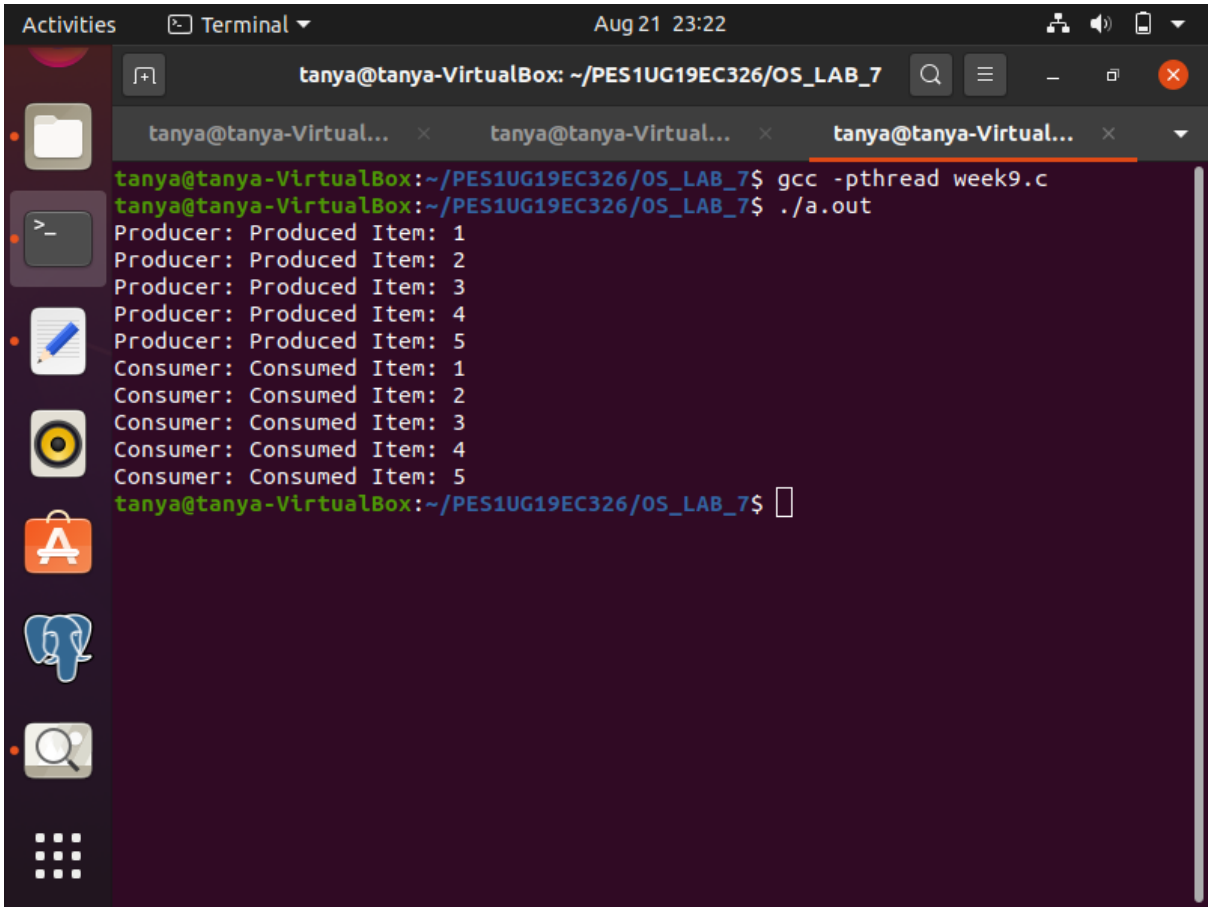
```
}
int main()
{
    buf = (int *)malloc(sizeof(int) * BUFFER_SIZE);
    pthread_t thread1, thread2;
    sem_init(&mutex, 0, 1);
    sem_init(&empty, 0, 1);
    sem_init(&full, 0, 0);
    pthread_create(&thread1, NULL, producer, NULL);
    sleep(1);
    pthread_create(&thread2, NULL, consumer, NULL);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    free(buf);
    return 0;
}
```



## //using pipes

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <unistd.h>
#include <wait.h>
void producer(FILE *);
void consumer(FILE *);
void producer(FILE *pipe_write)
{
   int item = 0;
   for (int i = 0; i < 5; ++i)
   {
      item++;
      fprintf(pipe_write, "%d ", item);
      printf("Producer: Produced Item: %d\n", item);
   }
   fclose(pipe_write);
   exit(0);
}
void consumer(FILE *pipe_read)
{
   int consumed, n;
   while (true)
   {
      n = fscanf(pipe_read, "%d", &consumed);
      if (n == 1)
         printf("Consumer: Consumed Item: %d\n", consumed);
      else
         break;
   }
```

```
    fclose(pipe_read);
    exit(0);
}
int main()
{
    int file_descriptor[2];
    if (pipe(file_descriptor) < 0)
        exit(1);
    FILE *pipe_read = fdopen(file_descriptor[0], "r");
    FILE *pipe_write = fdopen(file_descriptor[1], "w");
    pid_t producer_pid = fork();
    if (producer_pid == 0)
    {
        fclose(pipe_read);
        producer(pipe_write);
    }
    pid_t consumer_pid = fork();
    if (consumer_pid == 0)
    {
        fclose(pipe_write);
        consumer(pipe_read);
    }
    fclose(pipe_read);
    fclose(pipe_write);
    wait(NULL);
    wait(NULL);
    return 0;
}
```


Activities
Terminal
Aug 21 23:22
tanya@tanya-VirtualBox: ~/PES1UG19EC326/OS_LAB_7
tanya@tanya-Virtual...
tanya@tanya-Virtual...
tanya@tanya-Virtual...
tanya@tanya-VirtualBox:~/PES1UG19EC326/OS_LAB_7$ gcc -pthread week9.c
tanya@tanya-VirtualBox:~/PES1UG19EC326/OS_LAB_7$ ./a.out
Producer: Produced Item: 1
Producer: Produced Item: 2
Producer: Produced Item: 3
Producer: Produced Item: 4
Producer: Produced Item: 5
Consumer: Consumed Item: 1
Consumer: Consumed Item: 2
Consumer: Consumed Item: 3
Consumer: Consumed Item: 4
Consumer: Consumed Item: 5
tanya@tanya-VirtualBox:~/PES1UG19EC326/OS_LAB_7$