

Name: Aditya Somani

Roll No: T1851061

Div: A

PRN NO. 71901204L

ASSIGNMENT NO. 4

TITLE: Thread synchronization using counting semaphores.

Program:

```
#include<stdio.h>

#include<pthread.h>    //Thread library
#include<semaphore.h> //Headerfileforsemaphore

#define BUFF_SIZE 5

typedef int buffer_t;    //internalBuffer
buffer_t buff[BUFF_SIZE];
int bindex;

pthread_mutex_t buff_mutex; //Mutex for buffer

sem_t full;    //When 0 buffer isfull
sem_t empty;    //When 0 buffer isempty

/*==Inserting item into the buffer==*/

void insert_item(buffer_t item)
{
    if(bindex<BUFF_SIZE)
        buff[bindex++]=item;
```

else

```
printf("\nBuffer is full...!!!");
```

```
}
```

```
/*==Removing item from the buffer==*/
```

```
buffer_remove_item()
```

```
{
```

```
    if(bindex>0)
```

```
        return(buff[--bindex]);
```

```
    else
```

```
        printf("\nBuffer is Empty ...!!!!");
```

```
}
```

```
/*==Check if buffer is empty or not==*/
```

```
int isempty()
```

```
{
```

```
    if(bindex==0)
```

```
        return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

```
/*==Check if buffer is full or not==*/
```

```
int isfull()
```

```
{
```

```

        if(bindex==BUFF_SIZE)
            return 1;
        else
            return 0;
    }

/*===Producer will produce the elements===*/

void *producer(void *para)
{
    int *thread=(int *)para;
    buffer_tval;
    int i=0;
    sleep(rand()%10); //sleep for random time
    val=rand()%100;      //Produce a randomvalue
    pthread_mutex_lock(&buff_mutex);//lock
    do
    {
        pthread_mutex_unlock(&buff_mutex);//unlock
        sem_wait(&full);      //Decrement
        pthread_mutex_lock(&buff_mutex);//lock
    }while(isfull());
    insert_item(val); //insert item into the buffer
    pthread_mutex_unlock(&buff_mutex);    //unlock
    sem_post(&empty);    //release semaphore
    printf("\n\tProducer %d produces%d\n",*thread,val);
}

```

```

        i++;

        pthread_exit(0);
    }

/*===Consumer will Consumes the items===*/

void *consumer(void *ptr)
{
    int *thread=(int *)ptr;
    buffer_tval;
    int i=0;

    pthread_mutex_lock(&buff_mutex);
    do
    {
        pthread_mutex_unlock(&buff_mutex);
        sem_wait(&empty);    //increment
        pthread_mutex_lock(&buff_mutex);
    }while(isempty());
    val=remove_item();    //remove item frombuffer
    pthread_mutex_unlock(&buff_mutex);
    sem_post(&full);    //releasesemaphore
    printf("\tConsumer %d Consumed %d from buffer\n",*thread,val);
    i++;

    pthread_exit(0);
}

```

```
}
```

```
int main(int argc,char *argv[])
```

```
{
```

```
    pthread_mutex_init(&buff_mutex,NULL);    //initializemutex
```

```
    sem_init(&full,0,BUFF_SIZE);    //initialize semaphore
```

```
    sem_init(&empty,0,0);
```

```
    bindex=0;
```

```
    pthread_t tid1[10],tid2[10];
```

```
    int i;
```

```
    int num[5];
```

```
    // Create Multiple threads
```

```
    for(i=0;i<5;i++)
```

```
    {    num[i]=i;
```

```
        pthread_create(&tid1[i],NULL,&producer,&num[i]);
```

```
        pthread_create(&tid2[i],NULL,&consumer,&num[i]);
```

```
    }
```

```
    // Join threads
```

```
    for(i=0;i<5;i++)
```

```
    {
```

```
        pthread_join(tid1[i],NULL);
```

```
        pthread_join(tid2[i],NULL);
```

```
    }
```

```

// Destroying all semaphores & mutex
pthread_mutex_destroy(&buff_mutex);

sem_destroy(&full);

sem_destroy(&empty);

return 0;
}

```

Output :

The screenshot shows the OnlineGDB interface with the following output in the console:

```

main.c:67:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
main.c:67:8: warning: implicit declaration of function 'rand' [-Wimplicit-function-declaration]
Consumer 0 Consumed 35 from buffer

Producer 4 produces 86
Consumer 4 Consumed 86 from buffer

Producer 0 produces 35

Producer 1 produces 92
Consumer 3 Consumed 92 from buffer

Producer 3 produces 49
Consumer 2 Consumed 49 from buffer

Producer 2 produces 21
Consumer 1 Consumed 21 from buffer

...Program finished with exit code 0
Press ENTER to exit console.

```