



**NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES
(FAST-NUCES)**

OPERATING SYSTEM PROJECT REPORT (2021)

Group members: Ashmal Anis (19k-0305)

Hasnain Somani (19k-0204)

Project name: System Call for Producer Consumer Problem

Objective

The objective behind this project was to make a practical implementation of the producer consumer problem using bounded buffers, in C language, using the Linux operating system (Ubuntu version). The project was implemented, and results were printed on console screen through a system call.

Project Description

Processes communicate through various sources of communication- a buffer with a limited number of data slots is one of the majorly used source of communication. In the project, producer writes to one end of the buffer, and the consumer reads from the other end. Using semaphores, various test cases have been taken care of- which include prohibiting writing to a full buffer, or reading from an empty buffer. The count is kept through wait() and signal() function calls. Data will be written by a producer, and a specific consumer will be reading the data off. The data written, read, as well as the producer and consumer numbers have been displayed in the program to ensure correct input and output of data.

This output is displayed through the printf() function in the system call made inside the program file. This system call has been made by making amendments to makefile, creating a new C file for the system call. The system call is invoked by creating a new directory named "ProducerConsumer", and then creating a file inside the directory to run the c program file, which then calls the system call in it.

Code results

Code:

```
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <sys/syscall.h>
#include <linux/kernel.h>

#define MaxItems 5 // Maximum items a producer can produce or a consumer can consume
#define BufferSize 5 // Size of the buffer

sem_t empty;
```

```

sem_t full;
int in = 0;
int out = 0;
int buffer[BufferSize];
pthread_mutex_t mutex;

struct data{
    int _id;
    int input;
};

void *producer(void *pno)
{
    int item;
    for(int i = 0; i < MaxItems; i++) {
        struct data d;
        d = *((struct data*)pno);
        item = d.input;

        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;

        syscall(333,"Producer",d._id, "writes", buffer[in], in);
        in = (in+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
}

void *consumer(void *cno)
{
    for(int i = 0; i < MaxItems; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];

        struct data d;
        d = *((struct data*)cno);
    }
}

```

```

        syscall(333,"Consumer",d._id, "reads", item, out);
out = (out+1)%BufferSize;
pthread_mutex_unlock(&mutex);
sem_post(&empty);
    }
}

int main()
{

    pthread_t pro[5],con[5];
    pthread_mutex_init(&mutex, NULL);
    sem_init(&empty,0,BufferSize);
    sem_init(&full,0,0);

    struct data d[5];
    int i;

    printf("Enter 5 data : ");
    for(i=0; i<5; i++){
        scanf("%d",&d[i].input);
        d[i]._id= i+1;
    }

    for(int i = 0; i < 5; i++) {
        pthread_create(&pro[i], NULL, (void *)producer, (void *)&d[i]);
    }
    for(int i = 0; i < 5; i++) {
        pthread_create(&con[i], NULL, (void *)consumer, (void *)&d[i]);
    }

    for(int i = 0; i < 5; i++) {
        pthread_join(pro[i], NULL);
    }
    for(int i = 0; i < 5; i++) {
        pthread_join(con[i], NULL);
    }
}

```

```

}

pthread_mutex_destroy(&mutex);
sem_destroy(&empty);
sem_destroy(&full);

return 0;

}

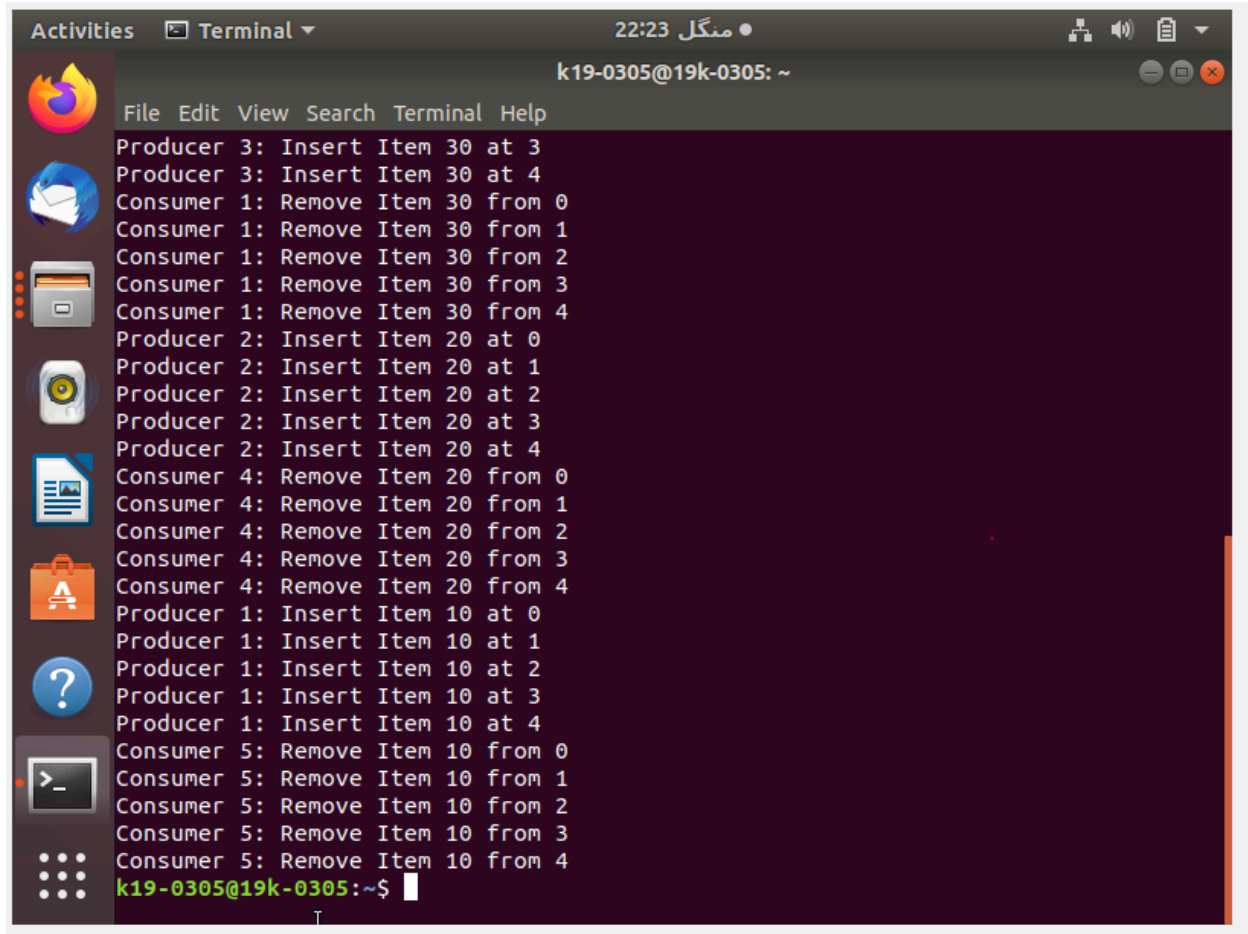
```

Results:

```

k19-0305@19k-0305: ~
22:22 مڭل
File Edit View Search Terminal Help
k19-0305@19k-0305:~$ gedit a.c
k19-0305@19k-0305:~$ gcc -o a a.c -lpthread
k19-0305@19k-0305:~$ ./a.c
bash: ./a.c: Permission denied
k19-0305@19k-0305:~$ ./a
Enter 5 data : 10 20 30 40 50
Producer 5: Insert Item 50 at 0
Producer 5: Insert Item 50 at 1
Producer 5: Insert Item 50 at 2
Producer 5: Insert Item 50 at 3
Producer 5: Insert Item 50 at 4
Consumer 2: Remove Item 50 from 0
Consumer 2: Remove Item 50 from 1
Consumer 2: Remove Item 50 from 2
Consumer 2: Remove Item 50 from 3
Consumer 2: Remove Item 50 from 4
Producer 4: Insert Item 40 at 0
Producer 4: Insert Item 40 at 1
Producer 4: Insert Item 40 at 2
Producer 4: Insert Item 40 at 3
Producer 4: Insert Item 40 at 4
Consumer 3: Remove Item 40 from 0
Consumer 3: Remove Item 40 from 1
Consumer 3: Remove Item 40 from 2
Consumer 3: Remove Item 40 from 3
Consumer 3: Remove Item 40 from 4
Producer 3: Insert Item 30 at 0
Producer 3: Insert Item 30 at 1
Producer 3: Insert Item 30 at 2

```



```
Activities  Terminal  22:23 منگل •
k19-0305@19k-0305: ~
File Edit View Search Terminal Help
Producer 3: Insert Item 30 at 3
Producer 3: Insert Item 30 at 4
Consumer 1: Remove Item 30 from 0
Consumer 1: Remove Item 30 from 1
Consumer 1: Remove Item 30 from 2
Consumer 1: Remove Item 30 from 3
Consumer 1: Remove Item 30 from 4
Producer 2: Insert Item 20 at 0
Producer 2: Insert Item 20 at 1
Producer 2: Insert Item 20 at 2
Producer 2: Insert Item 20 at 3
Producer 2: Insert Item 20 at 4
Consumer 4: Remove Item 20 from 0
Consumer 4: Remove Item 20 from 1
Consumer 4: Remove Item 20 from 2
Consumer 4: Remove Item 20 from 3
Consumer 4: Remove Item 20 from 4
Producer 1: Insert Item 10 at 0
Producer 1: Insert Item 10 at 1
Producer 1: Insert Item 10 at 2
Producer 1: Insert Item 10 at 3
Producer 1: Insert Item 10 at 4
Consumer 5: Remove Item 10 from 0
Consumer 5: Remove Item 10 from 1
Consumer 5: Remove Item 10 from 2
Consumer 5: Remove Item 10 from 3
Consumer 5: Remove Item 10 from 4
k19-0305@19k-0305: ~$
```

Conclusion

Bounded buffers are one of the most efficient approaches to read and write data through processes. This is because it limits the data to be stored, and keeps a count of the empty and filled slots in the buffer using a counting semaphore. This easily accommodates a solution to the producer consumer problem, and prevents data inefficiency.