Operating System Assignment 2

Name: Sayantan Biswas

Class: BCSE 3rd year 1st sem

Roll: 001910501057

Group: A2

Problem Statement: Write a program for *p*-producer *c*-consumer problem, *p, c* >= 1. A shared circular buffer that can hold 50 items is to be used. Each producer process can store any number between 1 to 100 (along with the producer id) and deposit in the buffer. Each consumer process reads a number from the buffer and adds it to a shared variable TOTAL (initialized to 0). Though any consumer process can read any of the numbers in the buffer, the only constraint being that every number written by some producer should be read exactly once by exactly one of the consumers. The program reads in the value of *p* and *c* from the user, and forks *p* producers and *c* consumers. After all the producers and consumers have finished (the consumers exit after all the data produced by all producers have been read), the parent process prints the value of TOTAL. Test the program with different values of *p* and *c*.

DESIGN:

The implementation has been done with the help of ***semaphores, fork and mmap.*** The code is commented and written in C language.

Input Format: value of producer and consumer as mentioned in the problem statement.
Output Format: Producer id, Consumer id, a live buffer and the value of TOTAL(mentioned in the problem).

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>
#include <time.h>
#include <sys/types.h>
#include <sys/mman.h>   //memory management declarations
#include <sys/wait.h>

int N = 50; // Buffer size
int *x;

// semaphores declaration
sem_t *left; //producer count
sem_t *mutex;
sem_t *empty;
sem_t *full;
int *in;
int *out;
//int *tl;
int *total; //TOTAL
//int y=*out;
void produce() //producer part
{
    //printf("\n");
    sem_wait(empty);
    sem_wait(mutex);

    //this part is just for randomizing
    long s = time(NULL);
    s = (s*11)%100;
    s = (s*s);

    x[*in] = s%100 + 1;
    printf("Producer%3d Produced:%3d  Buffer-> ", *in, x[*in]);
    *in = (*in+1)%N;

    for(int i = 0; i < N; i++)
```

```
    {
        if(x[i] == -1) printf("__ ");
        else printf("%2d ", x[i]);
    }printf("\n\n");

    sleep(1);

    sem_post(mutex);
    sem_post(full);
}

void consume() //consumer part
{
    //if(sem_trywait(left) == -1) return;
    sem_wait(full);
    sem_wait(mutex);

    // int y=*out;
    *total = *total + x[*out];
    printf("Consumer%3d Consumed:%3d  Buffer-> ",*out, x[*out]);
    x[*out] = -1;
    *out = (*out+1)%N;

    for(int i = 0; i < N; i++)
    {
        if(x[i] == -1) printf("__ ");
        else printf("%2d ", x[i]);
    }
    printf("TOTAL = %d\n\n", *total);
    //sem_getvalue(left, tl);
    //printf("left=%d\n\n", *tl);

    sleep(1);

    sem_post(mutex);
    sem_post(empty);
}

int main()
{
    printf("\n\n");
    srand(time(0));

    int P,C,i;
    printf("value of p: ");
```

```
    scanf("%d",&P);
    printf("value of c: ");
    scanf("%d",&C);

    int pid = getpid(),wpid;  //pid = process id

    //mapping in virtual address space
      x  =  mmap(NULL,  N*sizeof(int),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
     mutex  =  mmap(NULL,  sizeof(*mutex),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
     empty  =  mmap(NULL,  sizeof(*empty),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
     full  =  mmap(NULL,  sizeof(*full),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
     left  =  mmap(NULL,  sizeof(*left),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
      in  =  mmap(NULL,  sizeof(*in),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
      out  =  mmap(NULL,  sizeof(*out),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
     total  =  mmap(NULL,  sizeof(*total),  PROT_READ  |  PROT_WRITE,  MAP_SHARED  |
MAP_ANONYMOUS, -1, 0);
    //int y=*out;

    // semaphores initialization
    sem_init(mutex, 1, 1);
    sem_init(empty, 1, N);
    sem_init(full, 1, 0);
    sem_init(left, 1, P); // Producer count
    *in = 0; *out = 0; *total = 0;

    for(i=0;i<N;i++) x[i] = -1;

    for(i=1; i<=P; i++) if(pid != 0)
    {
        pid = fork();
    } // producer
    if(pid == 0){produce(); return 0;} // producer exit

    for(i=1; i<=C; i++) {if(pid != 0) pid = fork();} // consumer

    while(pid == 0 && (sem_trywait(left) != -1))
    {
        consume();
```

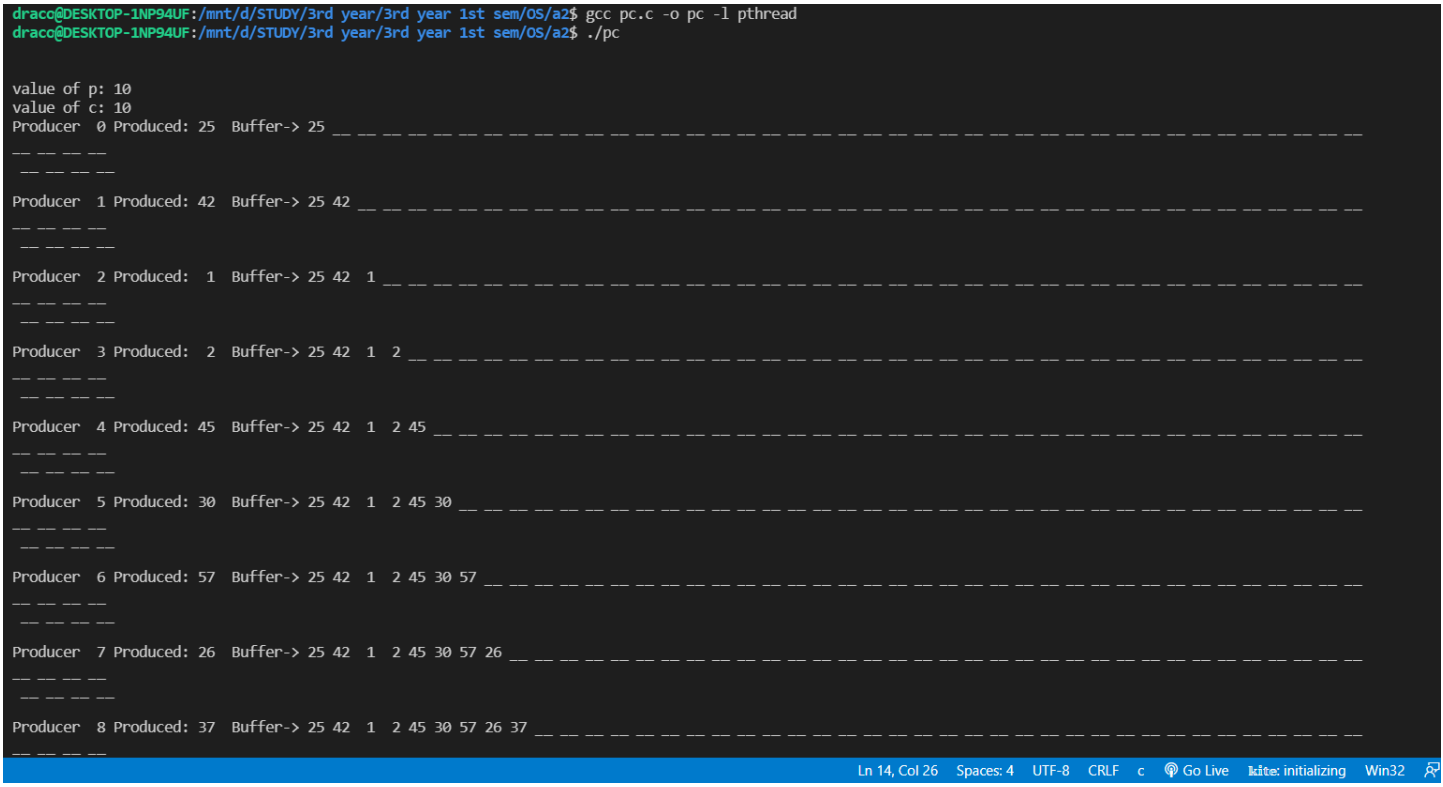
```
    }
    if(pid == 0) return 0; // comsumer exit

    while ((wpid = wait(NULL)) > 0);
    printf("\n\n");

    return 0;
}
```

OUTPUT:

Output screenshots->



```
Producer  9 Produced: 90  Buffer-> 25 42  1  2 45 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __

Consumer  0 Consumed: 25  Buffer-> __ 42  1  2 45 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 25

Consumer  1 Consumed: 42  Buffer-> __ __  1  2 45 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 67

Consumer  2 Consumed:  1  Buffer-> __ __ __  2 45 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 68

Consumer  3 Consumed:  2  Buffer-> __ __ __ __ __ 45 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 70

Consumer  4 Consumed: 45  Buffer-> __ __ __ __ __ __ 30 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 115

Consumer  5 Consumed: 30  Buffer-> __ __ __ __ __ __ __ 57 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 145

Consumer  6 Consumed: 57  Buffer-> __ __ __ __ __ __ __ __ 26 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 202

Consumer  7 Consumed: 26  Buffer-> __ __ __ __ __ __ __ __ __ 37 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 228

Consumer  8 Consumed: 37  Buffer-> __ __ __ __ __ __ __ __ __ __ 90 __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 265

Consumer  9 Consumed: 90  Buffer-> __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __ __
 __ __ __ __ TOTAL = 355
```
