CS266

ASSIGNMENT 7

NAME:

ARCHIT AGRAWAL

ROLL NO. :

202051213

SECTION:

2

Question

1 Problem 1

Design a programming solution to the bounded-buffer problem using the producer and consumer processes using three semaphores: empty and full, which count the number of empty and full slots in the buffer, and mutex, which is a binary (or mutualexclusion) semaphore that protects the actual insertion or removal of items in the buffer. For this project, you will use standard counting semaphores for empty and full and a mutex lock, rather than a binary semaphore, to represent mutex. The producer and consumer—running as separate threads—will move items to and from a buffer that is synchronized with the empty, full, and mutex structures. You can solve this problem using Pthreads.

Refer to detailed instruction and few sample codes in Programming Project-3, Chapter 5 in Operating System Concepts 9th Edition.

Code

```c
#include <pthread.h>
#include <semaphore.h>
#include <stdlib.h>
#include <stdio.h>

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

/*
This program provides a possible solution for producer-consumer problem using mutex and semaphore.
*/

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

void *producer(void *pno){
    int item;
    for(int i = 0; i < MaxItems; i++) {
        item = rand(); // Produce an random item
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = item;
```

```
        printf("Producer %d: Insert Item %d at %d\n", *((int
*)pno),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];
        printf("Consumer %d: Remove Item %d from %d\n",*((int *)cno),item,
out);
        out = (out+1)%BufferSize;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
}

int main(){
    int n;


    printf("Enter n (number of producers and consumers) : ");
    scanf("%d", &n);


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

    int a[n];
    for(int i = 0; i < n; i++){
        a[i] = i + 1;
    }

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

    for(int i = 0; i < n; i++) {
```

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

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

    return 0;

}
```

Output 1

```
archit@archit-VirtualBox:~$ cd Desktop/
archit@archit-VirtualBox:~/Desktop$ gcc boundedBuffer.c -o boundedBuffer -lpthread
archit@archit-VirtualBox:~/Desktop$ ./boundedBuffer
Enter n (number of producers and consumers) : 5
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 846930886 at 1
Producer 1: Insert Item 1681692777 at 2
Producer 1: Insert Item 1714636915 at 3
Producer 1: Insert Item 1957747793 at 4
Consumer 5: Remove Item 1804289383 from 0
Consumer 5: Remove Item 846930886 from 1
Consumer 5: Remove Item 1681692777 from 2
Producer 4: Insert Item 424238335 at 0
Producer 4: Insert Item 1189641421 at 1
Consumer 4: Remove Item 1714636915 from 3
Consumer 4: Remove Item 1957747793 from 4
Producer 4: Insert Item 1025202362 at 2
Producer 4: Insert Item 1350490027 at 3
Consumer 4: Remove Item 424238335 from 0
Consumer 4: Remove Item 1189641421 from 1
Producer 4: Insert Item 783368690 at 4
Producer 5: Insert Item 596516649 at 0
Producer 5: Insert Item 1102520059 at 1
Consumer 2: Remove Item 1025202362 from 2
Consumer 2: Remove Item 1350490027 from 3
Consumer 2: Remove Item 783368690 from 4
Consumer 1: Remove Item 596516649 from 0
Producer 5: Insert Item 2044897763 at 2
Producer 3: Insert Item 719885386 at 3
Producer 3: Insert Item 1365188540 at 4
Consumer 4: Remove Item 1102520059 from 1
Producer 3: Insert Item 1540383426 at 0
Consumer 5: Remove Item 2044897763 from 2
Consumer 5: Remove Item 719885386 from 3
Consumer 1: Remove Item 1365188540 from 4
Consumer 2: Remove Item 1540383426 from 0
Producer 5: Insert Item 1967513926 at 1
Producer 5: Insert Item 1303455736 at 2
Producer 3: Insert Item 304089172 at 3
Consumer 2: Remove Item 1967513926 from 1
Producer 2: Insert Item 1649760492 at 4
Producer 2: Insert Item 521595368 at 0
```

```
Producer 3: Insert Item 35005211 at 1
Consumer 3: Remove Item 1303455736 from 2
Consumer 3: Remove Item 304089172 from 3
Consumer 3: Remove Item 1649760492 from 4
Consumer 3: Remove Item 521595368 from 0
Consumer 1: Remove Item 35005211 from 1
Producer 2: Insert Item 294702567 at 2
Producer 2: Insert Item 1726956429 at 3
Producer 2: Insert Item 336465782 at 4
Consumer 1: Remove Item 294702567 from 2
Consumer 1: Remove Item 1726956429 from 3
Consumer 3: Remove Item 336465782 from 4
archit@archit-VirtualBox:~/Desktop$
```

Output 2

```
archit@archit-VirtualBox:~$ cd Desktop/
archit@archit-VirtualBox:~/Desktop$ ./boundedBuffer
Enter n (number of producers and consumers) : 1
Producer 1: Insert Item 1804289383 at 0
Producer 1: Insert Item 846930886 at 1
Producer 1: Insert Item 1681692777 at 2
Producer 1: Insert Item 1714636915 at 3
Producer 1: Insert Item 1957747793 at 4
Consumer 1: Remove Item 1804289383 from 0
Consumer 1: Remove Item 846930886 from 1
Consumer 1: Remove Item 1681692777 from 2
Consumer 1: Remove Item 1714636915 from 3
Consumer 1: Remove Item 1957747793 from 4
archit@archit-VirtualBox:~/Desktop$
```