CSB 302: Operating System

Lab5: Process Synchronization using Semaphore

Submitted By:

Name: **UMANG KUMAR**

Roll No: **201210051**

Branch: **CSE**

Semester: **5th Sem**

Submitted To: ***Dr. Rishav Singh***

Release Date: 19/09/2022 **Submitted Date:** 21/09/2022

NATIONAL INSTITUTE OF TECHNOLOGY DELHI

<u>Department of Computer Science and Engineering</u>

2022

Q.1 Write a program in C to implement the Producer Consumer problem using semaphores?

CODE:

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

#define Max 5          // Maximum items a producer can produce or a consumer can
#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;
void *producer(void *pno) {
   int item;
   for (int i = 0; i < Max; 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 < Max; 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);
```

```c
    }
}
int main() {
   pthread_t pro[5], con[5];
   pthread_mutex_init(&mutex, NULL);
   sem_init(&empty, 0, BufferSize);
   sem_init(&full, 0, 0);
   int a[5] = {1, 2, 3, 4,
               5};  // Just used for numbering the producer and consumer
   for (int i = 0; i < 5; i++) {
       pthread_create(&pro[i], NULL, (void *)producer, (void *)&a[i]);
   }
   for (int i = 0; i < 5; i++) {
       pthread_create(&con[i], NULL, (void *)consumer, (void *)&a[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;
}
```

OUTPUT:

```
Producer 2: Insert Item 16807 at 0
Producer 2: Insert Item 984943658 at 1
Producer 2: Insert Item 1144108930 at 2
Producer 4: Insert Item 282475249 at 3
Consumer 5: Remove Item 16807 from 0
Consumer 5: Remove Item 984943658 from 1
Consumer 5: Remove Item 1144108930 from 2
Consumer 5: Remove Item 282475249 from 3
Consumer 5: Remove Item 0 from 4
Consumer 1: Remove Item 16807 from 0
Consumer 1: Remove Item 984943658 from 1
Consumer 1: Remove Item 1144108930 from 2
Consumer 1: Remove Item 282475249 from 3
Consumer 1: Remove Item 0 from 4
Producer 2: Insert Item 470211272 at 4
Producer 5: Insert Item 1622650073 at 0
Producer 5: Insert Item 1458777923 at 1
Consumer 3: Remove Item 1622650073 from 0
Consumer 3: Remove Item 1458777923 from 1
Consumer 3: Remove Item 1144108930 from 2
Producer 3: Insert Item 282475249 at 2
Producer 3: Insert Item 823564440 at 3
Producer 1: Insert Item 16807 at 4
Consumer 4: Remove Item 823564440 from 3
Producer 5: Insert Item 2007237709 at 0
Producer 5: Insert Item 74243042 at 1
Consumer 3: Remove Item 16807 from 4
Producer 4: Insert Item 101027544 at 2
Producer 4: Insert Item 1137522503 at 3
Producer 4: Insert Item 1441282327 at 4
Producer 4: Insert Item 16531729 at 0
Consumer 4: Remove Item 16531729 from 0
Consumer 2: Remove Item 74243042 from 1
Consumer 2: Remove Item 101027544 from 2
Consumer 2: Remove Item 1137522503 from 3
Consumer 3: Remove Item 1441282327 from 4
Producer 3: Insert Item 1115438165 at 1
Producer 3: Insert Item 823378840 at 2
Consumer 4: Remove Item 16531729 from 0
Producer 5: Insert Item 114807987 at 3
Consumer 2: Remove Item 1115438165 from 1
Consumer 2: Remove Item 823378840 from 2
Producer 1: Insert Item 1784484492 at 4
Producer 1: Insert Item 896544303 at 0
Consumer 4: Remove Item 114807987 from 3
Producer 1: Insert Item 1474833169 at 1
Producer 3: Insert Item 143542612 at 2
Consumer 4: Remove Item 1784484492 from 4
Producer 1: Insert Item 1264817709 at 3
Producer 2: Insert Item 1457850878 at 4
```

Q.2 Write a program in C to implement Readers-Writers problem using semaphores?

CODE:

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

sem_t wrt;
pthread_mutex_t mutex;
int cnt = 1;
int numreader = 0;
void *writer(void *wno) {
   sem_wait(&wrt);
   cnt = cnt * 2;
   printf("Writer %d modified cnt to %d\n", (*((int *)wno)), cnt);
   sem_post(&wrt);
}
void *reader(void *rno) {
   pthread_mutex_lock(&mutex);
   numreader++;
   if (numreader == 1) {
       sem_wait(&wrt);
   }
   pthread_mutex_unlock(&mutex);
   printf("Reader %d: read cnt as %d\n", *((int *)rno), cnt);
   pthread_mutex_lock(&mutex);
   numreader--;
   if (numreader == 0) {
       sem_post(&wrt);
   }

   pthread_mutex_unlock(&mutex);
}
int main() {
   pthread_t read[5], write[5];
   pthread_mutex_init(&mutex, NULL);
   sem_init(&wrt, 0, 1);
   int a[5] = {1, 2, 3, 4, 5};

   for (int i = 0; i < 10; i++) {
       pthread_create(&read[i], NULL, (void *)reader, (void *)&a[i]);
   }
   for (int i = 0; i < 5; i++) {
       pthread_create(&write[i], NULL, (void *)writer, (void *)&a[i]);
   }
   for (int i = 0; i < 5; i++) {
       pthread_join(read[i], NULL);
   }

   for (int i = 0; i < 5; i++) {
       pthread_join(write[i], NULL);
   }
   pthread_mutex_destroy(&mutex);
   sem_destroy(&wrt);
   return 0;
}
```

Output:

```
Reader 1: read cnt as 1
Reader -1248053576: read cnt as 1
Reader 3: read cnt as 1
Reader 2: read cnt as 1
Reader 1: read cnt as 1
Reader 4: read cnt as 1
Reader 5: read cnt as 1
Reader 32759: read cnt as 1
Reader 209551360: read cnt as 1
Reader 1: read cnt as 1
Writer 1 modified cnt to 2
Writer 2 modified cnt to 4
Writer 5 modified cnt to 32
Writer 3 modified cnt to 8
Writer 4 modified cnt to 16
```