Ex. No. 8

PRODUCER CONSUMER USING SEMAPHORES

Aim: To write a program to implement solution to producer consumer problem using semaphores.

Algorithm:

1. Initialize semaphore empty, full and mutex.
2. Create two threads- producer thread and consumer thread.
3. Wait for target thread termination.
4. Call sem_wait on empty semaphore followed by mutex semaphore before entry into critical section.
5. Produce/Consume the item in critical section.
6. Call sem_post on mutex semaphore followed by full semaphore
7. before exiting critical section.
8. Allow the other thread to enter its critical section.
9. Terminate after looping ten times in producer and consumer Threads each.

Program Code:

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

#define MAX_ITEMS 10

sem_t empty, full, mutex;
int buffer[MAX_ITEMS];
int in = 0, out = 0;

void *producer(void *param) {
    int item;
    for (int i = 0; i < 10; i++) {
        item = rand() % 100;  // Produce a random item
        sem_wait(&empty);  // Decrease empty semaphore
        sem_wait(&mutex);  // Enter critical section

        // Add the item to the buffer
        buffer[in] = item;
        printf("Producer produced item: %d\n", item);
        in = (in + 1) % MAX_ITEMS;  // Circular buffer

        sem_post(&mutex);  // Exit critical section
        sem_post(&full);   // Increase full semaphore
    }
    return NULL;
}

void *consumer(void *param) {
    int item;
    for (int i = 0; i < 10; i++) {
        sem_wait(&full);  // Decrease full semaphore
        sem_wait(&mutex); // Enter critical section

        // Consume an item from the buffer
        item = buffer[out];
        printf("Consumer consumed item: %d\n", item);
        out = (out + 1) % MAX_ITEMS;  // Circular buffer

        sem_post(&mutex);  // Exit critical section
        sem_post(&empty);  // Increase empty semaphore
    }
    return NULL;
```

```c
int main() {
    pthread_t prod_thread, cons_thread;

    // Initialize semaphores
    sem_init(&empty, 0, MAX_ITEMS);  // Initially, buffer is empty
    sem_init(&full, 0, 0);           // Initially, no items are full
    sem_init(&mutex, 0, 1);          // Mutex for critical section (1 means unlocked)

    // Create producer and consumer threads
    pthread_create(&prod_thread, NULL, producer, NULL);
    pthread_create(&cons_thread, NULL, consumer, NULL);

    // Wait for threads to finish
    pthread_join(prod_thread, NULL);
    pthread_join(cons_thread, NULL);

    // Destroy semaphores
    sem_destroy(&empty);
    sem_destroy(&full);
    sem_destroy(&mutex);

    return 0;
}
```

Output:

```
[cse46@localhost ~]$ vi producer_consumer.c
[cse46@localhost ~]$ vi run_producer_consumer.sh
[cse46@localhost ~]$ chmod +x run_producer_consumer.sh
[cse46@localhost ~]$ ./run_producer_consumer.sh
Producer produced item: 83
Producer produced item: 86
Producer produced item: 77
Producer produced item: 15
Producer produced item: 93
Producer produced item: 35
Producer produced item: 86
Producer produced item: 92
Producer produced item: 49
Producer produced item: 21
Consumer consumed item: 83
Consumer consumed item: 86
Consumer consumed item: 77
Consumer consumed item: 15
Consumer consumed item: 93
Consumer consumed item: 35
Consumer consumed item: 86
Consumer consumed item: 92
Consumer consumed item: 49
Consumer consumed item: 21
```