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:

#include <stdio.h> #include <stdlib.h> #include <semaphore.h> #include <unistd.h>

#define BUFFER_SIZE 5 #define MAX_ITEMS 10

sem_t empty, full, mutex; int buffer[BUFFER_SIZE]; int in = 0, out = 0; // Indices for the producer and consumer

// Function to produce an item void produce() { sem_wait(&empty); // Wait for an empty slot in the buffer sem_wait(&mutex); // Enter critical section

// Produce an item (for simplicity, we just print a message)
buffer[in] = in + 1; // Produce item (just use the index + 1 as item)
printf("Produced item: %d\n", buffer[in]);
in = (in + 1) % BUFFER_SIZE; // Circular buffer logic

sem_post(&mutex);   // Exit critical section
sem_post(&full);   // Signal that an item is produced

}

// Function to consume an item void consume() { sem_wait(&full); // Wait for a full slot in the buffer sem_wait(&mutex); // Enter critical section

// Consume an item (for simplicity, we just print a message)
int item = buffer[out];
printf("Consumed item: %d\n", item);
out = (out + 1) % BUFFER_SIZE; // Circular buffer logic

sem_post(&mutex);   // Exit critical section
sem_post(&empty);   // Signal that an item is consumed

}

int main() { // Initialize semaphores sem_init(&empty, 0, BUFFER_SIZE); // Initially, all slots are empty sem_init(&full, 0, 0); // Initially, no slots are full sem_init(&mutex, 0, 1); // Mutex for critical section (initial value 1)

// Run the producer-consumer process 10 times each
for (int i = 0; i < MAX_ITEMS; i++) {

```
    produce();
    consume();
    sleep(1);  // Sleep to simulate the time taken by producer and consumer
}

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

return 0;


}
```

Output:

Produced item: 1

Consumed item: 1

Produced item: 2

Consumed item: 2

Produced item: 3

Consumed item: 3

Produced item: 4

Consumed item: 4

Produced item: 5

Consumed item: 5

Produced item: 6

Consumed item: 6

Produced item: 7

Consumed item: 7

Produced item: 8

Consumed item: 8

Produced item: 9

Consumed item: 9

Produced item: 10

Consumed item: 10

Result:

Program to implement solution to producer consumer problem using semaphores is successful.