Progressive Education Society’s

**MODERN COLLEGE OF ENGINEERING, Pune -05.**

(An Autonomous Institute Affiliated to Savitribai Phule Pune University)

MCA Department

**PRACTICAL SUBMISSION RECORD- A.Y. 2024-25**

|  |  |  |  |
| --- | --- | --- | --- |
| **Class: FYMCA Div: A**  **Semester: II** | **Course Code: MCA01554**  **Course Name: Laboratory Practice - II** | | **Batch: F2** |
| **Name: Pranav Raju Malwatkar** | | **Roll No: 51037** | |
| **CO No:** **CO517.1** | | **Assignment No: 3** | |

**Program Title: 3) Write a C-program to test inter process communication by implementing the producer –**

**consumer problem using semaphores.**

**Program Code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define BUFFER\_SIZE 5 // Size of the shared buffer

#define NUM\_ITEMS 10 // Number of items to produce/consume

int buffer[BUFFER\_SIZE]; // Shared buffer

int in = 0, out = 0; // Buffer indices

sem\_t empty, full; // Semaphores for buffer slots

pthread\_mutex\_t mutex; // Mutex for critical section

void \*producer(void \*arg) {

for (int i = 0; i < NUM\_ITEMS; i++) {

int item = rand() % 100; // Generate a random item

sem\_wait(&empty); // Wait for empty slot

pthread\_mutex\_lock(&mutex); // Lock critical section

// Add item to buffer

buffer[in] = item;

printf("Producer produced: %d\n", item);

in = (in + 1) % BUFFER\_SIZE;

pthread\_mutex\_unlock(&mutex); // Unlock critical section

sem\_post(&full); // Signal that buffer has data

sleep(1); // Simulate time taken to produce

}

return NULL;

}

void \*consumer(void \*arg) {

for (int i = 0; i < NUM\_ITEMS; i++) {

sem\_wait(&full); // Wait for available item

pthread\_mutex\_lock(&mutex); // Lock critical section

// Remove item from buffer

int item = buffer[out];

printf("Consumer consumed: %d\n", item);

out = (out + 1) % BUFFER\_SIZE;

pthread\_mutex\_unlock(&mutex); // Unlock critical section

sem\_post(&empty); // Signal that buffer has space

sleep(2); // Simulate time taken to consume

}

return NULL;

}

int main() {

pthread\_t prod, cons;

// Initialize semaphores

sem\_init(&empty, 0, BUFFER\_SIZE);

sem\_init(&full, 0, 0);

pthread\_mutex\_init(&mutex, NULL);

// Create producer and consumer threads

pthread\_create(&prod, NULL, producer, NULL);

pthread\_create(&cons, NULL, consumer, NULL);

// Wait for threads to finish

pthread\_join(prod, NULL);

pthread\_join(cons, NULL);

// Cleanup semaphores and mutex

sem\_destroy(&empty);

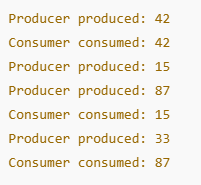
sem\_destroy(&full);

pthread\_mutex\_destroy(&mutex);

return 0;

}

**Output:**

****