Experiment No:6

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

#define BUFFER\_SIZE 5

#define NUM\_ITEMS 10

int buffer[BUFFER\_SIZE];

int count = 0; // Count of items in the buffer

sem\_t empty; // Semaphore to count empty slots sem\_t full; // Semaphore to count full slots

pthread\_mutex\_t mutex; // Mutex for mutual exclusion

void\* producer(void\* arg) {

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

sleep(rand() % 2); // Simulate variable production time

int item = rand() % 100; // Produce a random item sem\_wait(&empty); // Decrement empty count pthread\_mutex\_lock(&mutex); // Enter critical section

// Add item to buffer buffer[count++] = item;

printf("Produced: %d, Buffer Count: %d\n", item, count);

pthread\_mutex\_unlock(&mutex); // Exit critical section sem\_post(&full); // Increment full count

}

return NULL;

}

void\* consumer(void\* arg) {

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

sleep(rand() % 2); // Simulate variable consumption time

sem\_wait(&full); // Decrement full count pthread\_mutex\_lock(&mutex); // Enter critical section

// Remove item from buffer int item = buffer[--count];

printf("Consumed: %d, Buffer Count: %d\n", item, count);

pthread\_mutex\_unlock(&mutex); // Exit critical section sem\_post(&empty); // Increment empty count

}

return NULL;

}

int main() {

pthread\_t prod, cons;

// Initialize semaphores and mutex

sem\_init(&empty, 0, BUFFER\_SIZE); // Initially, all buffer slots are empty sem\_init(&full, 0, 0); // Initially, no slots are full pthread\_mutex\_init(&mutex, NULL); // Initialize mutex

// 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);

// Clean up sem\_destroy(&empty); sem\_destroy(&full); pthread\_mutex\_destroy(&mutex);

return 0;

}

