20. Construct a C program to simulate Reader-Writer problem using Semaphores.

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

#include <semaphore.h>

#include <unistd.h>

#define MAX\_READERS 5

#define MAX\_WRITERS 2

sem\_t mutex; // Semaphore for mutual exclusion

sem\_t writeBlock; // Semaphore for writers

int readCount = 0; // Number of active readers

void\* reader(void\* id) {

int readerID = \*((int\*)id);

// Start reading

sem\_wait(&mutex); // Enter critical section

readCount++;

if (readCount == 1) {

sem\_wait(&writeBlock); // First reader locks the writer

}

sem\_post(&mutex); // Exit critical section

// Simulate reading

printf("Reader %d is reading.\n", readerID);

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

// End reading

sem\_wait(&mutex); // Enter critical section

readCount--;

if (readCount == 0) {

sem\_post(&writeBlock); // Last reader unlocks the writer

}

sem\_post(&mutex); // Exit critical section

return NULL;

}

void\* writer(void\* id) {

int writerID = \*((int\*)id);

// Start writing

sem\_wait(&writeBlock); // Wait for access to write

// Simulate writing

printf("Writer %d is writing.\n", writerID);

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

// End writing

sem\_post(&writeBlock); // Release access to write

return NULL;

}

int main() {

pthread\_t readers[MAX\_READERS], writers[MAX\_WRITERS];

int readerIDs[MAX\_READERS], writerIDs[MAX\_WRITERS];

// Initialize semaphores

if (sem\_init(&mutex, 0, 1) != 0) {

perror("Semaphore initialization failed");

exit(EXIT\_FAILURE);

}

if (sem\_init(&writeBlock, 0, 1) != 0) {

perror("Semaphore initialization failed");

exit(EXIT\_FAILURE);

}

// Create reader threads

for (int i = 0; i < MAX\_READERS; i++) {

readerIDs[i] = i + 1;

if (pthread\_create(&readers[i], NULL, reader, &readerIDs[i]) != 0) {

perror("Failed to create reader thread");

exit(EXIT\_FAILURE);

}

}

// Create writer threads

for (int i = 0; i < MAX\_WRITERS; i++) {

writerIDs[i] = i + 1;

if (pthread\_create(&writers[i], NULL, writer, &writerIDs[i]) != 0) {

perror("Failed to create writer thread");

exit(EXIT\_FAILURE);

}

}

// Wait for all threads to finish

for (int i = 0; i < MAX\_READERS; i++) {

pthread\_join(readers[i], NULL);

}

for (int i = 0; i < MAX\_WRITERS; i++) {

pthread\_join(writers[i], NULL);

}

// Destroy semaphores

sem\_destroy(&mutex);

sem\_destroy(&writeBlock);

return 0;

}

