#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

// Mutex and condition variables

pthread\_mutex\_t mutex;

pthread\_cond\_t writer\_cond;

// Shared resource and counters

int read\_count = 0; // Number of readers accessing the resource

int shared\_data = 0; // Shared resource

// Reader function

void \*reader(void \*arg) {

int reader\_id = \*((int \*)arg);

free(arg);

// Reader process starts

pthread\_mutex\_lock(&mutex);

read\_count++;

// If this is the first reader, lock the writer

if (read\_count == 1) {

pthread\_cond\_wait(&writer\_cond, &mutex);

}

pthread\_mutex\_unlock(&mutex);

// Reading the shared resource

printf("Reader %d is reading the shared data: %d\n", reader\_id, shared\_data);

sleep(1); // Simulating reading time

// Reader process ends

pthread\_mutex\_lock(&mutex);

read\_count--;

// If this is the last reader, signal the writer

if (read\_count == 0) {

pthread\_cond\_signal(&writer\_cond);

}

pthread\_mutex\_unlock(&mutex);

pthread\_exit(NULL);

}

// Writer function

void \*writer(void \*arg) {

int writer\_id = \*((int \*)arg);

free(arg);

// Writer process starts

pthread\_mutex\_lock(&mutex);

// Writing to the shared resource

shared\_data += 1;

printf("Writer %d is writing the shared data: %d\n", writer\_id, shared\_data);

sleep(1); // Simulating writing time

pthread\_mutex\_unlock(&mutex);

// Wake up waiting readers if any

pthread\_cond\_signal(&writer\_cond);

pthread\_exit(NULL);

}

int main() {

pthread\_t readers[5], writers[2];

int i;

// Initialize the mutex and condition variables

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

pthread\_cond\_init(&writer\_cond, NULL);

// Creating reader threads

for (i = 0; i < 5; i++) {

int \*reader\_id = malloc(sizeof(int));

\*reader\_id = i + 1;

pthread\_create(&readers[i], NULL, reader, reader\_id);

}

// Creating writer threads

for (i = 0; i < 2; i++) {

int \*writer\_id = malloc(sizeof(int));

\*writer\_id = i + 1;

pthread\_create(&writers[i], NULL, writer, writer\_id);

}

// Joining all threads

for (i = 0; i < 5; i++) {

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

}

for (i = 0; i < 2; i++) {

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

}

// Destroy mutex and condition variables

pthread\_mutex\_destroy(&mutex);

pthread\_cond\_destroy(&writer\_cond);

return 0;

}

