**Experiment No. 6**

NAME:Omkar Khanvilkar ROLL NO: 07

CLASS: TY\_IT-B BATCH: 2

DATE OF PERFORMANCE: 26/09/2024 PRN No: 12210313

1. **Implementation of Classical problems Reader Writer  using Threads and Mutex**

**Code in c**#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

*pthread\_mutex\_t* mutex, write\_mutex;

*int* read\_count = 0;

*int* shared\_data = 0;

*void*\* reader(*void*\* *arg*) {

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

    pthread\_mutex\_lock(&mutex);

    read\_count++;

    if (read\_count == 1) {

        pthread\_mutex\_lock(&write\_mutex);

    }

    pthread\_mutex\_unlock(&mutex);

    // Reading the shared data

    printf("Reader %d: reads shared data as %d\n", reader\_id, shared\_data);

    pthread\_mutex\_lock(&mutex);

    read\_count--;

    if (read\_count == 0) {

        pthread\_mutex\_unlock(&write\_mutex); .

    }

    pthread\_mutex\_unlock(&mutex);

    return NULL;

}

*void*\* writer(*void*\* *arg*) {

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

    pthread\_mutex\_lock(&write\_mutex);

    // Writing to the shared data

    shared\_data += 10;

    printf("Writer %d: writes shared data as %d\n", writer\_id, shared\_data);

    pthread\_mutex\_unlock(&write\_mutex);

    return NULL;

}

*int* main() {

*int* num\_readers, num\_writers;

    printf("Enter the number of readers: ");

    scanf("%d", &num\_readers);

    printf("Enter the number of writers: ");

    scanf("%d", &num\_writers);

*pthread\_t* readers[num\_readers], writers[num\_writers];

*int* reader\_ids[num\_readers], writer\_ids[num\_writers];

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

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

    // Creating reader threads

    for (*int* i = 0; i < num\_readers; i++) {

        reader\_ids[i] = i + 1;

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

    }

    // Creating writer threads

    for (*int* i = 0; i < num\_writers; i++) {

        writer\_ids[i] = i + 1;

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

    }

    // Joining reader threads

    for (*int* i = 0; i < num\_readers; i++) {

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

    }

    // Joining writer threads

    for (*int* i = 0; i < num\_writers; i++) {

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

    }

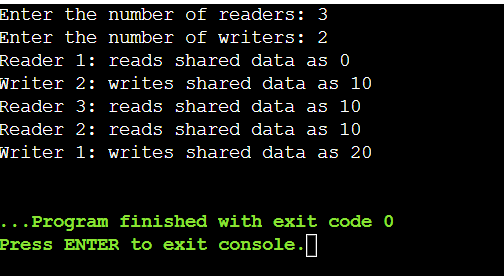
    pthread\_mutex\_destroy(&mutex);

    pthread\_mutex\_destroy(&write\_mutex);

    return 0;

}

**Output :**

****

1. **Implementation of Classical problems Reader Writer  using Threads and Samaphore**

**Code in C :**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

*sem\_t* rw\_mutex, mutex;

*int* read\_count = 0;

*int* shared\_data = 0;

*void*\* reader(*void*\* *arg*) {

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

    sem\_wait(&mutex);

    read\_count++;

    if (read\_count == 1) {

        sem\_wait(&rw\_mutex); // First reader locks the writer.

    }

    sem\_post(&mutex);

    // Reading the shared data

    printf("Reader %d: reads shared data as %d\n", reader\_id, shared\_data);

    sem\_wait(&mutex);

    read\_count--;

    if (read\_count == 0) {

        sem\_post(&rw\_mutex); // Last reader unlocks the writer.

    }

    sem\_post(&mutex);

    return NULL;

}

*void*\* writer(*void*\* *arg*) {

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

    sem\_wait(&rw\_mutex);

    // Writing to the shared data

    shared\_data += 10;

    printf("Writer %d: writes shared data as %d\n", writer\_id, shared\_data);

    sem\_post(&rw\_mutex);

    return NULL;

}

*int* main() {

*int* num\_readers, num\_writers;

    printf("Enter the number of readers: ");

    scanf("%d", &num\_readers);

    printf("Enter the number of writers: ");

    scanf("%d", &num\_writers);

*pthread\_t* readers[num\_readers], writers[num\_writers];

*int* reader\_ids[num\_readers], writer\_ids[num\_writers];

    sem\_init(&rw\_mutex, 0, 1);

    sem\_init(&mutex, 0, 1);

    // Creating reader threads

    for (*int* i = 0; i < num\_readers; i++) {

        reader\_ids[i] = i + 1;

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

    }

    // Creating writer threads

    for (*int* i = 0; i < num\_writers; i++) {

        writer\_ids[i] = i + 1;

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

    }

    // Joining reader threads

    for (*int* i = 0; i < num\_readers; i++) {

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

    }

    // Joining writer threads

    for (*int* i = 0; i < num\_writers; i++) {

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

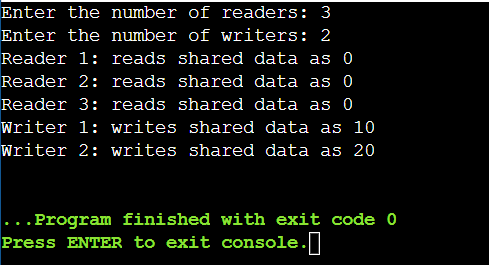
    }

    sem\_destroy(&rw\_mutex);

    sem\_destroy(&mutex);

    return 0;

}

**Output :   
  
**