

Lab Assignment-9

BCSE303P Operating Systems



***Name: Chillara V L N S Pavana Vamsi***

***Register Number: 21BCE5095***

1. **Develop a readers and writers problem with minimum 2 readers and 2 writers, Ensure that synchronisation is done with semaphore and satisfy the 4 below mentioned conditions. Shared data as an integer variable and let the writers do the increment operations and readers do the shared variable read operation.**

* **Use thread functions from pthread.h header file to create pthread which will perform read and write operations.**

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Process 1 | Process 2 | Allowed/Not Allowed |
| Case 1 | **Writing** | **Writing** | **Not Allowed** |
| **Case 2** | **Writing** | **Reading** | **Not Allowed** |
| **Case 3** | **Reading** | **Writing** | **Not Allowed** |
| **Case 4** | **Reading** | **Reading** | **Allowed** |

* **Use functions from semaphore.h header file to do synchronisation to perform r-w operations without any data inconsistency and hence avoid the race conditions.**
* **If thread creation is done one after other.**

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

pthread\_mutex\_t mutex;

sem\_t writeMutex;

int readCount = 0;

int value = 42;

void \*Reader(void \*i)

{

    int \*n = (int \*)i;

    pthread\_mutex\_lock(&mutex);

    readCount++;

    if (readCount > 0)

        sem\_wait(&writeMutex);

    pthread\_mutex\_unlock(&mutex);

    printf("Reader %d reads %d value\n", \*n, value);

    pthread\_mutex\_lock(&mutex);

    readCount--;

    if (readCount == 0)

        sem\_post(&writeMutex);

    pthread\_mutex\_unlock(&mutex);

}

void \*Writer(void \*i)

{

    int \*n = (int \*)i;

    sem\_wait(&writeMutex);

    value = value - 1;

    printf("Writer %d modifies to %d value\n", \*n, value);

    sem\_post(&writeMutex);

}

int main()

{

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

    sem\_init(&writeMutex, 0, 1);

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

    for (int t = 1, i = 0; i < 2; i++, t++)

    {

        pthread\_create(&readers[i], NULL, (void \*)Reader, (void \*)&t);

        sleep(1);

    }

    for (int t = 1, i = 0; i < 2; i++, t++)

    {

        pthread\_create(&writers[i], NULL, (void \*)Writer, (void \*)&t);

        sleep(1);

    }

    for (int i = 0; i < 2; i++)

    {

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

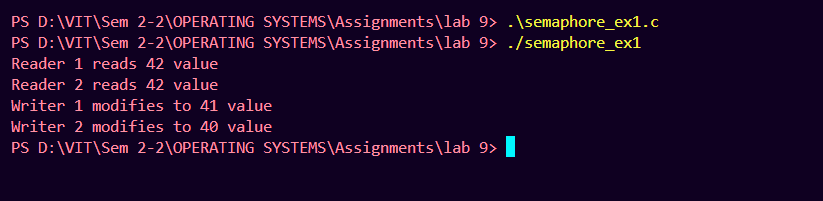
        pthread\_join(writers[i], NULL);

    }

    pthread\_mutex\_destroy(&mutex);

    sem\_destroy(&writeMutex);

    return 0;

}

* **If thread creation is parallelly done**

    for (int t = 1, i = 0; i < 2; i++, t++)

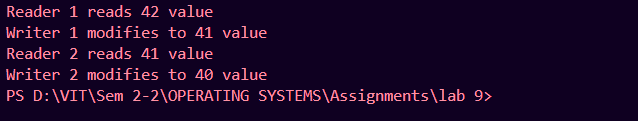
    {

        pthread\_create(&readers[i], NULL, (void \*)Reader, (void \*)&t);

        sleep(1);

        pthread\_create(&writers[i], NULL, (void \*)Writer, (void \*)&t);

        sleep(1);

    }

1. **Modify the question 1 as 1 writer performs increment operation and another writer performs decrement operation of the same account. Readers read and display the shared variable.**

* **If thread creation is parallelly done**

#include <stdio.h>

#include <unistd.h>

#include <pthread.h>

#include <semaphore.h>

pthread\_mutex\_t mutex;

sem\_t writeMutex;

int readCount = 0;

int value = 42;

void \*Reader(void \*i)

{

    int \*n = (int \*)i;

    pthread\_mutex\_lock(&mutex);

    readCount++;

    if (readCount > 0)

        sem\_wait(&writeMutex);

    pthread\_mutex\_unlock(&mutex);

    printf("Reader %d reads %d value\n", \*n, value);

    pthread\_mutex\_lock(&mutex);

    readCount--;

    if (readCount == 0)

        sem\_post(&writeMutex);

    pthread\_mutex\_unlock(&mutex);

}

void \*WriterInc(void \*i)

{

    int \*n = (int \*)i;

    sem\_wait(&writeMutex);

    value = value - 1;

    printf("Writer %d modifies to %d value\n", \*n, value);

    sem\_post(&writeMutex);

}

void \*WriterDec(void \*i)

{

    int \*n = (int \*)i;

    sem\_wait(&writeMutex);

    value = value + 1;

    printf("Writer %d modifies to %d value\n", \*n, value);

    sem\_post(&writeMutex);

}

int main()

{

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

    sem\_init(&writeMutex, 0, 1);

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

    for (int t = 1, i = 0; i < 2; i++, t++)

    {

        pthread\_create(&readers[i], NULL, (void \*)Reader, (void \*)&t);

        sleep(1);

    }

    for (int t = 1, i = 0; i < 2; i++, t++)

    {

        if (i % 2 == 0)

            pthread\_create(&writers[i], NULL, (void \*)WriterInc, (void \*)&t);

        else

            pthread\_create(&writers[i], NULL, (void \*)WriterDec, (void \*)&t);

        sleep(1);

    }

    for (int i = 0; i < 2; i++)

    {

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

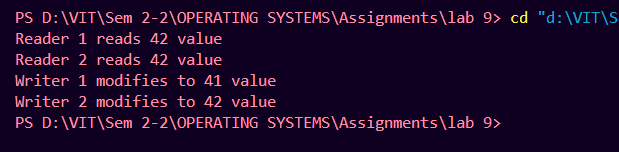
        pthread\_join(writers[i], NULL);

    }

    pthread\_mutex\_destroy(&mutex);

    sem\_destroy(&writeMutex);

    return 0;

}

* **If thread creation is done one after other.**

    for (int t = 1, i = 0; i < 2; i++, t++)

    {

        pthread\_create(&readers[i], NULL, (void \*)Reader, (void \*)&t);

        sleep(1);

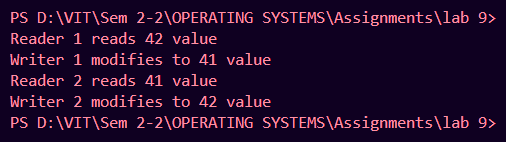
        if (i % 2 == 0)

            pthread\_create(&writers[i], NULL, (void \*)WriterInc, (void \*)&t);

        else

            pthread\_create(&writers[i], NULL, (void \*)WriterDec, (void \*)&t);

        sleep(1);

    }