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.**
    - **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.**

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

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

```c
#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;
}
```

```
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9> .\semaphore_ex1.c
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9> ./semaphore_ex1
Reader 1 reads 42 value
Reader 2 reads 42 value
Writer 1 modifies to 41 value
Writer 2 modifies to 40 value
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9>
```

- **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);
    }
```

```
Reader 1 reads 42 value
Writer 1 modifies to 41 value
Reader 2 reads 41 value
Writer 2 modifies to 40 value
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9>
```

2. **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;
}
```

```
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9> cd "d:\VIT\S
Reader 1 reads 42 value
Reader 2 reads 42 value
Writer 1 modifies to 41 value
Writer 2 modifies to 42 value
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9>
```

- **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);
    }
```

```
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9>
Reader 1 reads 42 value
Writer 1 modifies to 41 value
Reader 2 reads 41 value
Writer 2 modifies to 42 value
PS D:\VIT\Sem 2-2\OPERATING SYSTEMS\Assignments\lab 9>
```