4 First Readers-Writers Problem

4.1 Aim

Implement the first Readers-Writers problem

4.2 Theory

4.2.1 Critical Section Problem

Critical Section problem is a scenario in which there is a shared resource and multiple processes or threads are using this resource.When multiple processes or threads try to modify this resource an ambiguity can arise i.e different versions of the resource will be available to each resource.This is the critical section problem.

4.2.2 Semaphores

Semaphore is a variable used to control access to a shared resource.If the semaphore is 0 then the process must wait till the semaphore is changed.This is the wait function.Incrementing the semaphore is done using a function called signal.

4.2.3 Readers-Writers Problem

The readers-writers problem is a classical critical section problem in which there are two types of processes,Reader process and Writer process.The reader process can read the shared resource and the writer process can only modify the shared resource.The first Readers-Writers problem is a constrained version of the general Readers-Writers problem in which multiple readers can read at the same time.

4.3 Algorithm

Algorithm 1 ReaderCallBack Procedure

```
procedure ReaderCallBack(ID)                    ▷ ID is the thread number
    wait(readmutex)                  ▷ readmutex is the mutex for readcount
    readcount++
    if readcount == 1 then
        wait(resource)
    end if
    signal(readmutex)
    CRITICAL SECTION
    wait(readmutex)
    readcount–
    if readcount == 0 then
        signal(resource)          ▷ resource is the mutex for shared resource
    end if
    signal(readmutex)
    exitThread
end procedure
```

Algorithm 2 WriterCallBack

```
procedure WriterCallBack(ID)                    ▷ Id is the thread number
    wait(resource)           ▷ resource is the semaphore for shared resource
    CRITICAL SECTION
    signal(resource)
    exitThread                                          ▷ Exits the thread
end procedure
```

4.4 Code

```
#include<pthread.h>
#include<stdio.h>
#include<semaphore.h>
#include<cstdlib>
#include<time.h>
using namespace std;
sem_t resource;
sem_t rmutex;
int readcount = 0;
int shared_resource = 0;
```

```
void *ReaderCallBack(void * thread_no){
    sem_wait(&rmutex);
    readcount++;
    if(readcount == 1)
        sem_wait(&resource);
    sem_post(&rmutex);
    printf("Thread Number :%ld ,Shared Resource : %ld \n",(long)thread_no,shared
    sem_wait(&rmutex);
    readcount--;
    if(readcount == 0)
        sem_post(&resource);
    sem_post(&rmutex);
    pthread_exit(NULL);
}
void *WriterCallBack(void * thread_no){

    sem_wait(&resource);
    shared_resource++;
    printf("Shared resource was modified by thread number : %ld \n",(long)thread
    sem_post(&resource);
    pthread_exit(NULL);
}
int main(){
    sem_init(&resource,0,1);
    sem_init(&rmutex,0,1);
    srand (time(NULL));

    for(int i = 0;i<14;i++){
        int randomNo = rand()%100+1;
        if(randomNo <50){
            pthread_t reader;
            int iret;
            iret = pthread_create(&reader,NULL,ReaderCallBack,(void *)i);
            if(iret){
                return 1;
            }
            }
        else{
            pthread_t writer;
            int iret;
            iret = pthread_create(&writer,NULL,WriterCallBack,(void *)i);
            if(iret)
                return 1;

            }
```

```
    }

    return 0;
}
```

4.5 Output

```
Thread Number :2 ,Shared Resource : 0
Thread Number :3 ,Shared Resource : 0
Shared resource was modified by thread number : 1
Thread Number :4 ,Shared Resource : 1
Thread Number :0 ,Shared Resource : 1
Shared resource was modified by thread number : 6
Shared resource was modified by thread number : 7
Shared resource was modified by thread number : 8
Thread Number :9 ,Shared Resource : 4
Thread Number :10 ,Shared Resource : 4
Shared resource was modified by thread number : 11
Thread Number :12 ,Shared Resource : 5
Thread Number :13 ,Shared Resource : 5
```

4.6 Result

Implemented the program to demonstrate the First Reader-Writer problem in c++ and compiled using g++ 8.2.1 on arch linux,kernel version 4.20.6.The above output was obtained.