NAME - KAUSHAL ROHIT OZA
SRN – 201900754

ROLL NO - 40 (A)

OS ASSIGNMENT 5

Implement Reader Writer Problem using:

a) **threads and semaphores**
b) **Threads And Mutex**

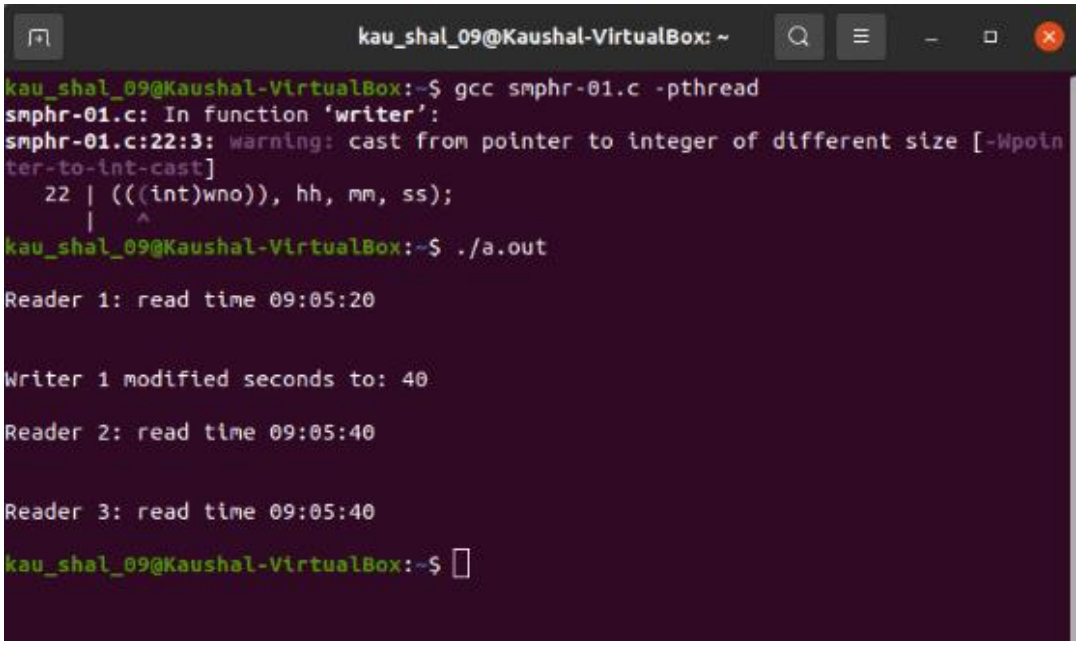
A) threads and semaphores

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<semaphore.h>

#include<pthread.h>



/* This program implements reader writer problem on a military clock using threads
and semaphores */



sem_t wrt; sem_t
mutex;


int hh=9, mm=5, ss=20; int
numreader=0; //writer void
*writer(void *wno)

{ sem_wait(&wrt);


        if(hh == 23 && mm == 59 && ss == 59)

        { hh=0;
                mm=0
                ;

                ss=0; printf("\nWriter %d: modified time %02d:%02d:%02d\n\n",

            (*((int*)wno)), hh, mm, ss);

        }
```

```
        else
        {
                ss = ss + 20; printf("\nWriter %d modified seconds to: %d\n", (*((int *)wno)),
                ss);

        } sem_post(&wrt);

}



//reader void *reader(void
*rno) {
        //acquire the lock, to read sem_wait(&mutex);
        numreader++; if(numreader == 1)
        { sem_wait(&wrt);          //block the writer
        } sem_post(&mutex);

        //read data printf("\nReader %d: read time %02d:%02d:%02d\n\n", (*((int *)rno)), hh,
mm, ss);


        //getting out
        sem_wait(&mutex); numreader--; if(numreader
        == 0)//there are no readers
        { sem_post(&wrt);  //if no reader, wake up writer
        }

        sem_post(&mutex);
}
```

```
int main(int argc, char* argv[])

{

    pthread_t read[3],write[1];
    sem_init(&wrt,0,1); sem_init(&mutex, 0, 1);
    int a[10] = {1,2,3,4,5,6,7,8,9,10};



        //create reader-0 pthread_create(&read[0], NULL, (void *)reader, (void
        *)&a[0]); pthread_join(read[0], NULL);

        //create writer-1 pthread_create(&write[0], NULL, (void*)writer, (void
        *)&a[0]); pthread_join(write[0], NULL);



        //create reader-1 pthread_create(&read[1], NULL, (void *)reader, (void
        *)&a[1]); pthread_join(read[1], NULL);



        //create reader-2 pthread_create(&read[2], NULL, (void *)reader, (void
        *)&a[2]); pthread_join(read[2], NULL);




    sem_destroy(&wrt); sem_destroy(&mutex);

    return 0;

}
```

OUTPUT:



B) Using threads and mutex:

```
#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<semaphore.h>

#include<pthread.h>

#include<time.h>



/* This program implements reader writer problem on a military clock using
   threads and mutex */



pthread_mutex_t wrt; pthread_mutex_t
mutex;

int hh=9, mm=5, ss=20; int
numreader=0;



//writer void *writer(void
*wno)
```

```
{ pthread_mutex_lock(&wrt); if(hh == 23 && mm == 59
        && ss == 59)

        { hh=0;
                mm=0
                ;

                ss=0; printf("\nWriter %d: modified time %02d:%02d:%02d\n\n",

            (*((int*)wno)), hh, mm, ss);

        }

        else

        {

                ss = ss + 20; printf("\nWriter %d modified seconds to: %d\n", (*((int *)wno)),
                ss);

        } pthread_mutex_unlock(&wrt);

        //free(wno);

}




//reader void *reader(void
*rno) {

        //acquire the lock, to read
        pthread_mutex_lock(&mutex);
        numreader++; if(numreader == 1)

        { pthread_mutex_lock(&wrt);     //block the writer
        } pthread_mutex_unlock(&mutex);


        //read data printf("\nReader %d: read time %02d:%02d:%02d\n", (*((int *)rno)), hh,

mm ,ss);



        //getting out pthread_mutex_lock(&mutex);
        numreader--; if(numreader == 0)//there are no
        readers

        { pthread_mutex_unlock(&wrt);  //if no reader, wake up writer
```

```
        }

        pthread_mutex_unlock(&mutex);
}




int main(int argc, char* argv[])
{

    pthread_t read[4],write[2];

    //pthread_mutex_init(&mutex, NULL);

    pthread_mutex_init(&wrt, NULL);

    pthread_mutex_init(&mutex, NULL); int a[10] =

    {1,2,3,4,5,6,7,8,9,10};




        //create reader-1 pthread_create(&read[0], NULL, (void *)reader, (void
        *)&a[0]); pthread_join(read[0], NULL);

        //create writer-1 pthread_create(&write[0], NULL, (void*)writer, (void
        *)&a[0]); pthread_join(write[0], NULL);


        //create reader-2

        pthread_create(&read[1], NULL, (void *)reader, (void *)&a[1]);
        pthread_join(read[1], NULL);


        //create reader-3 pthread_create(&read[2], NULL, (void *)reader, (void
        *)&a[2]); pthread_join(read[2], NULL);


        //create writer-2 pthread_create(&write[1], NULL, (void*)writer, (void
        *)&a[1]); pthread_join(write[1], NULL);
```

```
        //create reader-4 pthread_create(&read[3], NULL, (void *)reader, (void
        *)&a[3]); pthread_join(read[3], NULL);



    pthread_mutex_destroy(&wrt);

    pthread_mutex_destroy(&mutex); return 0;

}
```

OUTPUT: