NAME : V.Moneesha

Reg.No : 11706225

Email : [venutherlamoneesha@gmail.com](mailto:venutherlamoneesha@gmail.com)

Github Link :

DESCRIPTION :

This is a multi-thread program that gives readers priority over writers concerning a shared (global) variable. Esentially,if any readers are waiting,then they have priority over writer threads – writers can only write when there are no readers.

CONNDITIOS :

* No.of readers to be present for a writer to get executed = 0.
* Both the readers and writers shouls access the same global variable.
* Readers preemption over writer is allowed.

Functions Present :

* \*reader()
* \*writer()
* Main()
* No,of readers = 5
* No.of writers = 5
* Mutex semaphores : Mutex1,Mread

CODE :

#include<stdio.h>

#include<semaphore.h>

#include<pthread.h>

int rc=0,val,wc=0;

pthread\_mutex\_t mutex1,mread;

pthread\_t tr1,tr2,tr3,tr4,tr5,tw1,tw2,tw3,tw4,tw5;

pthread\_attr\_t tr1attr,tr2attr,tr3attr,tr4attr,tr5attr,tw1attr,tw2attr,tw3attr,tw4attr,tw5attr;

void \*reader();

void \*writer();

int main()

{

pthread\_mutex\_init(&mutex1,NULL);

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

pthread\_attr\_init(&tr1attr);

pthread\_attr\_init(&tr2attr);

pthread\_attr\_init(&tr3attr);

pthread\_attr\_init(&tr4attr);

pthread\_attr\_init(&tr5attr);

pthread\_attr\_init(&tw1attr);

pthread\_attr\_init(&tw2attr);

pthread\_attr\_init(&tw3attr);

pthread\_attr\_init(&tw4attr);

pthread\_attr\_init(&tw5attr);

printf("\n Writer 1 created");

pthread\_create(&tw1,&tw1attr,writer,NULL);

printf("\n Reader 1 created");

pthread\_create(&tr1,&tr1attr,reader,NULL);

printf("\n Writer 2 created");

pthread\_create(&tw2,&tw2attr,writer,NULL);

printf("\n Reader 2 created");

pthread\_create(&tr2,&tr2attr,reader,NULL);

printf("\n Writer 3 created");

pthread\_create(&tw3,&tw3attr,writer,NULL);

printf("\n Reader 3 created");

pthread\_create(&tr3,&tr3attr,reader,NULL);

printf("\n Writer 4 created");

pthread\_create(&tw4,&tw4attr,writer,NULL);

printf("\n Reader 4 created");

pthread\_create(&tr4,&tr4attr,reader,NULL);

printf("\n Writer 5 created");

pthread\_create(&tw5,&tw5attr,writer,NULL);

printf("\n Reader 5 created");

pthread\_create(&tr5,&tr5attr,reader,NULL);

pthread\_join(tw1,NULL);

pthread\_join(tr1,NULL);

pthread\_join(tw2,NULL);

pthread\_join(tr2,NULL);

pthread\_join(tw3,NULL);

pthread\_join(tr3,NULL);

pthread\_join(tw4,NULL);

pthread\_join(tr4,NULL);

pthread\_join(tw5,NULL);

pthread\_join(tr5,NULL);

return 0;

}

void \*writer()

{

pthread\_mutex\_lock(&mutex1);

wc++;

sleep(1);

printf("\n\n\nwriter %d initiated",wc);

printf("\n\nreaders present : %d",rc);

printf("\n\n Enter data: ");

scanf("%d",&val);

pthread\_mutex\_unlock(&mutex1);

// printf("\n writer %d executed",wc);

sleep(1);

printf("\n\n value written by writer is : %d",val);

pthread\_exit(0);

}

void \*reader()

{

pthread\_mutex\_lock(&mread);

rc++;

if(rc==1)

pthread\_mutex\_lock(&mutex1);

pthread\_mutex\_unlock(&mread);

printf("\n Value read by reader %d: %d",rc,val);

pthread\_mutex\_lock(&mread);

rc--;

if(rc==0)

pthread\_mutex\_unlock(&mutex1);

pthread\_mutex\_unlock(&mread);

printf("\n Reader %d executing",rc+1);

//sleep(1);

pthread\_exit(0);

}

EXICUTION :

Varible declaration :

All the reqired global varibles like wc,rc,val are declared as integer type.

Mutex semaphores mutex1 and mread are declared using Mutex class.

Required no.of threads and their attributes are de

clared using respective Thread classes which are derives from the above pthread.h header file.

Main function :

All the readers and writers are created using their function call with respective attributes passing into them.

The attributes passing a NULL value ,as the function doesn’t contain any parameters.

Writer function :

In the writer function, the writer gets initiated and also displays the no.of readers present at that instant of time .

Then the writer gets locked and reader function gets preempted.

After that, user will be asked to enter the value that should be read and written.

The writer gets executed ,if no readers are present.

Reader function :

In the reader function, the reader prremts the previous write function.

Here,the reader count is incremented and two test cases get checked using IF condition.

If the rc =0 ,then the reader goes back to the writer scope.

If the rc !=0,then readers get executed.

OUTPUT :

