[](https://www.bing.com/images/search?q=lpu+logo&id=8B7BE5F78CA3009715949D0D70C19E5AAF0B8670&FORM=IQFRBA)

**OPERATING SYSTEM PROJECT REPORT**

**CREATED BY: Amit Kumar**

**Roll NO: 68**

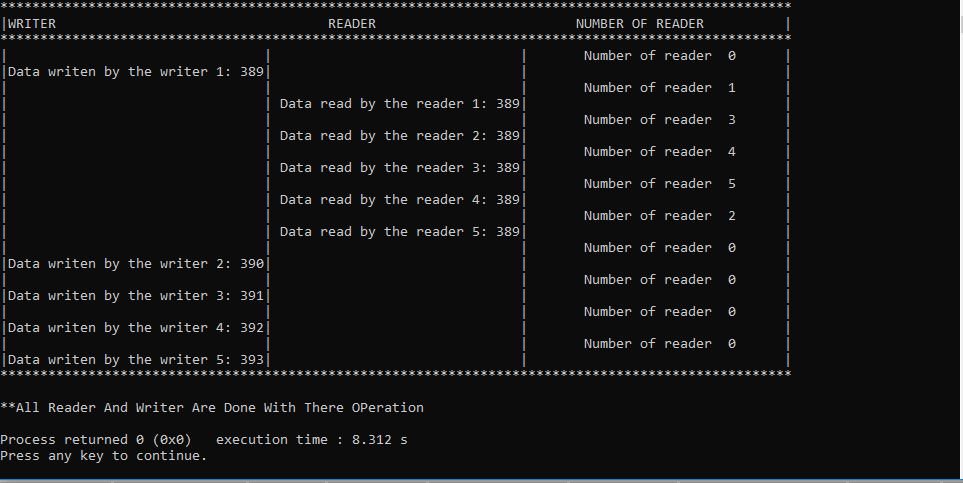
**REG NO: 11716034**

**SECTION: K17BG**

**What is Reader – Writer Problem**

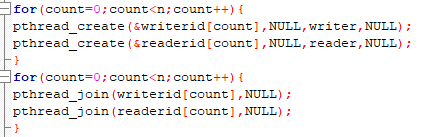
Reader-writer problem is another example of a classic synchronization problem. There are many variants of this problem.

There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are **reader** and **writer**. Any number of **readers** can read from the shared resource simultaneously, but only one **writer** can write to the shared resource. When a **writer** is writing data to the resource, no other process can access the resource. A **writer** cannot write to the resource if there are non-zero number of readers accessing the resource at that time.

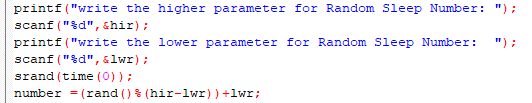


Full Project code Overview with explanation

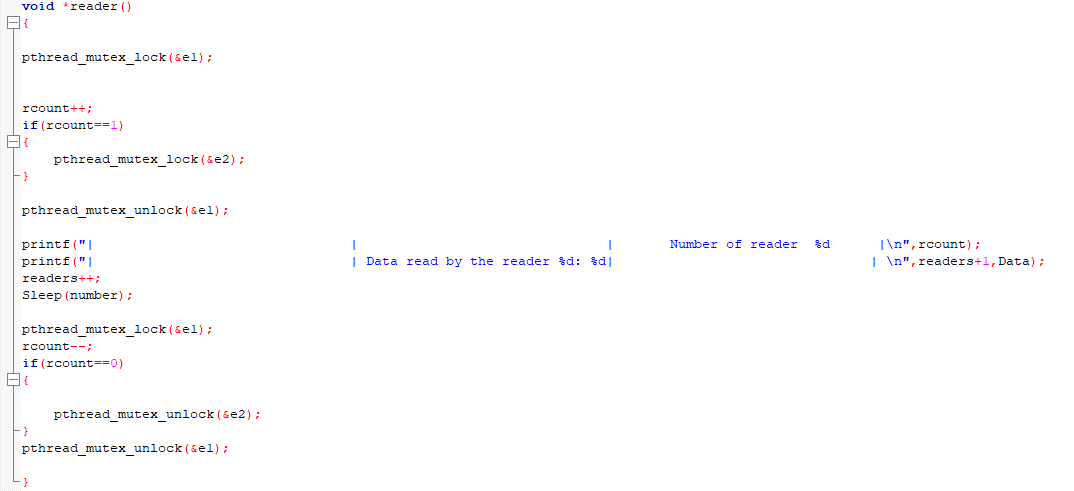
1) Multithread were used in this program.



2)Random function used in this program for making some random interval between reader and writer.



3) Reader Function in which we give priority to reader over writer means writer only execute when there will be no reader present.



4. Writer function which is simple because we don’t have to give any priority to writer over reader.



5) We use sleep function with some variable called number. Sleep just work for stop process for some time like in above program sleep will going to stop process time as per the value of number.

**First Part I created for this program on 30TH MARCH**

#include<stdio.h>

#include<pthread.h>

#include<semaphore.h>

#include<sys/types.h>

#include<unistd.h>

void \*fun1(void \*args);

void \*fun2(void \*args);

int readercount=0;

pthread\_mutex\_t t1;

pthread\_mutex\_t t2;

int i,y=0;

int main()

{

pthread\_mutex\_init(&t1,NULL);

pthread\_mutex\_init(&t2,NULL);

pthread\_t reader,reader1;

pthread\_t writer,writer1;

printf("\n");

printf("\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf( " VALUES WRITER INFO READER INFO NUMBER OF READER \n" );

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n");

printf("\n");printf(" writer is waiting\n\n");

int m;

pthread\_create(&reader1, NULL, fun1, (void\*) &m);

pthread\_create(&writer1, NULL, fun2, (void\*) &m);

pthread\_create(&reader, NULL, fun1, (void\*) &m);

pthread\_create(&writer, NULL, fun2, (void\*) &m);

pthread\_join(reader, NULL);

pthread\_join(writer, NULL);

pthread\_join(reader1, NULL);

pthread\_join(writer1, NULL);

return 0;

}

void \*fun1(void \*arg)

{ int f;

f = ((int)arg);

pthread\_mutex\_lock(&t1);

readercount++;

if(readercount==1){

pthread\_mutex\_lock(&t2);printf(" writer is waiting\n\n");

}

pthread\_mutex\_unlock(&t1);

printf("Data read by the reader%d is %d\n",f,y);

Sleep(1);

pthread\_mutex\_lock(&t1);

readercount = readercount--;

if(readercount==0){

pthread\_mutex\_lock(&t2);

}

pthread\_mutex\_unlock(&t1);

}

void \*fun2(void \*arg)

{

int f;

f = ((int) arg);

printf("writer is waiting\n\n");

pthread\_mutex\_lock(&t2);

printf("number of reder %d",readercount);

y++;

printf("information given by the writer%d is %d\n\n",f,y);

Sleep(1);

pthread\_mutex\_unlock(&t2);

}

* This was just a initialize step I used multi threads which calls reader and writer function .No user could give any value on the time of execution means simple reader- writer problem in which reader got the priority over writer

**Second part of this project on 7th of April**

|  |
| --- |
|  |
|  |  |  | #include<pthread.h> |
|  |  |  | #include<unistd.h> |
|  |  |  | #include<time.h> |
|  |  |  |  |
|  |  |  |  |
|  |  |  | pthread\_mutex\_t e1,e2; |
|  |  |  | int rcount = 0; |
|  |  |  | int count=0,n; |
|  |  |  | void \*reader(); |
|  |  |  | void \*writer(); |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | int main() |
|  |  |  | { |
|  |  |  | printf("number of reader and writer :"); |
|  |  |  | scanf("%d",&n); |
|  |  |  | pthread\_t readerid[n],writerid[n]; |
|  |  |  |  |
|  |  |  | pthread\_mutex\_init(&e1,NULL); |
|  |  |  | pthread\_mutex\_init(&e2,NULL); |
|  |  |  |  |
|  |  |  | printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); |
|  |  |  | printf("|WRITER READER NUMBER OF READER |\n"); |
|  |  |  | printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); |
|  |  |  |  |
|  |  |  | while(count<n){ |
|  |  |  | pthread\_create(&writerid[count],NULL,writer,NULL); |
|  |  |  | pthread\_create(&readerid[count],NULL,reader,NULL); |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  | pthread\_join(writerid[count],NULL); |
|  |  |  | pthread\_join(readerid[count],NULL); |
|  |  |  | count++; |
|  |  |  | } |
|  |  |  |  |
|  |  |  |  |
|  |  |  | printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"); |
|  |  |  | printf("\n"); |
|  |  |  | printf("\*\*All Reader And Writer Are Done With There OPeration"); |
|  |  |  | printf("\n"); |
|  |  |  | return 0; |
|  |  |  |  |
|  |  |  | } |
|  |  |  | void \*reader() |
|  |  |  | { |
|  |  |  | pthread\_mutex\_lock(&e1); |
|  |  |  |  |
|  |  |  |  |
|  |  |  | rcount++; |
|  |  |  | if(rcount==1) |
|  |  |  | { |
|  |  |  | pthread\_mutex\_lock(&e2); |
|  |  |  | } |
|  |  |  | pthread\_mutex\_unlock(&e1); |
|  |  |  |  |
|  |  |  | printf("| | Data read by the reader %d| | \n",count); |
|  |  |  |  |
|  |  |  | Sleep(1000); |
|  |  |  |  |
|  |  |  | pthread\_mutex\_lock(&e1); |
|  |  |  | rcount--; |
|  |  |  | if(rcount==0) |
|  |  |  | { |
|  |  |  |  |
|  |  |  | pthread\_mutex\_unlock(&e2); |
|  |  |  | } |
|  |  |  |  |
|  |  |  | pthread\_mutex\_unlock(&e1); |
|  |  |  |  |
|  |  |  | } |
|  |  |  | void \*writer() |
|  |  |  | { |
|  |  |  |  |
|  |  |  | pthread\_mutex\_lock(&e2); |
|  |  |  | printf("| | | Number of reader %d |\n",rcount); |
|  |  |  | printf("|Data writen by the writer %d| | |\n",count); |
|  |  |  | Sleep(1200); |
|  |  |  |  |
|  |  |  | pthread\_mutex\_unlock(&e2); |
|  |  |  |  |
|  |  |  |  |
|  |  |  | } |

* In this I simply add functionality like user can give any value on the time of execution.
* Used sleep function with given number for just to do differentiate between reader and writer process.
* In this program we can take as many Reader- Writer as we want.

**THE END**