ASSIGNMENT NO.:5

- **Title**:

  Thread Synchronization and mutual exclusion using mutex**.**

- **Objective:-**

  To understand the classical synchronization problem:reader writers problem.

- **Problem statement:**Thread Synchronization and mutual exclusion using mutex.application to demonstate :reader –writer with reader priority.
- **Theory:** The readers-writers problem**.**

The readers-writers problem is based upon a system in which an unspecified number of readers and writers process exist.in addition this system also includes a shared data to be accessed by readers and writers.here reader processes only read the data and writer processes can modify or update the data.

The reader writer problem is characterized by the fact that any number of readers can read from the shared resource simultaneousl y,but only one writer can write to the shared resource.

In addition when a write is writing or modif ying the shared resource no other process is allowed to access the resource.

In the similar fashion a writer is not allowed to access the resource while a non-zero number of readers are accessing theresource.

The ideal solution of the readers – writers problem should satisfy the following conditions:

1) Many readers are allowed to read concurrently.
2) While a writer is writing no reader is allowed to read.

3) At any one instance of time only one writer is allowed to write on the shared resource.

In these three conditions the preferences for reader and writer processes are not assumed but in a situation where priority is required to assign one more additional condition is imposed.

A readers get non-preemptive priority over writer and does not has to wait for the writer.this means if writer is not active while writing into shared medium a reader process will read the shared data before writer starts writing.

In this case the system is considered as readers preffered readers-writers system can also be considered.

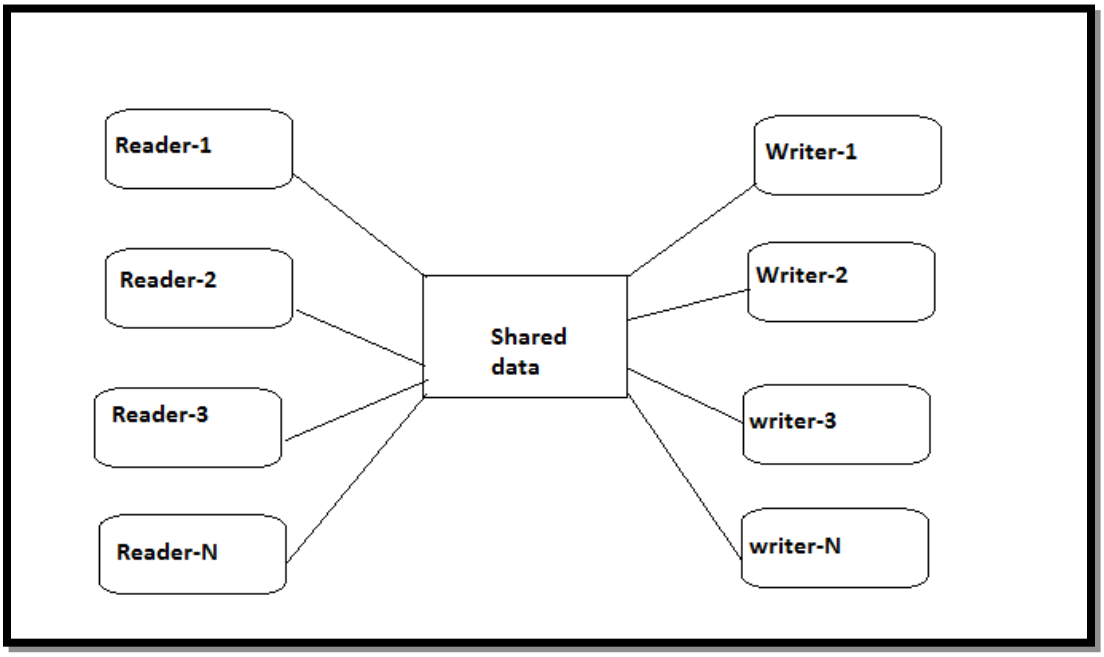

Fig:Readers-writers

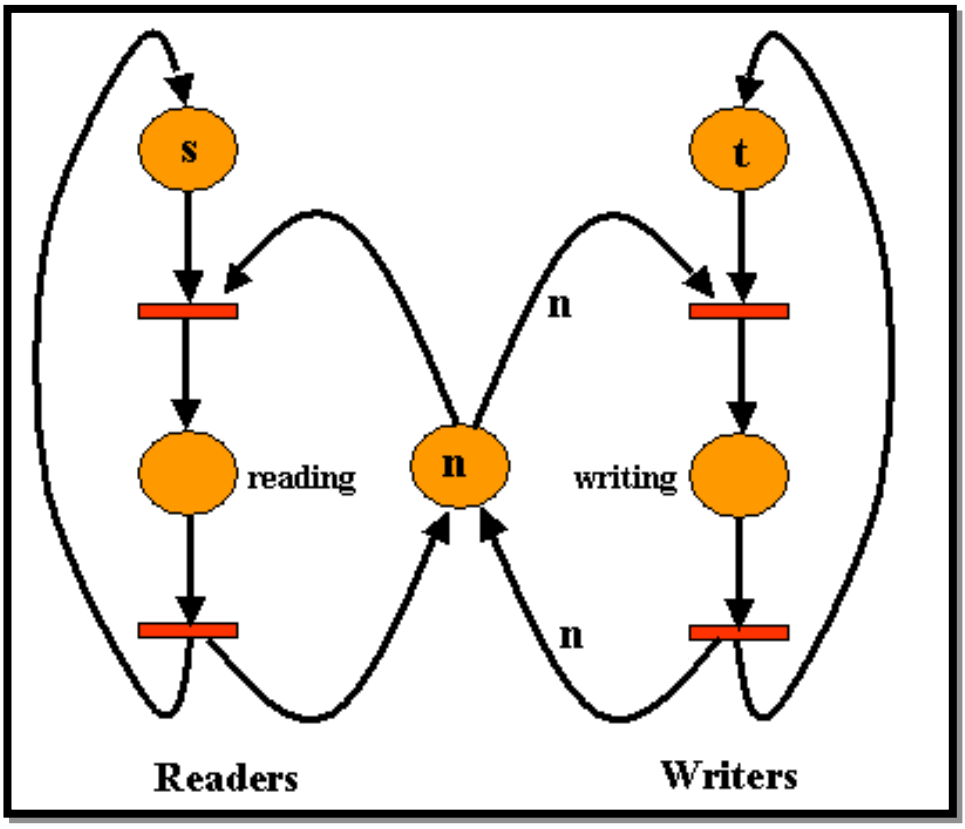

fig:reader writer problem reader priority

The dining philosophers problem is useful for modeling process that are competing for exclusive access to a limited number of resource,such as I/O devices.another famous problem is the readers and writers problem which models access to a database.imagine,for ex. An airline reservation system,with many competing processes wishing to read and wright it.it is acceptable to have multiple processes reading the database at the same time,but if one process is updating (writing)the database,no other processes may have access to the database,not even readers.

- **Program for solution to the readers and writers :**

Typedef int semaphore;

```
Semaphore mutex =1;

semaphore  db=1;

int  rc=0;

void reader(void)

{

While(TRUE) {

Down(&mutex);

rc=rc+1;

if(rc==1)down(&db);

up(&mutex);

rc=rc-1;

if(rc==0)up(&db)up(&mutex);

use-data-read();

}

}
```

In this solution,the first reader to get access to the database does a down on the semaphore db.subsequent readers merely increment a counter,rc.as readers leave , they allowing a blocked writer,if there is one,to get in.

Header files :

1) #include<sys/wait.h> : It is used declarations for waiting.

2)#include<stdio.h> : This ANSI header file relates to "standard" input/output functions. Files,dvices and directories are referenced using pointers to objects of the type FILE . The header file contains declarations for these functions and macros,

defines the FILE type, and contains various constants related to files.

3)#include<stdlib.h>:This ANSI header file contains declarations for many standard functions,

excluding those declared in other header files discussed in this section.

- ## Conclusion:-

The readers writers problem can be seen as a acceptable to have multiple processes reading the database at the same time,but one process is updating writing the database no other process may have access to the database,not even readers.