Student Name: Shubham Kumar

Student ID : 11603220

Email Address : [kumar.shubham16@outlook.com](mailto:kumar.shubham16@outlook.com)

Code : 1

GitHub Link: https://github.com/Shubham-16/Reader-Writer-Problem

**Description:-**

* The given problem is one of the classical problem of Process Synchronization which is know as Reader Writer Problem.(Readers have priority over Writers)
* Suppose there are multiple process running at the same time and a part of the data of code is shared between them. Some of them want to read the the data , while at the same time some of them want to modify it. If two or more readers want to read the data there might not be a problem but at the same time if a write comes and modifies it ,then the updated value might not be read by the readers. So. We must need a way that if a writer is writing(modifying) the data then there must not be any other writers or readers should be there in the shared portion(critical section) i.e. writer should have a exclusive access to the critical section. When the writer completes its work in critical section then it must signal the other waiting reader or writer process to access it.
* In case a reader is accessing the shared the data then it must insure that after reading the data it first give the chance to the other process which is waiting to read the data then it should signal to the process which is waiting for modifying the data. Here priority should be given to readers. This is also known as first readers-writers problem .
* There are two variations of reader-Writer problem one is first readers-writers problem which we are going to implement an the other one is second readers-writers problem where if a writer is ready then it first complete its writing process as soon as possible then only the readers can access the shared data.Here Writers have priority over readers.
* Both the variations may result to starvation. For example in first case where readers have priority over writers , writers may starve i.e it is possible that a writer waiting to enter the critical section may not get a chance. Similarly, In second case readers may not get a chance which may lead to starvation.

**-:Algorithm:-**

In this problem readers process will share the following three data structures:-

Semaphore mutex1=1

Semaphore mutex2= 1

Int rc=0

Semaphore mutex1 will be shared by both reader and writer process. It function as mutual exclusion semaphore for writers, it also is used by the first or last reader that enters or exits the critical section.

The mutex2 semaphore is used to ensure mutual exclusion between readers and writers.

The rc variable keeps the track of the number of readers in the critical section.

**-:Algorithm for readers;-**

do{

wait(mutex2);

rc++;

if (rc = =1)

wait(mutex1),

Signal(mutex2);

//reading is process

Wait(mutex2);

rc--;

If(rc = = 0)

Signal(mutex1);

Signal(mutex2);

} while(true);

Here in the algorithm it is ensured that if a reader is waiting then it will be allowed to enter the critical section rather than the writers.

**-:Algorithm for Writers:-**

do{

Wait(mutex1);

// writing process

Signal(mutex1);

} while(true);

In the above two algorithms wait() function is used to decrement the value of semaphore variables while signal() is used to increment the semaphore variable value.



