Student Name: Manish Kumar

Student Id: 11801808

Email Address: [mbunny1803@gmail.com](mailto:mbunny1803@gmail.com)

GitHub Link: <https://github.com/manish-prog/reader-writer-problem>

Code: 1

DESCRIPTION

The question is to write a multi-thread C program that give reader the priority over writer concerning a global variable. In this program Multi-threaded means that the program can handle more than one user at a single instance of time, or in other words the system can fulfill requests concurrently. It can manage multiple requests from the user without having to multiple copies of the program running.

ALGORITHM

**Write process:**

1. Writer requests the entry to critical section.
2. If allowed i.e. lock() gives a true value, it enters and performs the write. If not allowed, it keeps on waiting.
3. It exits the critical section.

do {

// writer requests for critical section

lock(wrt);

// performs the write

// leaves the critical section

unlock(wrt);

} while(true);

**Reader process:**

1. Reader requests the entry to critical section.
2. If allowed:
   * it increments the count of number of readers inside the critical section. If this reader is the first reader entering, it locks the **wrt** semaphore to restrict the entry of writers if any reader is inside.
   * It then, signals mutex as any other reader is allowed to enter while others are already reading.
   * After performing reading, it exits the critical section. When exiting, it checks if no more reader is inside, it signals the pthread “wrt” as now, writer can enter the critical section.
3. If not allowed, it keeps on waiting.

do {

// Reader wants to enter the critical section

lock(mutex);

// The number of readers has now increased by 1

count++;

// there is atleast one reader in the critical section

**// this ensure no writer can enter if there is even one reader**

**// thus we give preference to readers here**

if (count==1)

lock(wrt);

// other readers can enter while this current reader is inside

// the critical section

unlock(mutex);

// current reader performs reading here

lock(mutex);   // a reader wants to leave

count--;

// that is, no reader is left in the critical section,

if (count== 0)

unlock(wrt);         // writers can enter

unlock(mutex); // reader leaves

} while(true);

COMPLEXITY

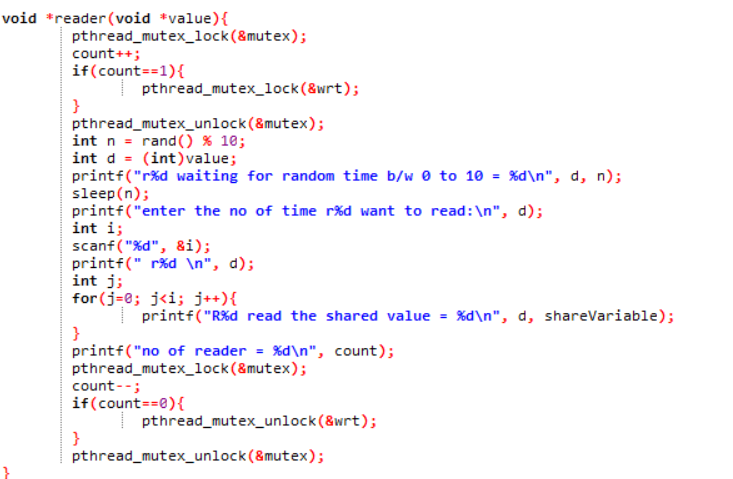
For Reader=> O(n)

For Writer=>O(n)

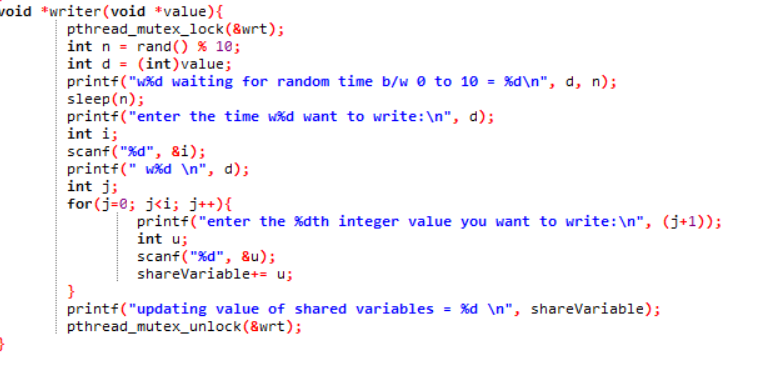
Overall complexity is=> O(n)

CONSTRAINTS

* Readers must read the shared variable X number of times.
* Readers must print the value read



* Writers must write the shared variable X number of times



Boundary condition

* The constraints for this code is that writer is given priority over reader means if any reader is waiting then they have the priority over writer threads, i.e writer can only write if there are no readers.
* Neither value of reader or writer can be negative. In this case program will terminate.
* Neither values of reader or writer can be zero, if value of writer is zero then only reader

operation is performed. And if reader is zero then only write.

* For succesfull execution of the program, both values of reader writer has to be greater than zero.

Test cases

* Test case 1-

I/P :- When user enter the negative value of reader

O/P :- Program terminates

No read or write operation is done.

* Test case 2-

I/P :- When user enter the negative value of writer

O/P :- Program terminates

No read or write operation is done

* Test case 3-

I/P :- When user enter the 0 value of reader

O/P :- Only writer thread is created and no reader thread is created

Only writer thread will update the share variable

* Test case 4-

I/P :- When user enter the 0 value of writer

O/P :- Only reader thread is created and no writer thread is created

Share variable will not be updated

* Test case 4-

I/P:- When the user enter values other than zero for both reader writer

O/P:- Thread will be created successfully and any writer can update the share variables any no times.