**Operating Systems 2**

**Assignment-4 Report**

**Varun Gopal**

**CO22BTECH11015**

The main objective of this assignment is to solve the classic readers-writers problem using semaphores. The Writers preference as well as the Fair solution are written in the two source-files Assgn4Src-CO22BTECH11015.cpp and Assgn4Src2-CO22BTECH11015.cpp. Semaphores are higher level of abstraction of synchronisation tools which are used by the different threads to gain access to shared memory locations viz. Critical sections.

Explanation of Code:-

Global variables are declared in the start of the program to be used in common across all the threads. If there exists any shared variable/Critical section variables, They can be declared in this section. The Global variables used are:-

Kw: Number of times each writer thread accesses critical section

Kr: Number of times each reader thread accesses critical section

Nr: Number of reader threads

Nw: Number of writer threads

Ucs: Mean time in critical section

Urem: Mean time in remaining section

\*fptr: File pointer to write the log files of each method

\*outfile: File pointer to write the average and the worst case times of each reader as well as writer.

\*writertime: A pointer to a vector to store the times taken by all the writer threads to enter the critical section

\*readertime: A pointer to a vector to store the times taken by all the reader threads to enter the critical section.

\*worstcasereader: A pointer to a vector to store the worst case times taken by all the reader threads to enter the critical section.

\*worstcasewriter: A pointer to a vector to store the worst case times taken by all the writer threads to enter the critical section.

All the above global shared variables are common to both files. Next some functions are defined to get the required time used by the reader as well as the writer threads. The getSysTime() function returns the current time as a string. The generate\_exponential() function generates a random number from the exponential distribution given the mean 1/lambda.

We define a semaphore struct which contains members:

1. Readers: total number of readers in CS.
2. Wr\_witing: Number of threads waiting to take control of CS.
3. Commonlock: Lock which allows many-readers/one writer to enter CS.
4. Readlock: Lock which is used by the readers to modify ‘readers’.
5. Wrwait: Used to take control of the queue
6. Writelock: used by writers to modify ‘Wr\_waiting’
7. Rwqueue: Similar to Wrwait used in the writers preference solution.

Writers Preference Solution:

We first initialise the semaphore using the function initialiselock(). Next we have functions for acquire\_readlock() which acquires the wrwait lock and the readlock, increments the ‘readers’ variable and waits for the common lock if there were no previous readers. After acquiring the common lock it posts back the wrwait as well as the readlock semphores.

After the reading part is done, The reader waits for the readlock,decrement the count of readers. If there are no more readers, the common lock is posted back. Finally the readlock is posted for it to be used by other readers.

The writers acquire the writelock first. Then increase the number of writers waiting viz. wr\_waiting. If It is the first writer, it waits for the control of the queue. Then it posts the write lock and waits for the control from the readers using the commonlock. After it finishes its work in the critical section, The writer releases the common lock and acquires the writelock to modify wr\_waiting. If It is the last writer, it releases the control of the queue to the readers. Then it posts the writelock.

Each reader and writer thread performs the operations on Critical section kr and kw number of times respectively.

Fair Readers-Writers Solution:

In this solution, the reader first acquires the control of the queue and the readlock. Then it increments the readers. If it is the first reader, it waits to acquire the commonlock. After this, It releases the readlock as well as the control of the queue. After leaving the Critical section, The reader decrements the number of readers by acquiring the readlock. If it is the last reader, It posts back the common lock and finally releases the readlock.

The writer acquires the queue and after that the commonlock. After it finishes its work in the critical section. It releases the both the locks.

Analysis of the Outputs:

1. Average Waiting time v/s Number of Readers:

As the number of readers increase, the average waiting time of each thread slightly increases in both the solutions. In the Writers Preference solution, The time almost remains constant for both reader as well as writer threads. This is due to the fact that as the execution progresses, Writers would eventually get control and execute at once. After all the waiting writers have finished their execution, Then the readers would get control. But the Readers threads have no contention after the writers have finished their task. Therefore the Average time take for all the readers threads is less compared to fair-solution. The writers have least waiting time in writers first solution due to obvious reasons. In the fair solution, There is contention between readers and writers after every request made. This is done using the First come first serve basis.

The following graph depicts the Experimental Results obtained by varying the number of readers for nw=10;ucs=10;urem=5;kw=10;kr=10

1. Average Waiting time vs Number of Writers:

In both the Writers Preference and Fair Readers Writers solutions, as the number of writers increases, the average waiting time of each writer thread also increases. However, the behaviour differs slightly between the two solutions. In the Writers Preference solution, the average waiting time for writers remains relatively constant as the number of writers increases. This is because writer threads are given priority over reader threads, allowing them to access the critical section without waiting for other threads. Therefore, even with an increasing number of writers, they can quickly access and execute their tasks. On the other hand, in the Fair Readers Writers solution, as the number of writers increases, the average waiting time for writers also increases. This is because in this solution, writers and readers are given equal priority to access the critical section. As a result, with more writers contending for access to the critical section, they may experience longer waiting times due to contention with reader threads.

Similar explanations can be extrapolated to the reader threads. Readers suffer more waiting time in average as all the writers need to finish before they start executing in the Writers preference solution.

The below graph shows the Average waiting time obtained by varying the number of writers with nr=10;kw=10;kr=10;ucs=10;urem=5

1. Worst Case Waiting time vs Number of readers:

In both solutions, the worst-case waiting time for both reader and writer threads increases as the number of reader threads increases. However, in the Writers Preference solution, the increase in worst-case waiting time for both readers and writers is generally more controlled and remains relatively stable compared to the Fair Readers Writers solution. This is because writers are given priority, reducing contention and waiting periods for readers. In contrast, the Fair Readers Writers solution experiences a more pronounced increase in worst-case waiting time for both readers and writers, as the equal treatment of threads leads to higher contention and longer waiting periods, particularly when the number of contending threads is high. In general the worst case time of readers in writers first would be more compared to all other trendlines due to the fact that all the writers need to finish their work before a reader enters the CS. In the fair solution there is contention among both readers and writers to attain control of the critical section. As a result the trendlines are usually higher.

The graph below shows the variation of Worst CS access time by changing the number of readers and having set the parameters nw=10, kw=10, kr=10, ucs=10 ,urem=5

1. Worst Case Waiting time vs Number of writers:

As the number of writers increases, the worst-case waiting time for both readers and writers would generally increase in both the Writers Preference and Fair Readers Writers solutions. nitially, as the number of writers increases, the worst-case waiting time for writers in the Writers Preference solution may remain relatively stable or increase slightly. This is because writers are given priority access to the critical section, allowing them to execute their tasks without much contention from readers. While the worst-case waiting time for writers may remain relatively stable, the increase in the number of writers can lead to longer waiting times for readers. This is because readers must wait for all active writers to finish their tasks before gaining access to the critical section. Therefore, an increase in the number of writers can potentially prolong the waiting time for readers, especially if there are many active writers contending for access. As the number of writers increases, the worst-case waiting time for writers in the Fair Readers Writers solution is expected to increase significantly. With more writers contending for access, readers may experience longer waiting times as they must wait for all active writers to finish their tasks before gaining access to the critical section.

The graph below shows the variation of Worst CS access time by changing the number of Writers and having set the parameters nw=10, kw=10, kr=10, ucs=10 ,urem=5