Readers Writers Problem:

Semaphore Based Solution

Multiple processes (readers and writers) trying to access common variables is known as the readers-writers problem. The difficulty is in making sure that readers' simultaneous access does not conflict with writers' need for exclusive access while they are writing. Semaphores are used to regulate and synchronize access to the shared resource in order to address this.

*Use of Semaphores:*

Mutex and writeMutex are the two semaphores used in this solution.

* mutex: This semaphore is used to update the ‘readerCount’ and manage access to shared variables.
* writeMutex: This semaphore guarantees writers' only access.

*Readers:*

Reading Process (readLock()):

* A reader first obtains the mutex in order to update the ‘readerCount’ in a safe manner when it wants to read.
* In order to stop writers from accessing the resource, it also obtains the writeMutex if ‘readerCount’ == 1.
* The reader releases the mutex once ‘readerCount’ has been updated.

Completing the reading (readUnLock()):

* The reader acquires the mutex to safely update the ’readerCount’ when it has finished reading.
* It releases the writeMutex to let writers access the resource if it's the final reader (readerCount == 0).
* The reader finally lets go of the mutex.

*Writers:*

Writing Process (writeLock()):

* When a writer wants to write, it directly acquires the ‘writeMutex’ for exclusive access.

Writing Completion (writeUnLock()):

* A writer releases the ‘writeMutex’ when they are done writing so that other writers or readers can use the resource.

*In outline:*

This approach guarantees that:

* No other writer or reader may access the shared resource while the writer is working on it.
* No writer can access the resource until all readers have completed reading if there are one or more readers.

This method achieves the needed synchronization between readers and writers in the readers-writers dilemma by successfully controlling access to the shared resource through the use of semaphores (mutex and writeMutex).