Computer Architecture and Operating Systems MONSOON SEMESTER of 2019

INSTRUCTOR: DR. Sambuddho Chakravarty

Readers-Writers Problem Implementation

A situation has been created where a queue is shared between multiple **Writers** and multiple **Readers**. Here, the conditions are:

1. If one of the Writers tries editing the file, no other Writer/Reader should be reading or writing at the same time, otherwise, changes will not be visible to him/her.
2. However, if some person is reading the file, then others may read it at the same time.

This situation is called the **Readers-Writers Problem**.

Code Description:

To solve the Readers-Writers Problem, **Semaphores** are used. Two semaphores have been used in the code, ***mutex,*** and ***writeblock.*** Semaphore *Mutex* is used to ensure mutual exclusion when the reader counter is updated i.e. when any reader enters or exits from the critical section and semaphore *writeblock* is used by both readers and writers. Input is taken for the number of Writers and Readers and the corresponding number of threads are spawned.

Writer Process:

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

Reader Process:

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

Usage:

1. Input the number of Reader and Writer Threads to be spawned.

Errors:

1. Errors corresponding to input has been handled. Only 5 threads for Reader and Writer can be spawned.
2. Errors corresponding to thread creation has been handled.
3. If a Reader thread reads data before any writer has added any data, NoDataException error is thrown.