33231

CLASSMATE
Date :
Page :

ASSIGNMENT 5

Title : Thread synchronization and Mutual Exclusion using Mutex

Problem Statement : Thread synchronization and mutual exclusion using mutex. Application to demonstrate : Reader-Writer problem with reader priority.

Theory :

Mutual Exclusion – property of process synchronization which states that "no two processes can exist in the critical section at any given point of time".

Reader-Writer Problem :

- One set of data is shared among a number of processes.
- Once a writer is ready, it performs its write. Only one writer may write at a time.
- If a process is writing, no other process can read it
- If atleast one reader is reading, no other process can write
- Readers may not write and only read.

Three variables are used : mutex, wrt, readcnt to implement solution

```
semaphore mutex, wrt ;
/* semaphore mutex is used to ensure mutual exclusion when
readcnt is updated i.e. when any reader enters or exit from
the critical section and semaphore wrt is used by both
readers and writers */

int readcnt ;
/* readcnt tells the number of processes performing read in the
critical section, initially 0 */
```

Functions for semaphore :

- wait() - decrements the semaphore value
- signal() - increments the semaphore value

Writer Process :

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

Writer Algorithm :

```
do {
    // writer requests for critical section
    wait (wrt);
    // performs the write
    // leaves the critical section
    signal (wrt);
} while (true);
```

Reader process :

Reader requests the entry to critical section. If allowed, it increments the count of numbers of readers inside critical section. If this reader is the first entering, it locks the wrt semaphore to restrict entry of writers if any reader is inside. If 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 semaphore "wrt" as

now, writer can enter the critical section. If not allowed, it keeps on waiting.

Reader Algorithm:

```
do { // Reader wants to enter the critical section
     wait (mutex) ;
     // The no. of readers has now increased by 1
     readcnt ++ ;
     // there is atleast one reader in the critical section
     // this ensures no writer can enter if there is even one reader
     // thus we give preference to readers here
     if (readcnt == 1)
         wait (wrt) ;
     // other readers can enter while this current reader is inside
     // the critical section
     signal (mutex)
     // current reader performs reading here
     wait (mutex) ; // a reader wants to leave
     readcnt -- ;
     //i.e. no reader is left in the critical section
     if (readcnt == 0)
         signal (wrt) ;    // writters can enter
     signal (mutex) ;      // reader leaves
} while (true) ;
```

Conclusion: I have successfully implemented Reader-Writer problem with reader priority. I have also understood thread synchronization and mutual exclusion using mutex.