Dr. Vishwanath Karad MIT WORLD PEACE UNIVERSITY | PUNE

TECHNOLOGY, RESEARCH, SOCIAL INNOVATION & PARTNERSHIPS

Assignment 5: Readers Writers Problem

School of Computer Engineering and Technology

Readers Writers Problem

- There is a data area shared among a number of processes.
- The data area could be a file or record
- There are number of processes that only read the data area(readers) and a number of processes that only write the data area (writers).
- Conditions that must be satisfied are as follows:
  - Any number of readers may simultaneously read the file.
  - Only one writer at a time may write to the file.
  - If a writer is writing to the file, no reader may read it.

Pseudo Code reader writer: readers have priority

- **int readcount = 0; // keeps track of number of readers**
- **semaphore mutex = 1, //binary, used for updating reader count**
- **semaphore wrt = 1; // binary, common to readers & writers. Mutual exclusion for writers & is used by 1st & last reader that enters or exits CS. Not used by readers who enter or exit while other readers are in their CS**

Pseudo Code readers-writers

```
void reader()
{while(true)
 {
    wait(mutex);

    readcount++;
         if(readcount == 1)
                wait(wrt);
     signal(mutex);
              ………
                reading is performed
                      ………..

    wait(mutex);
            readcount--;
         if (readcount == 0)
           signal(wrt);
     signal(mutex);
   }
```

```
void writer()
{
 while(true)
 {
    wait(wrt);
              ………
                     writing is performed
                  ………..

     signal(wrt);
    }
 }
```

Implementation

- #include <semaphore.h>

  and declare a semaphore of type sem_t in C.
- Some important methods that can be used with semaphore in C
  - **sem_init** -> Initialise the semaphore to some initial value
  - **sem_wait** -> Same as wait() operation
  - **sem_post** -> Same as Signal() operation
  - **sem_destroy** -> Destroy the semaphore

Implementation

```
#include<semaphore.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include<pthread.h>
sem_t mutex,wrt;
pthread_t tid;
int sharedvar=99;
pthread_t writers[5],readers[5];
int readercount=0;
```

Implementation

```
void reader()
{   sem_wait(&mutex);
    readercount++;
    if(readercount==1)
                sem_wait(&wrt);
    sem_post(&mutex);
    printf("\n%d reader is reading sharedvar=%d",readercount, sharedvar);
    printf("\nReader is done”);
        sem_wait(&mutex);
    readercount--;
    if(readercount==0)
                sem_post(&wrt);
    sem_post(&mutex);
    }
```

Implementation

```
void writer()
{
    printf("\nWriter is trying to enter");
    sem_wait(&wrt);
    printf("\nWriter has entered CS");
    sharedvar++;
    printf("\nWriter CHANGED THE VALUE OF SHARED VAR TO %d",sharedvar);
    sem_post(&wrt);
    printf("\nWriter is out of CS");
}
```

Implementation

```
int main()
{    int n2,i;
     printf("Enter the number of readers & writers:");
     scanf("%d",&n2);
     sem_init(&mutex,0,1);
     sem_init(&wrt,0,1);
     for(i=0;i<n2;i++)
     {   pthread_create(&writers[i],NULL, (void *)writer, NULL);
         pthread_create(&readers[i],NULL, (void *)reader, NULL);    }
     for(i=0;i<n2;i++)
     {  pthread_join(writers[i],NULL);
         pthread_join(readers[i],NULL);   }
}
```