MIT World Peace University

Operating Systems
Second Year B. Tech, Semester 3

# Process Synchronization - Simulation of Reader-Writer Problem in C

Assignment 2
Practical Report

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A2, PA 20

November 3, 2022

# 1 Code

```c
// Reader writer problem.

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <semaphore.h>

sem_t sem_wrt;
sem_t sem_mutex;
int shared_variable = 0;
int number_of_readers;

void *reader()
{
    sem_wait(&sem_mutex);
    printf("\nRead: %d\n", shared_variable);
    printf("Reader finished its CS so releasing mutex\n");
    sem_post(&sem_mutex);
}

void *writer()
{
    sem_wait(&sem_wrt);
    sem_wait(&sem_mutex);
    printf("Blocking sem wait and mutex variable so no other writer can write rn.
    \n");
    shared_variable++;
    printf("Wrote to the shared variable %d\n", shared_variable);
    sem_post(&sem_wrt);
    sem_post(&sem_mutex);
}

int main()
{
    pthread_t t1, t2;
    sem_init(&sem_mutex, 0, 1);
    sem_init(&sem_wrt, 0, 1);

    printf("Enter how many readers and Writers you want (Same number of both are
    taken by default): ");
    scanf("%d", &number_of_readers);

    for (int i = 0; i < number_of_readers; i++)
    {
        pthread_create(&t2, NULL, writer, NULL);
        pthread_create(&t1, NULL, reader, NULL);
    }

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    sem_destroy(&sem_mutex);
}
```

Listing 1: Assignment 5.Cpp

## 2 Input and Output

```
Enter how many readers and Writers you want (Same number of both are taken by
    default): 8
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 1

Read: 1
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 2

Read: 2
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 3

Read: 3
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 4

Read: 4
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 5

Read: 5
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 6

Read: 6
Reader finished its CS so releasing mutex

Read: 6
Reader finished its CS so releasing mutex

Read: 6
Reader finished its CS so releasing mutex
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 7
Blocking sem wait and mutex variable so no other writer can write rn.
Wrote to the shared variable 8
```

Listing 2: Input and Output.Cpp