**Assignment Number 04**

**Name:** Mihir Unmesh Patil

**Roll NO:** TYCOC213

**Batch**: C/ C-3

**CODE:**

#include <stdio.h>

#include <unistd.h>

#include <stdlib.h>

#include <time.h>

#include <pthread.h>

#include <semaphore.h>

#include <sched.h>

#define MAX 50

int sleepMod = 5;

int readCount = 0;

int isSync = 1; // Default: Synchronous mode

sem\_t readAccess, bookAccess;

void \*reader\_func(void \*);

void \*writer\_func(void \*);

int main() {

srand(time(0));

int readers, writers, mode;

printf("Choose mode: 1 for Synchronous, 2 for Asynchronous: ");

scanf("%d", &mode);

isSync = (mode == 1) ? 1 : 0;

printf("Number of readers (max 50): ");

scanf("%d", &readers);

printf("Number of writers (max 50): ");

scanf("%d", &writers);

if (readers > 5) sleepMod = readers;

pthread\_t readers\_t[MAX], writers\_t[MAX];

sem\_init(&readAccess, 0, 1);

sem\_init(&bookAccess, 0, 1);

int i;

for (i = 0; i < readers; i++)

pthread\_create(&readers\_t[i], NULL, reader\_func, &i);

for (i = 0; i < writers; i++)

pthread\_create(&writers\_t[i], NULL, writer\_func, &i);

for (i = 0; i < writers; i++)

pthread\_join(writers\_t[i], NULL);

for (i = 0; i < readers; i++)

pthread\_join(readers\_t[i], NULL);

sem\_destroy(&readAccess);

sem\_destroy(&bookAccess);

return 0;

}

void \*reader\_func(void \*r) {

int rNo = \*((int \*)r) + 1;

printf("\n Reader %d: wanting to read", rNo);

sleep(rand() % sleepMod);

if (isSync)

sem\_wait(&readAccess);

else

sched\_yield();

readCount++;

if (readCount == 1)

sem\_wait(&bookAccess);

printf("\n Reader %d: reading", rNo);

if (isSync)

sem\_post(&readAccess);

sleep(rand() % sleepMod);

if (isSync)

sem\_wait(&readAccess);

readCount--;

printf("\n Reader %d: leaving reading", rNo);

sleep(rand() % sleepMod);

if (readCount == 0)

sem\_post(&bookAccess);

if (isSync)

sem\_post(&readAccess);

printf("\n Reader %d: finished", rNo);

sleep(rand() % sleepMod);

pthread\_exit(0);

}

void \*writer\_func(void \*w) {

int wNo = \*((int \*)w) + 1;

printf("\n Writer %d: wanting to write", wNo);

sleep(rand() % sleepMod);

if (isSync)

sem\_wait(&bookAccess);

else

sched\_yield();

printf("\n Writer %d: writing", wNo);

sleep(rand() % sleepMod);

printf("\n Writer %d: leaving writing", wNo);

sleep(rand() % sleepMod);

if (isSync)

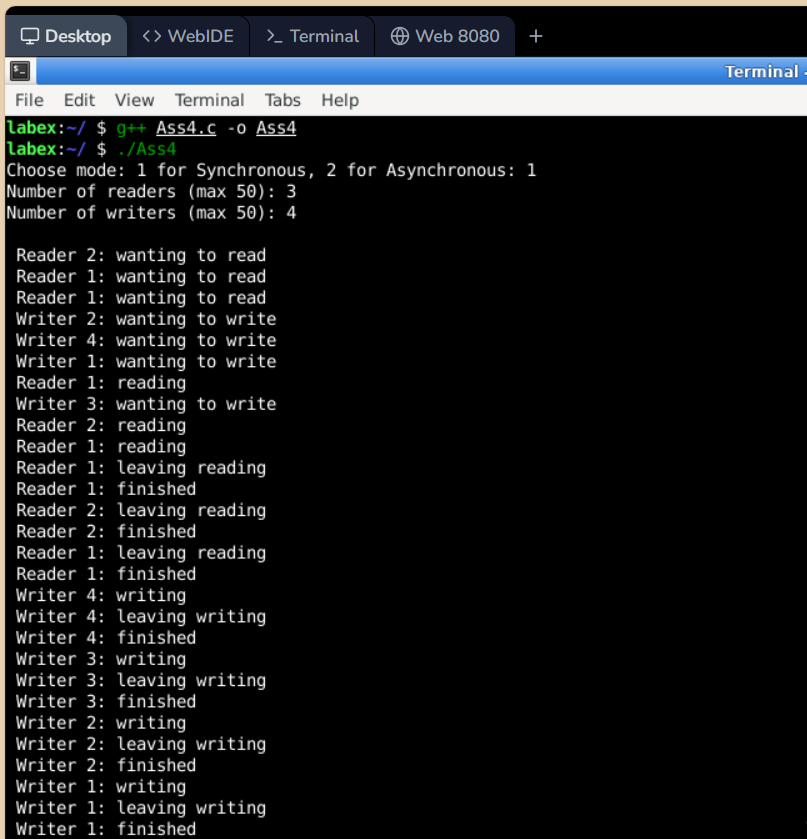
sem\_post(&bookAccess);

printf("\n Writer %d: finished", wNo);

sleep(rand() % sleepMod);

pthread\_exit(0);

}

**Output:**

