A black sign with white text

Description automatically generated

CSE-316: OPERATING SYSTEMS

Programming Assignment

Submitted to:

Dr. Baljit Singh Saini

Problem Statement

Write a multi-threaded C program that gives readers priority over writers concerning a shared (global) variable. Essentially, if any readers are waiting, then they have priority over writer threads -- writers can only write when there are no readers. This program should adhere to the following constraints: Multiple readers/writers must be supported (5 of each is fine) Readers must read the shared variable X number of times Writers must write the shared variable X number of times Readers must print: The value read The number of readers present when value is read Writers must print:

The written value The number of readers present were when value is written (should be 0) Before a reader/writer attempts to access the shared variable it should wait some random amount of time Note: This will help ensure that reads and writes do not occur all at once Use pthreads, mutexes, and condition variables to synchronize access to the shared variable

**Approach to solution**

Approach was to make a program which can adhere multiple users and to learn the basic about library so that things can be done smoothly.

Code

#include<stdio.h>

#include<pthread.h>

#include<stdlib.h>

#include<unistd.h>

pthread\_mutex\_t mutex, wrt;

int s, rcount = 0;

void \*writer(void \*arg){

pthread\_mutex\_lock(&wrt);

int n = rand() % 10;

int d = ((int)arg);

printf("--------------------------------------------------\n");

printf("W%d Wait for Random time between 0ns and 10ns = %d\n", d, n);

sleep(n);

printf("Enter the number of time W%d want to write:\n", d);

int t;

scanf("%d", &t);

printf("Now W%d is writing... i.e. ADDING...\n", d);

int j;

for(j=0; j<t; j++){

printf("Enter the %dth INTEGER value to write:\n", (j+1));

int u;

scanf("%d", &u);

s = s + u;

}

printf("UPDATED value of Shared variable = %d \n", s);

printf("--------------------------------------------------\n");

pthread\_mutex\_unlock(&wrt);

}

void \*reader(void \*arg){

//Entry Part

pthread\_mutex\_lock(&mutex);

rcount++;

if(rcount==1){

pthread\_mutex\_lock(&wrt);//No writer should come

}

pthread\_mutex\_unlock(&mutex);//so next reader can come

//Exit Part

int n = rand() % 10;

int d = ((int)arg);

printf("R%d wait for Random time between 0ns and 10ns = %d\n", d, n);

sleep(n);

printf("Enter the number of time R%d want to read:\n", d);

int t;

scanf("%d", &t);

printf("Now R%d is reading....\n", d);

int j;

for(j=0; j<t; j++){

printf("R%d read the shared value = %d\n", d, s);

}

printf("Number of Readers present = %d\n", rcount);

pthread\_mutex\_lock(&mutex);

rcount--;

if(rcount==0){//Now writer can come if they want

pthread\_mutex\_unlock(&wrt);

}

pthread\_mutex\_unlock(&mutex);

}

void main(){

printf("Enter the 'INTEGER' Initial value of share variable: \n");

scanf("%d", &s);

printf("---------------------------------------------\n");

int rn, wn, i;

printf("Enter the no. of Reader:\n");

scanf("%d", &rn);

for(i=0; i<rn; i++){

printf("R%d\n", i);

}

printf("---------------------------------------------\n");

printf("Enter the no. of Writer:\n");

scanf("%d", &wn);

for(i=0; i<wn; i++){

printf("W%d\n", i);

}

printf("---------------------------------------------\n");

pthread\_t r[rn], w[wn];

pthread\_mutex\_init(&wrt, NULL);

pthread\_mutex\_init(&mutex, NULL);

if(rn<0 || wn<0){

printf("Sorry: You have Entered NEGATIVE number of READER | WRITER\n");

printf("Program is Terminating....\n");

return;

}else if(rn == 0){

printf("Sorry: You have not taken any READER\n");

printf("No READER thread will be creaded\n");

}else if(wn == 0){

printf("Sorry: You have not taken any WRITER\n");

printf("No WRITER thread will be creaded\n");

}else{

printf("Thread Creating....\n");

}

printf("---------------------------------------------\n");

if(wn==rn){

for(i=0; i<wn; i++){

pthread\_create(&w[i], NULL, &writer, (int \*)i);

pthread\_create(&r[i], NULL, &reader, (int \*)i);

}

for(i=0; i<wn; i++){

pthread\_join(w[i], NULL);

pthread\_join(r[i], NULL);

}

}else if(wn>rn){

for(i=0; i<rn; i++){

pthread\_create(&w[i], NULL, &writer, (int \*)i);

pthread\_create(&r[i], NULL, &reader, (int \*)i);

}

for(i=rn; i<wn; i++){

pthread\_create(&w[i], NULL, &writer, (int \*)i);

}

for(i=0; i<rn; i++){

pthread\_join(w[i], NULL);

pthread\_join(r[i], NULL);

}

for(i=rn; i<wn; i++){

pthread\_join(w[i], NULL);

}

}else{

for(i=0; i<wn; i++){

pthread\_create(&w[i], NULL, &writer, (int \*)i);

pthread\_create(&r[i], NULL, &reader, (int \*)i);

}

for(i=wn; i<rn; i++){

pthread\_create(&r[i], NULL, &reader, (int \*)i);

}

for(i=0; i<wn; i++){

pthread\_join(w[i], NULL);

pthread\_join(r[i], NULL);

}

for(i=wn; i<rn; i++){

pthread\_join(r[i], NULL);

}

}

printf("-------------After joining the thread---------\n");

printf("Final value of share variable = %d\n", s);

}

Explanation for above code

C does not contain any built-in support for multithreaded applications. Instead, it relies entirely upon the operating system to provide this feature.

This tutorial assumes that you are working on Linux OS and we are going to write multi-threaded C program using POSIX. POSIX Threads, or Pthreads provides API which are available on many Unix-like POSIX systems

**pthread\_create** creates a new thread and makes it executable. This routine can be called any number of times from anywhere within your code. Here is the description of the parameters.

| **Parameter** | **Description** |
| --- | --- |
| thread | An opaque, unique identifier for the new thread returned by the subroutine. |
| attr | An opaque attribute object that may be used to set thread attributes. You can specify a thread attributes object, or NULL for the default values. |
| start\_routine | The C routine that the thread will execute once it is created. |
| arg | A single argument that may be passed to start\_routine. It must be passed by reference as a pointer cast of type void. NULL may be used if no argument is to be passed. |

The maximum number of threads that may be created by a process is implementation dependent. Once created, threads are peers, and may create other threads. There is no implied hierarchy or dependency between threads.

Snapshots and Explanations

1) When the User Enter Number of ‘WRITER’ is Negative

A screenshot of a computer screen

Description automatically generated

In this case

i. No WRITER and READER thread will be created

ii. No read or write operation done

iii. Program will be terminated

#### 2) When the User Enter Number of ‘READER’ is Negative

#### 

In this case

i. No WRITER and READER thread will be created

ii. No read or write operation done

iii. Program will be terminated

#### A screenshot of a computer Description automatically generated3) When Number of ‘WRITER=0’ is taken as input

In This Case:

i. Only READER thread will be created

ii. No WRITER thread is created

iii. No updating of the Shared Variable

iv. Only READER is reading the shared variable. From the example given above R1, R0 wait for sometimes. Then R1 read 1 time and R0 read 2 times. Finally Shared variable value will be remain same.

#### 4) When Number of “READER=0’ is taken as input

A screenshot of a computer screen

Description automatically generated

In This Case:

i. Only WRITER thread is created

ii. No READER thread is created

iii. No reading of the shared variable

iv. Only WRITER is updating the shared variable. From the same example given above W1, W0 will make change on the shared variable. W1 writes 2 times and W0 writes 1 time. Finally shared variable value will be changes/updated.

#### 5) User taken same number of READER and WRITER i.e. READER = WRITER

A screenshot of a computer

Description automatically generated

In This Case:

i. Both READER and WRITER threads created

ii. Any WRITER can update the Shared variable any number of times

iii. Any READER can read that Value any number of times

iv. In the screen slot, W0 wait for some random times. Then he has the choice to update the variable any number of times with any number. Here W0 update for 1 time. He added with value 5 and result in 26

#### 6) User taken a greater number of WRITER as compared with READER i.e. READER < WRITER

#### A screenshot of a computer screen Description automatically generated

In This Case:

i. Both READER and WRITER threads created

ii. Any WRITER can update the Shared variable any number of times

iii. Any READER can read that Value any number of times

iv. In this screen slot, W0 wait for some time and add some value with shared variable. Then R0 given the choice to read any number of times. Then again, the turn of next writer i.e. W1 came.

#### 7) User taken a greater number of READER as compared with WRITER i.e. READER > WRITER

#### A screenshot of a computer screen Description automatically generated

#### 8) Shared Variable value is taken non-INTEGER

#### A screenshot of a cell phone Description automatically generated

In This Case:

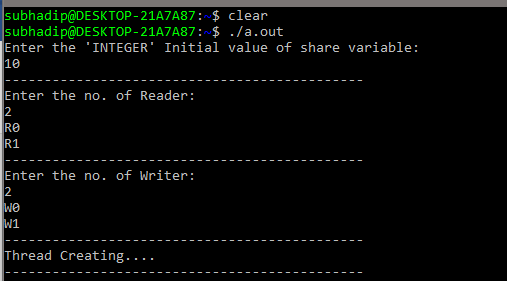
i. User unable to take any input

ii. No READER and WRITER thread were created

iii. We will not get output as we expected

iv. But get the Shared variable value as 0

# Program Working with a Proper example

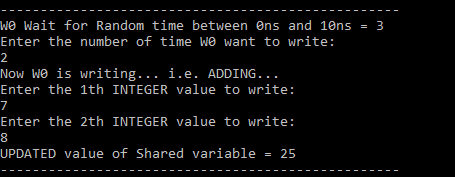
[](https://user-images.githubusercontent.com/44317938/55866691-7d7fd200-5b9e-11e9-8a08-334c01df5cac.PNG)

ii. User taken the initial value of share variable(S) as 10

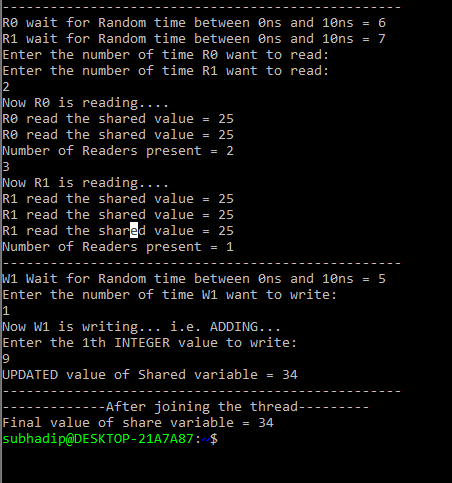
iii. User taken the no. of reader(R) is 2 as input i.e. R0 & R1

iv. User taken the no. of writer(W) is 2 as input i.e. W0 & W1

v. Now READER and WRITER threads were created successfully

[](https://user-images.githubusercontent.com/44317938/55866707-82dd1c80-5b9e-11e9-88da-895b75c15be2.PNG)

vii. W0 wait for some random time. Then he chooses that he will update S two time. Fist time he add 7 to S. S = 10 + 7 = 17 then second time he add 8 to S. Now S = 17 + 8 = 25. So, New updated value is 25.

[](https://user-images.githubusercontent.com/44317938/55866716-8670a380-5b9e-11e9-8461-2d75ce7cd750.PNG)

ix. Now R0 & R1 wait for some random times. Then R0 wants that he wants to read the shared variable 2 time. Then R1 read for 3 times. At this instant, 1 reader is present. Then W1 wait for some times and he want to add 9 to the shared variable for 1 time. S = 25 + 9 = 34.

x. Finally, after joining the thread, we got the shared variable value as 34.