Text

Description automatically generated

Lab-06

Course Title: Operating System

Course Code: CSE325

Section: 01

Submitted to:

Tanni Mittra

Senior Lecturer

Department of Computer Science & Engineering

Submitted by:

Name: Adri Saha

ID: 2019-1-60-024

Date of submission: 21 April 2020

**Lab06\_Problem01:**

Write a C program that will create two threads. One thread stores value into global array and other thread and print the values from the global array. Now synchronized the problem solution by using mutex lock.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

int arr[20],i=0,j=0;

void \*Reader(void \*arg)

{

pthread\_mutex\_lock(&mutex);

printf("Enter 5 array integers: \n");

for(i=0; i<5; i++)

{

scanf("%d",&arr[i]);

}

pthread\_mutex\_unlock(&mutex);

}

void \*Writter(void \*arg )

{

pthread\_mutex\_lock(&mutex);

printf("After printing numbers are : \n");

printf("Generating.......\n");

for(j=0; j<5; j++)

{

printf("%d ",arr[j]);

}

printf("\n");

pthread\_mutex\_unlock(&mutex);

}

int main()

{

int t1,t2;

pthread\_t thread1, thread2;

if((t1=pthread\_create(&thread1,NULL,&Reader,NULL)))

{

printf("Thread 1 couldn't create %d\n", t1);

}

pthread\_join( thread1, NULL);

if((t2=pthread\_create( &thread2,NULL,&Writter,NULL )))

{

printf("Thread 2 couldn't create %d\n", t2);

}

pthread\_join( thread2, NULL);

pthread\_mutex\_destroy(&mutex);

exit(0);

}

Output:

Text

Description automatically generated

**1\_Extension:**

#include <stdio.h>

#include <stdlib.h>

#include <pthread.h>

pthread\_mutex\_t mutex=PTHREAD\_MUTEX\_INITIALIZER;

int arr1[]= {2,4,6,8,1,3,5,7,9,12},arr2[10],pos=0;

int length = sizeof(arr1)/sizeof(arr1[0]);

void \*Reader(void \*arg)

{

int i=0;

pthread\_mutex\_lock(&mutex);

while(i<2)

{

arr2[i]=arr1[pos];

i++;

pos++;

}

pthread\_mutex\_unlock(&mutex);

}

void \* Writer(void \*arg)

{

pthread\_mutex\_lock(&mutex);

printf("Generating.......\n");

for(int j=0; j<2; j++)

{

printf("%d ",arr2[j]);

}

printf("\n");

pthread\_mutex\_unlock(&mutex);

}

int main()

{

int t1,t2;

pthread\_t thread1,thread2;

printf("After printing numbers are : \n");

while(pos<length)

{

t1=pthread\_create(&thread1,NULL,&Reader,NULL);

t2=pthread\_create(&thread2,NULL,&Writer,NULL);

if((t1!=0))

{

printf("Thread 1 couldn't create %d\n", t1);

}

pthread\_join( thread1, NULL);

if((t2!=0))

{

printf("Thread 2 couldn't create %d\n", t2);

}

pthread\_join(thread2, NULL);

}

pthread\_mutex\_destroy(&mutex);

}

**Output:**

Text

Description automatically generated

**Lab06\_Problem02:**

Write a C program that creates multiple threads with NULL as parameter sent to the thread execution

function, runner. Each thread inside the runner function must reads an integer value, n, and then produce

sum of values from 1 to n(inclusive) and prints the sum. Here sum is global variable. Ensure that the main ()

must wait for the termination of these threads and synchronized the problem solution by using semaphore

lock.

Code:

#include <stdio.h>

#include <stdlib.h>

#include<unistd.h>

#include<sys/wait.h>

#include <semaphore.h>

#include <pthread.h>

int arr[20],i,j;

int sum=0;

sem\_t mutex;

void \*runner(void \*arg)

{

sem\_wait(&mutex);

sleep(1);

int n;

printf("Enter a integer want to sum up: \n");

scanf("%d",&n);

sum=0;

for(int i=1; i<=n; i++)

{

sum+=i;

}

printf("summation: %d\n",sum);

sem\_post(&mutex);

}

int main()

{

pthread\_t thread[20];

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

for(int i=0;i<5;i++)

{

int t=pthread\_create(&thread[i],NULL,&runner,NULL);

if((t!=0))

{

printf("Thread couldn't create %d\n",t);

}

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

}

sem\_destroy(&mutex);

}

Output: Text

Description automatically generated