**LAB 4 ASSIGNMENT**

**Name:** Birva Babaria

**Roll no.:** CE010

**ID:** 19CEUON064

**Aim:** Thread creation and Termination. Synchronization using mutex lock and unlock. (Use of pthread\_create, ptread\_join library functions of Pthread library).

**Description:**

1. **‘pthread\_create’**

**Library:** #include<pthread.h>

**Syntax:** int pthread\_create(pthread\_t \*thread, const pthread\_attr\_t \*attr, void \*(\*start\_routine) (void \*), void \*arg);

**Description:** pthread\_create() function creates a new thread in the current calling process. To strart the execution of the new thread start\_routine() function is invoked. The arg argument passes the argument to the thread function. If arg is NULL then the thread is created with the default attributes.

On success, pthread\_create function returns 0. On error, it returns an error number and the contents of the thread are undefined.

**Example:** int status = pthread\_create(&thread, NULL, hello, NULL);

1. **‘pthread\_join’**

**Library:** #include<pthread.h>

**Syntax:** int pthread\_join(pthread\_t thread, void \*\*retval);

**Description:** pthread\_join() function waits for the thread specified by the thread to terminate. When the thread terminates, the pthread\_join() function returns immediately.

**Example:** pthread\_join(thread,NULL);

**1). Write a program to create a thread using pthread\_create.**

**CODE:**

//display hello from thread function

#include<stdio.h>

#include<pthread.h>

void \*hello()

{

printf("Hello from thread!!\n");

}

void main()

{

pthread\_t thread;

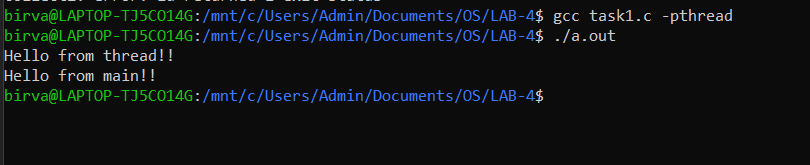
int status = pthread\_create(&thread, NULL, hello, NULL);

pthread\_join(thread,NULL);

printf("Hello from main!!\n");

}

**OUTPUT:**



**2). Write a program to call two different thread functions from different threads.**

**CODE:**

//create two threads and each thread should invoke different function

#include<stdio.h>

#include<pthread.h>

void \*hello1()

{

printf("Hello from thread one!!\n");

}

void \*hello2()

{

printf("Hello from thread two!!\n");

}

void main()

{

pthread\_t thread;

int status1 = pthread\_create(&thread, NULL, hello1, NULL);

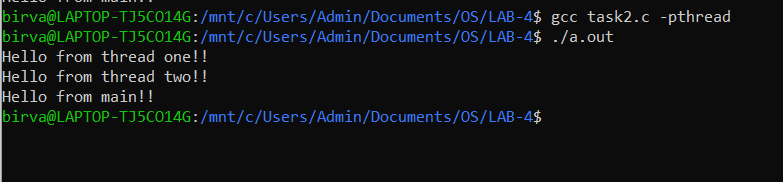
int status2 = pthread\_create(&thread, NULL, hello2, NULL);

pthread\_join(thread,NULL);

printf("Hello from main!!\n");

}

**OUTPUT:**



**3). Write a program to pass a character string to the threaded function.**

**CODE:**

//pass a character string to the threaded function

#include<stdio.h>

#include<pthread.h>

void \*func(void \*arg)

{

char \*str = (char \*)arg;

printf("Argument is: %s\n",str);

}

void main()

{

pthread\_t thread;

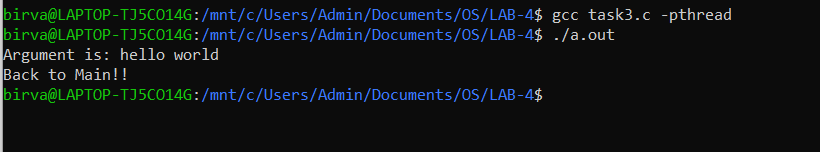
int status = pthread\_create(&thread, NULL, func, (void \*)"hello world");

pthread\_join(thread,NULL);

printf("Back to Main!!\n");

}

**OUTPUT:**

****

**4). Write a program pass multiple arguments to the threaded function using struct.**

**CODE:**

//pass multiple arguments to the threaded function using struct

#include<stdio.h>

#include<pthread.h>

struct mystruct{

int a;

int b;

};

void \*add(void \*arg)

{

struct mystruct \*strct = (struct mystruct \*)arg;

int sum = strct->a + strct->b;

printf("a: %d and b: %d\n",strct->a,strct->b);

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

}

void main()

{

pthread\_t thread;

struct mystruct \*str;

str->a = 10;

str->b = 20;

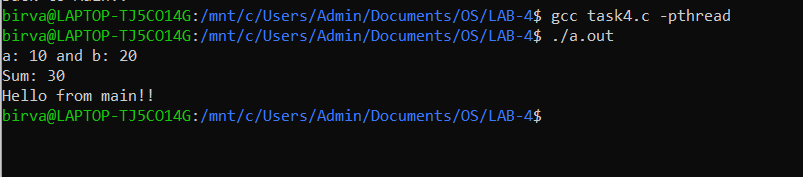
int status = pthread\_create(&thread, NULL, add, (void \*)str);

pthread\_join(thread,NULL);

printf("Hello from main!!\n");

}

**OUTPUT:**

****

**5). Write a program to implement simple calculator using threads.**

**CODE:**

//construct simple calculator

#include<stdio.h>

#include<pthread.h>

struct mystruct{

int a;

int b;

};

void \*add(void \*arg)

{

struct mystruct \*strct = (struct mystruct \*)arg;

int sum = strct->a + strct->b;

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

}

void \*sub(void \*arg)

{

struct mystruct \*strct = (struct mystruct \*)arg;

int sub = strct->a - strct->b;

printf("Subtraction: %d\n",sub);

}

void \*mul(void \*arg)

{

struct mystruct \*strct = (struct mystruct \*)arg;

int mul = strct->a \* strct->b;

printf("Multiplication: %d\n",mul);

}

void \*div(void \*arg)

{

struct mystruct \*strct = (struct mystruct \*)arg;

int div = strct->a / strct->b;

printf("Division: %d\n",div);

}

void main()

{

pthread\_t thread;

struct mystruct \*str;

int a,b;

printf("Enter a: ");

scanf("%d",&a);

printf("Enter b: ");

scanf("%d",&b);

str->a = a;

str->b = b;

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

int status1 = pthread\_create(&thread, NULL, add, (void \*)str);

int status2 = pthread\_create(&thread, NULL, sub, (void \*)str);

int status3 = pthread\_create(&thread, NULL, mul, (void \*)str);

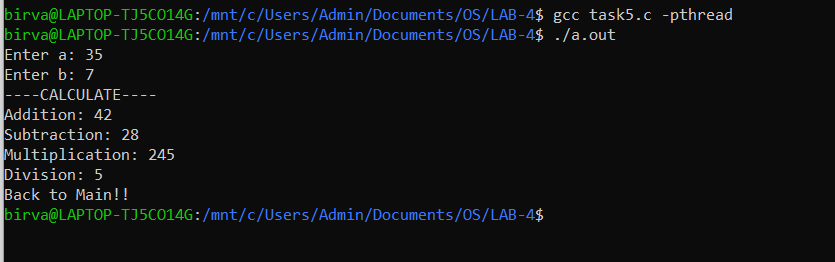
int status4 = pthread\_create(&thread, NULL, div, (void \*)str);

pthread\_join(thread,NULL);

printf("Back to Main!!\n");

}

**OUTPUT:**

****

**6). Write a program to multiply two matrices.**

**CODE:**

//multiply two matrices (4 X 4)

#include<stdio.h>

#include<pthread.h>

int a[4][4] = {{5,7,9,10},

{2,3,3,8},

{8,10,2,3},

{3,3,4,8}

};

int b[4][4] = {{3,10,12,18},

{12,1,4,9},

{9,10,12,2},

{3,12,4,10}

};

void \*mul(void \*arg)

{

int i = \*(int \*)arg;

int arr[4] = {0};

for(int j=0;j<4;j++)

{

for(int k=0;k<4;k++)

{

arr[j] += (a[i][k] \* b[k][j]);

}

}

for(int j=0;j<4;j++)

{

printf(" %3d ",arr[j]);

}

printf("\n");

}

void main()

{

pthread\_t thread;

int a=0,b=1,c=2,d=3;

printf("----MATRIX MULTIPLICATION (4 X 4)----\n");

int status1 = pthread\_create(&thread, NULL, mul, (void \*)&a);

pthread\_join(thread,NULL);

int status2 = pthread\_create(&thread, NULL, mul, (void \*)&b);

pthread\_join(thread,NULL);

int status3 = pthread\_create(&thread, NULL, mul, (void \*)&c);

pthread\_join(thread,NULL);

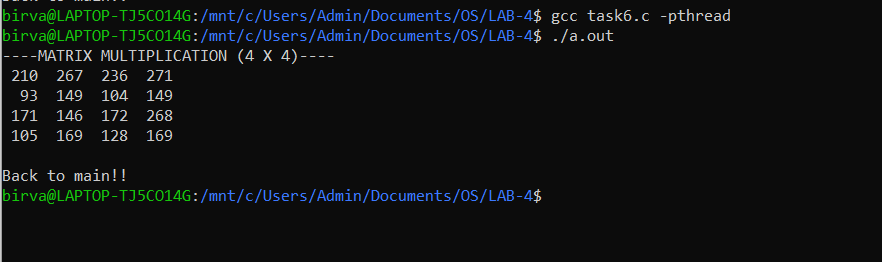
int status4 = pthread\_create(&thread, NULL, mul, (void \*)&d);

pthread\_join(thread,NULL);

printf("\nBack to main!!\n");

}

**OUTPUT:**

****