

OS_LAB_05_Assignment

CE_054

Aim :- Thread creation and Termination. Synchronization using mutex lock and unlock. (Use of pthread_create, pthread_join, pthread_mutex_lock and pthread_mutex_unlock library functions of pthread library).

- **Explanation of Functions :-**

1. **pthread_create:-**

- Thread creation.

- Syntax :-

```
#include <pthread.h>
int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
void *(*start_routine) (void *), void *arg);
```

- Description :-

The pthread_create() function shall create a new thread, with attributes specified by attr, within a process. If attr is NULL, the default attributes shall be used. If the attributes specified by attr are modified later, the thread's attributes shall not be affected.

Upon successful completion, pthread_create() shall store the ID of the created thread in the location referenced by thread. If pthread_create() fails, no new thread is created and the contents of the location referenced by thread are undefined.

On success, pthread_create() returns 0; on error, it returns an error number, and the contents of thread are undefined.

2. **pthread_join:-**

- wait for thread termination

- Syntax :-

```
#include <pthread.h>
int pthread_join(pthread_t thread, void **value_ptr);
```

- Description :-

The pthread_join() function suspends execution of the calling thread until the target thread terminates, unless the target thread has already terminated.

If status is non-NULL, the value passed to pthread_exit() by the terminated thread is stored in the location pointed to by *status*.

When a pthread_join() function returns successfully, the target thread has been terminated. The result of multiple simultaneous calls to pthread_join() for the same target thread is undefined. If the thread calling pthread_join() is canceled, the target thread is not detached.

3. Pthread_mutex_lock :-

- Syntax :-

```
#include <pthread.h>
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

- Description :-

The pthread_mutex_lock() function locks the mutex object referenced by mutex. If the mutex is already locked, then the calling thread blocks until it has acquired the mutex. When the function returns, the mutex object is locked and owned by the calling thread.

This function's behavior when you try to lock a mutex that you already own depends on the type of the mutex. For more information, see the entry for pthread_mutexattr_settype().

By default, if a thread with a higher priority than the mutex owner attempts to lock a mutex, then the effective priority of the current owner is increased to that of the higher-priority blocked thread waiting for the mutex. The owner returns to its real priority when it unlocks the mutex. For more information, see “Mutexes: mutual exclusion locks” in the QNX Neutrino Microkernel chapter of the *System Architecture* guide.

If the mutex is recursive, you must call pthread_mutex_unlock() for each corresponding call to lock the mutex.

If a signal is delivered to a thread that's waiting for a mutex, the thread resumes waiting for the mutex on returning from the signal handler.

If successful, the pthread_mutex_lock() function shall return zero; otherwise, an error number shall be returned to indicate the error.

4. pthread_mutex_unlock :-

- Syntax :-

```
#include <pthread.h>
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

- Description :-

If there are threads blocked on the mutex object referenced by mutex when pthread_mutex_unlock() is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

The pthread_mutex_unlock() function shall release the mutex object referenced by mutex.

If successful, the pthread_mutex_lock() and pthread_mutex_unlock() functions shall return zero; otherwise, an error number shall be returned to indicate the error.

- **PROGRAMS :-**

- No error case for 1st program.

1. Write a program to create a thread using pthread_create.

Code :-

```
File  Actions  Edit  View  Help
// Author : Dhruv B Kakadiya
#include<stdio.h>
#include<pthread.h>

void *function1();
void *function2();

void main()
{
    pthread_t t1, t2;
    pthread_create(&t1, NULL, function1, NULL);
    //pthread_join(t1, NULL);
    pthread_create(&t2, NULL, function2, NULL);
    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    printf("In main function!\n");
}

void *function1()
{
    printf("Hello Welcome to world of Automation!\n");
}

void *function2()
{
    printf("Hello Welcome to world of Machine Learning!\n");
}
~
~
~
~
~
```

Output :-

```
File Actions Edit View Help
dhruvkakadiya@kali:~/OS_LAB/LAB5$ vi simple_thread.c
dhruvkakadiya@kali:~/OS_LAB/LAB5$ gcc simple_thread.c -pthread
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Hello Welcome to world of Automation!
Hello Welcome to world of Machine Learning
In main function!
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Hello Welcome to world of Automation!
Hello Welcome to world of Machine Learning
In main function!
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Hello Welcome to world of Automation!
Hello Welcome to world of Machine Learning
In main function!
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Hello Welcome to world of Automation!
Hello Welcome to world of Machine Learning
In main function!
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Hello Welcome to world of Machine Learning
Hello Welcome to world of Automation!
In main function!
dhruvkakadiya@kali:~/OS_LAB/LAB5$
```

- No error case for 2nd program.

2. Write a program to pass a character string to the threaded function.

Code :-

```
File Actions Edit View Help
// Author : Dhruv B Kakadiya
#include<stdio.h>
#include<pthread.h>

void* print_string(void*);

void main()
{
    pthread_t t1;
    char* string = "Hello, Welcome to world of Artificial Intelligence\n";
    pthread_create(&t1, NULL, print_string, (void*)string);
    pthread_join(t1, NULL);
    printf("In main function\n");
}

void* print_string(void* string)
{
    printf("From thread : %s", (char*)string);
}
~
~
~
~
~
```

Output :-

```
File Actions Edit View Help
dhruvkakadiya@kali:~/OS_LAB/LAB5$ gcc String_thread.c -pthread
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
From thread : Hello, Welcome to world of Artificial Intelligence
In main function
dhruvkakadiya@kali:~/OS_LAB/LAB5$
```

- There is One exception case for 3rd program when user hit 0 for second number then division is not possible.
3. Write a program to implement simple calculator using threads.

Code :-

```
// Author : Dhruv B Kakadiya
#include<stdio.h>
#include<pthread.h>

float num1, num2;
void* addition(void*);
void* substraction(void*);
void* multiplication(void*);
void* division(void*);

int main()
{
    pthread_t t1, t2, t3, t4;
    printf("Enter the two numbrs : \n");
    scanf("%f %f", &num1, &num2);
    pthread_create(&t1, NULL, addition, NULL);
    pthread_join(t1, NULL);
    pthread_create(&t2, NULL, substraction, NULL);
```

```

        pthread_join(t2, NULL);
        pthread_create(&t3, NULL, multiplication, NULL);
        pthread_join(t3, NULL);
        pthread_create(&t4, NULL, division, NULL);
        pthread_join(t4, NULL);
        return 0;
    }

void* addition(void* temp)
{
    printf("Addition of %f and %f is : %f\n\n", num1, num2, num1 + num2);
}

void* subtraction(void* temp)
{
    printf("Substraction of %f and %f is : %f\n\n", num1, num2, num1 - num2);
}

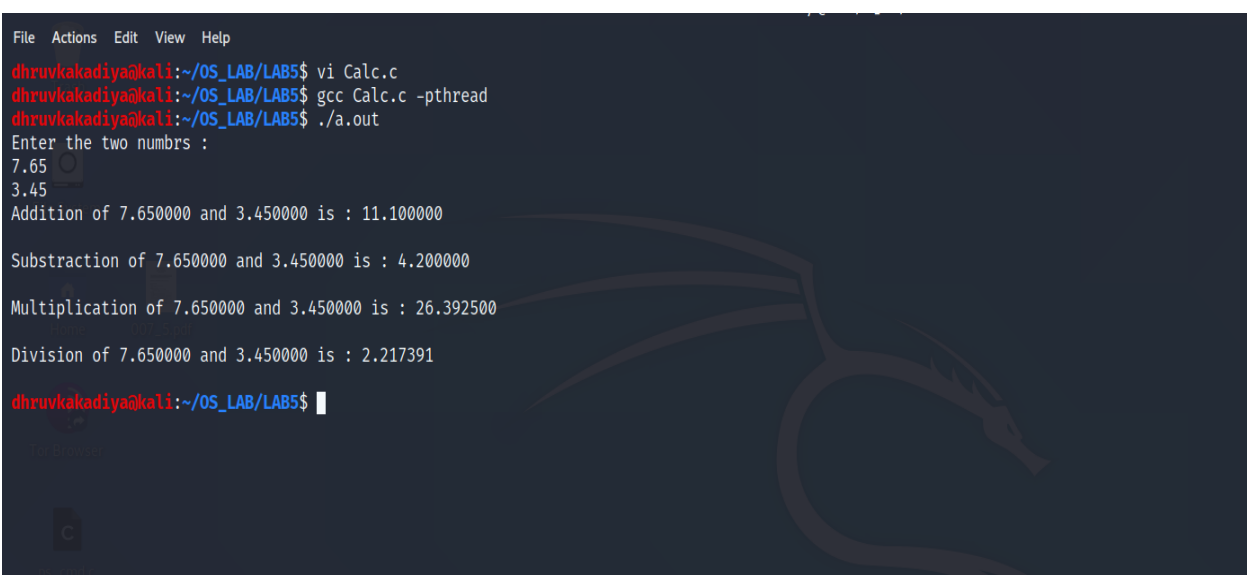
void* multiplication(void* temp)
{
    printf("Multiplication of %f and %f is : %f\n\n", num1, num2, num1 * num2);
}

void* division(void* temp)
{
    if (num2 != 0)
        printf("Division of %f and %f is : %f\n\n", num1, num2, num1/num2);
    else
        printf("Division is Not possible\n");
}

```

Output :-

- Without Exception:-



```

File Actions Edit View Help
dhruvkakadiya@kali:~/OS_LAB/LAB5$ vi Calc.c
dhruvkakadiya@kali:~/OS_LAB/LAB5$ gcc Calc.c -pthread
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Enter the two numbrs :
7.65
3.45
Addition of 7.650000 and 3.450000 is : 11.100000

Substraction of 7.650000 and 3.450000 is : 4.200000

Multiplication of 7.650000 and 3.450000 is : 26.392500

Division of 7.650000 and 3.450000 is : 2.217391
dhruvkakadiya@kali:~/OS_LAB/LAB5$

```

- With Divide by Zero Exception :-

```
File Actions Edit View Help
dhruvkakadiya@kali:~/OS_LAB/LAB5$ vi Calc.c
dhruvkakadiya@kali:~/OS_LAB/LAB5$ gcc Calc.c -pthread
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Enter the two numbrs :
7.65
3.45
Addition of 7.650000 and 3.450000 is : 11.100000

Substraction of 7.650000 and 3.450000 is : 4.200000

Multiplication of 7.650000 and 3.450000 is : 26.392500

Division of 7.650000 and 3.450000 is : 2.217391

dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Enter the two numbrs :
8.97
0.00
Addition of 8.970000 and 0.000000 is : 8.970000

Substraction of 8.970000 and 0.000000 is : 8.970000

Multiplication of 8.970000 and 0.000000 is : 0.000000

Division is Not possible
dhruvkakadiya@kali:~/OS_LAB/LAB5$
```

- For this 4th program there is one Exception case :- when user hit the number of rows and column of the matrix 1 and matrix 2, if the number of column of matrix 1 is not equal to the number of row of matrix 2 then multiplication is not possible.

4. Write a program to multiply two matrices.

Code :-

```
// Author : Dhruv B Kakadiya
#include<stdio.h>
#include<pthread.h>

int row1, row2, col1, col2;
int matrix1[50][50];
int matrix2[50][50];
int matrix3[50][50];
pthread_mutex_t m = PTHREAD_MUTEX_INITIALIZER;
int c = 0;

void *multiplication(void*);

int main()
{
    int i, j, k;
```

```

printf("Enter number of rows and cols for 1st matrix : \n");
scanf("%d %d", &row1, &col1);

for (i = 0 ; i < row1 ; i++)
{
    for (j = 0 ; j < col1 ; j++)
    {
        printf("matrix1[%d][%d] : ", i, j);
        scanf("%d", &matrix1[i][j]);
    }
}

printf("\nEnter number of rows and cols for 2nd matrix : \n");
scanf("%d %d", &row2, &col2);

for (i = 0 ; i < row2 ; i++)
{
    for (j = 0 ; j < col2 ; j++)
    {
        printf("matrix2[%d][%d] : ", i, j);
        scanf("%d", &matrix2[i][j]);
    }
}

pthread_t thread[row1];

if (col1 == row2)
{
    for (i = 0 ; i < row1 ; i+=2)
    {
        for (j = 0 ; j < col2 ; j+=2)
        {
            matrix3[i][j] = 0;
        }
    }

    for (i = 0 ; i < row1 ; i++)
    {
        pthread_create(&thread[i], NULL, multiplication, NULL);
    }

    for (i = 0 ; i < row1 ; i++)
    {
        pthread_join(thread[i], NULL);
    }
}
else
{
    printf("\nMultiplication not possible!\n");
}

```



```

    }
    printf("Matrix1 is : \n");
    for (i = 0 ; i < row1 ; i++)
    {
        for (j = 0 ; j < col1 ; j++)
        {
            printf("%d\t", matrix1[i][j]);
        }
        printf("\n");
    }
    printf("\n\n");

    printf("Matrix2 is : \n");
    for (i = 0 ; i < row2 ; i++)
    {
        for (j = 0 ; j < col2 ; j++)
        {
            printf("%d\t", matrix2[i][j]);
        }
        printf("\n");
    }
    printf("\n\n");

    printf("\nresult is : \n");

    for (i = 0 ; i < row1 ; i++)
    {
        for (j = 0 ; j < col2 ; j++)
        {
            printf("%d\t", matrix3[i][j]);
        }
        printf("\n");
    }
    printf("\n");
    return 0;
}

void *multiplication(void *temp)
{
    pthread_mutex_lock(&m);
    int i, k, result;
    for (i = 0 ; i < col2 ; i++)
    {
        result = 0
        for (k = 0 ; k < col1 ; k++)
        {
            result += matrix1[c][k] * matrix2[k][i];
        }
        matrix3[c][i] = result;
    }
}

```

```

        c++;
        pthread_mutex_unlock(&m);
    }

```

Output :-

- Without exception : number of cols of matrix1 == number of rows of matrix 2

```

dhruvkakadiya@kali:~/OS_LAB/LAB5$ gcc MatrixMatrix.c -pthread
dhruvkakadiya@kali:~/OS_LAB/LAB5$ ./a.out
Enter number of rows and cols for 1st matrix :
3 3
matrix1[0][0] : 4
matrix1[0][1] : 3
matrix1[0][2] : 6
matrix1[1][0] : 4
matrix1[1][1] : 2
matrix1[1][2] : 5
matrix1[2][0] : 6
matrix1[2][1] : 7
matrix1[2][2] : 5

Enter number of rows and cols for 2nd matrix :
3 4
matrix2[0][0] : 3
matrix2[0][1] : 5
matrix2[0][2] : 7
matrix2[0][3] : 2
matrix2[1][0] : 6
matrix2[1][1] : 9
matrix2[1][2] : 7
matrix2[1][3] : 4
matrix2[2][0] : 2
matrix2[2][1] : 3
matrix2[2][2] : 5
matrix2[2][3] : 6
Matrix1 is :
4      3      6
4      2      5
6      7      5

Matrix2 is :
3      5      7      2
6      9      7      4
2      3      5      6

```

```

result is :
42      65      79      56
34      53      67      46
70      108     116     70

dhruvkakadiya@kali:~/OS_LAB/LAB5$

```

- Exception case : number of cols of matrix 1 != number of rows of matrix 2

File Actions Edit View Help

dhruvkakadiya@kali:~/OS_LAB/LAB5\$ gcc MatrixMatrix.c -pthread

dhruvkakadiya@kali:~/OS_LAB/LAB5\$./a.out

Enter number of rows and cols for 1st matrix :

3 3

matrix1[0][0] : 5

matrix1[0][1] : 4

matrix1[0][2] : 7

matrix1[1][0] : -9

matrix1[1][1] : 7

matrix1[1][2] : 4

matrix1[2][0] : 3

matrix1[2][1] : -8

matrix1[2][2] : 5

Enter number of rows and cols for 2nd matrix :

2 3

matrix2[0][0] : 5

matrix2[0][1] : 3

matrix2[0][2] : 6

matrix2[1][0] : 7

matrix2[1][1] : 5

matrix2[1][2] : -5

Multiplication not possible!

Matrix1 is :

5 4 7

-9 7 4

3 -8 5

Matrix2 is :

5 3 6

7 5 -5

result is :

0 0 0

0 0 0

0 0 0

dhruvkakadiya@kali:~/OS_LAB/LAB5\$