



Assignment 05 | Operating System

CE-092

Assignment submission for Operating System subject week 5.

nevilparmar24@gmail.com

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).

Pthread_create:

```
#include <pthread.h>
```

```
int pthread_create(pthread_t *thread, const  
pthread_attr_t *attr,
```

```
void *(*start_routine) (void *), void *arg);
```

The `pthread_create()` function starts a new thread in the calling process. The new thread starts execution by invoking `start_routine()`; `arg` is passed as the sole argument of `start_routine()`.

The `attr` argument points to a `pthread_attr_t` structure whose contents are used at thread creation time to determine attributes for the new thread; this structure is initialized using `pthread_attr_init` and related functions. If `attr` is NULL, then the thread is created with default attributes.

Before returning, a successful call to `pthread_create()` stores the ID of the new thread in the buffer pointed to by `thread`; this identifier is used to refer to the thread in subsequent calls to other pthreads functions.

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

Pthread_join:

```
#include <pthread.h>
```

```
int pthread_join(pthread_t thread, void **retval);
```

Compile and link with *-pthread*.

The `pthread_join()` function waits for the thread specified by *thread* to terminate. If that thread has already terminated, then `pthread_join()` returns immediately. The thread specified by *thread* must be joinable.

On success, `pthread_join()` returns 0; on error, it returns an error number.

Pthread_mutex_lock:

```
#include <pthread.h>
```

```
int pthread_mutex_lock(pthread_mutex_t *mutex);
```

The mutex object referenced by *mutex* shall be locked by calling `pthread_mutex_lock()`. If the mutex is already locked, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by *mutex* in the locked state with the calling thread as its owner.

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.

Pthread_mutex_unlock:

```
#include <pthread.h>
```

```
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

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

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

Task 1:

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

Code:

```
#include<stdio.h>
#include<pthread.h>

void * f1()
{
    printf("Hello from thread\n");
}

int main()
{
    pthread_t t1;
    pthread_create(&t1, NULL , f1, NULL);
    // pthread_join(t1, NULL);

    return 0 ;
}

/*
```

It prints nothing because we are not making the main thread to wait for this t1 thread.

Hence the output got stuck in the buffer and nothing gets printed on the terminal

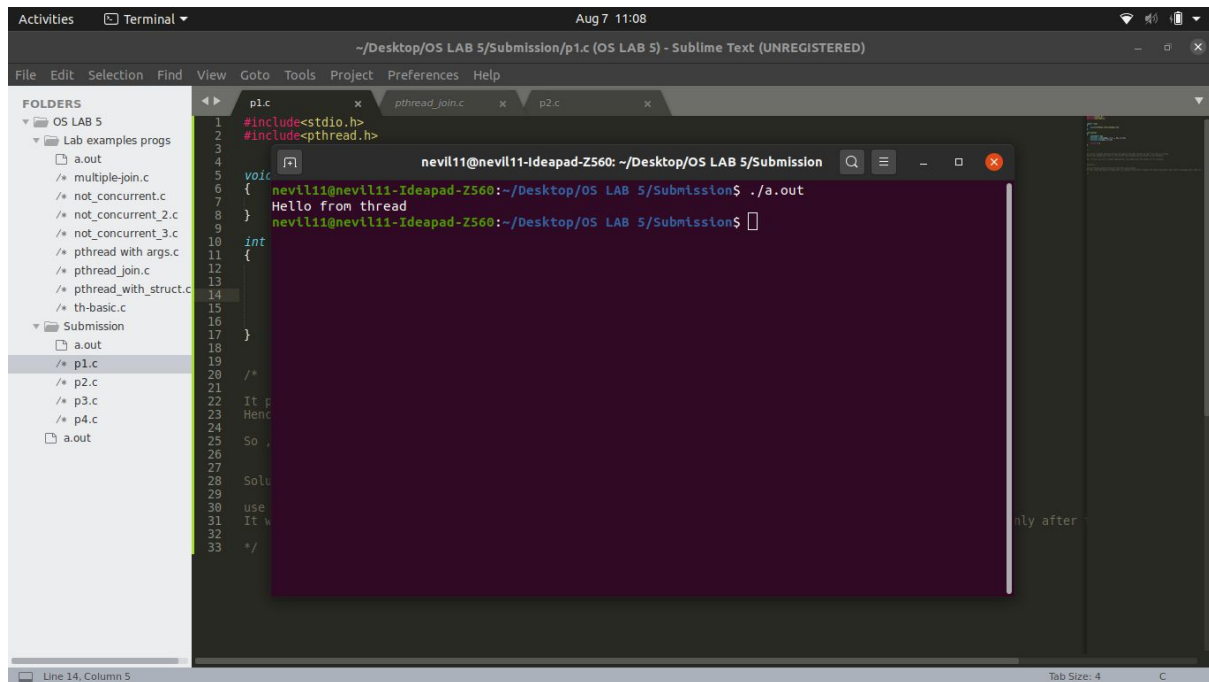
So , if you run it 5 6 times continuously, you might see the output of f1 function.

Solution :

use of pthread_join() function from the same library. It will make the parent thread wait to execute first child thread and hence the parent will start executing only after the child gets destroyed.

** /*

Output:



The screenshot shows a terminal window with the following content:

```
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission$ ./a.out
Hello from thread
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission$
```

Task 2:

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

Code:

```
#include<stdio.h>
#include<pthread.h>
```

```

void * fun(void *str)
{
    printf("Passed String: %s\n",str);
    return NULL;
}

int main()
{
    pthread_t newth1;
    char *s = "I am a string passed as arg.";
    pthread_create(&newth1,NULL,fun,s);
    pthread_join(newth1,NULL);

    printf("I am the last line in main.\n");
    return 0;
}

```

Output:

The screenshot shows a terminal window with the following output:

```

nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$ gcc -pthread p2.c
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$ ./a.out
Reply from thread : Passed message to thread !
Parent (Here main thread, i.e main method) Thread joins
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$

```

The background shows the Sublime Text editor with the source code of the program, and a file explorer on the left showing the project structure.

Task 3:

Write a program to implement a simple calculator using threads.

Code:

```
// simple calc

#include<stdio.h>
#include<pthread.h>
#include<time.h>

struct nums
{
    int x;
    int y;
};

void *add(void *numsref)
{
    printf("Add: %d\n", ((struct nums*)numsref)->x +
    ((struct nums*)numsref)->y);
    return NULL;
}

void *sub(void *numsref)
{
    printf("Sub: %d\n", ((struct nums*)numsref)->x -
    ((struct nums*)numsref)->y);
    return NULL;
}

void *mul(void *numsref)
{

```

```
    printf("Mul: %d\n", ((struct nums*)numsref)->x *
((struct nums*)numsref)->y);
    return NULL;
}

void *div(void *numsref)
{
    printf("Div: %d\n", ((struct nums*)numsref)->x /
((struct nums*)numsref)->y);
    return NULL;
}

int main()
{
    // Calculate the time taken by fun()
    clock_t t;
    t = clock();

    int i;

    struct nums n;
    n.x = 10;
    n.y = 20;

    pthread_t threads[4];

    // array of function pointers
    void * f[4];
    f[0] = add;
    f[1] = sub;
    f[2] = mul;
    f[3] = div;
```

```
    for(i=0;i<4;i++)
    {
        pthread_create(&threads[i],NULL,f[i],&n);
    }

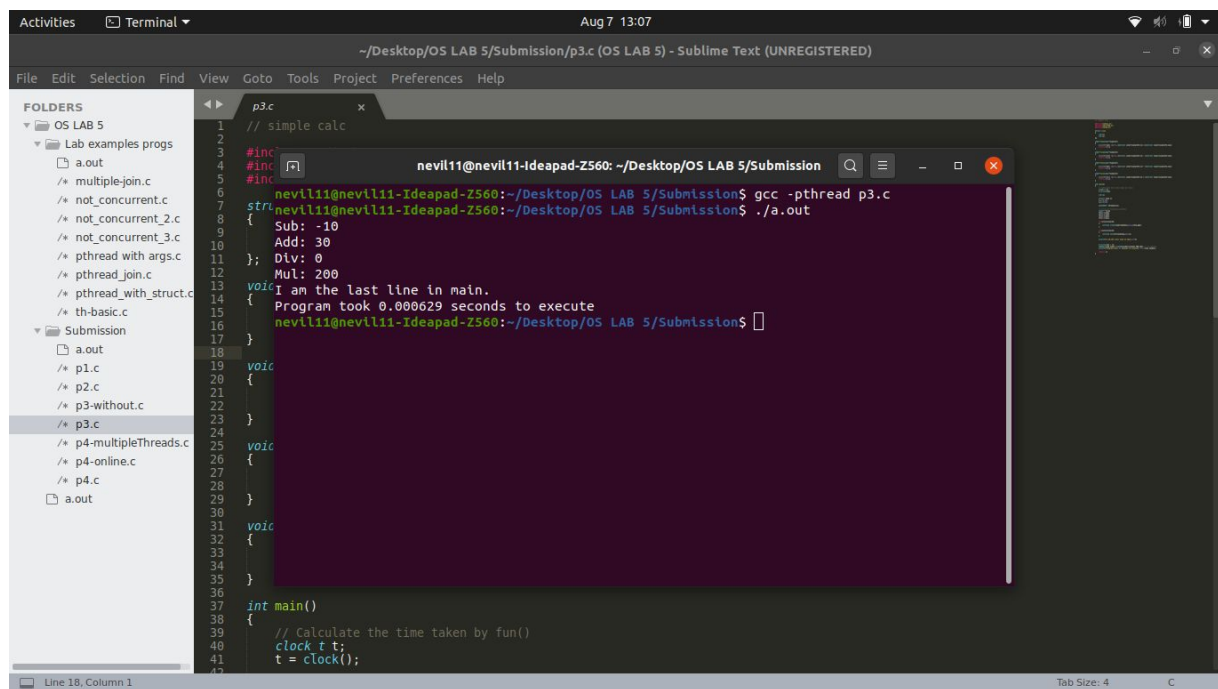
    for(i=0;i<4;i++)
    {
        pthread_join(threads[i],NULL);
    }

    printf("I am the last line in main.\n");

    t = clock() - t;
    double time_taken = ((double)t)/CLOCKS_PER_SEC; //
in seconds
    printf("Program took %f seconds to execute \n",
time_taken);

    return 0;
}
```


Output:



The screenshot shows a Sublime Text editor window titled "p3.c (OS LAB 5) - Sublime Text (UNREGISTERED)". The editor displays a C program named p3.c. The program includes headers for stdio.h, pthread.h, and stdlib.h. It defines three 10x10 integer matrices: MAT1, MAT2, and MAT3. It also defines four integer variables: r1, c1, r2, and c2. A function pointer *multiply is declared, taking a void* argument. The main function is partially visible, starting with a comment to calculate the time taken by fun(). A terminal window is overlaid on the editor, showing the compilation and execution of the program. The terminal output shows the program's execution, including the calculation of the time taken by fun() and the final output of the program.

```
1 // simple calc
2
3 #include <stdio.h>
4 #include <pthread.h>
5 #include <stdlib.h>
6
7 int MAT1[10][10];
8 int MAT2[10][10];
9 int MAT3[10][10];
10
11 int r1,c1,r2,c2;
12
13 void *multiply(void *);
14
15 int main()
16 {
17     // Calculate the time taken by fun()
18     clock_t t;
19     t = clock();
20 }
```

Terminal Output:

```
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission$ gcc -pthread p3.c
nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission$ ./a.out
Sub: -10
Add: 30
Div: 0
Mul: 200
I am the last line in main.
Program took 0.000629 seconds to execute
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$
```

Task 4:

Write a program to multiply two matrices.

Code:

```
# include <stdio.h>
# include <pthread.h>
#include<stdlib.h>

int MAT1[10][10];
int MAT2[10][10];
int MAT3[10][10];

int r1,c1,r2,c2;

void *multiply(void *);

int main()
```

```
{

pthread_t tid;
int i,j,kCount;

printf("Enter Number of Rows For Matrix 1 :");
scanf("%d",&r1);

printf("Enter Number of Columns For Matrix 1 :");
scanf("%d",&c1);

for(i=0;i<r1;i++)
    for(j=0;j<c1;j++)
        MAT1[i][j] = rand() %10;

printf("Enter Numer of Rows For Matrix 2 :");
scanf("%d",&r2);

printf("Enter Number of Columns For Matrix 2 :");
scanf("%d",&c2);

for(i=0;i<r2;i++)
    for(j=0;j<c2;j++)
        MAT2[i][j] = rand() %10;

if(c1!=r2)
{
    printf("Multipication of Matrix not Possible
!!!");
}
else
{

```

```

        for(i=0;i<r1;i=i+2)
        {
            for(j=0;j<c2;j=j+2)
            {
                MAT3[i][j]=0;
            }
        }

pthread_create(&tid,NULL,multiply,NULL);

for(i=0;i<r1;i=i+2)
{
    for(j=0;j<c2;j++)
    {
        for(kCount=0;kCount<c1;kCount++)
        {
            MAT3[i][j]+=MAT1[i][kCount] *
MAT2[kCount][j];
        }
    }
}

pthread_join(tid,NULL);

printf("\nMatrix 1 \n");

for(i=0;i<r1;i++)
{
    for(j=0;j<c1;j++)
    {
        printf("%d \t",MAT1[i][j]);
    }
}

```

```

    }
    printf("\n");
}

printf("\nMatrix 2 \n");

for(i=0;i<r2;i++)
{
    for(j=0;j<c2;j++)
    {
        printf("%d \t",MAT2[i][j]);
    }
    printf("\n");
}

printf("\nMultiplication of Matrix ...\n");

for(i=0;i<r1;i++)
{
    for(j=0;j<c2;j++)
    {
        printf("%d \t",MAT3[i][j]);
    }
    printf("\n");
}
return 0;
}

void *multiply(void *para)
{
    int i,j,kCount;
    for(i=1;i<r1;i=i+2)

```

```

    {
        for(j=0;j<c2;j++)
        {
            for(kCount=0;kCount<c1;kCount++)
            {
                MAT3[i][j] += MAT1[i][kCount] *
MAT2[kCount][j];
            }
        }
    }

pthread_exit(NULL);
}

```

Output:

A terminal window titled "nevil11@nevil11-Ideapad-Z560: ~/Desktop/OS LAB 5/Submission" showing the execution of a program. The program prompts for the number of rows and columns for two matrices. Matrix 1 is 3x4 and Matrix 2 is 4x3. The program then displays the multiplication of these matrices, resulting in a 3x3 matrix.

```

nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$ ./a.out
Enter Number of Rows For Matrix 1 :3
Enter Number of Columns For Matrix 1 :4
Enter Number of Rows For Matrix 2 :4
Enter Number of Columns For Matrix 2 :3

Matrix 1
3      6      7      5
3      5      6      2
9      1      2      7

Matrix 2
0      9      3
6      0      6
2      6      1
8      7      9

Multiplication of Matrix ...
90     104    97
58     77     63
66     142    98
nevil11@nevil11-Ideapad-Z560:~/Desktop/OS LAB 5/Submission$

```