

Submitted by:

Name: Aditi Sudhir Igade

Batch: A3

Roll no.:221081

PRN No.:22220311

Assignment 5

Aim : Thread management using pthread library. Implement matrix multiplication using multithreading. Application should have pthread_create, pthread_join, pthread_exit. In the program, every thread must return the value and must be collected in pthread_join in the main function. Final sum of row column multiplication must be done by main thread (main function).

Theory :

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.

Process	Thread
A process is an instance of a program that is being executed or processed.	Thread is a segment of a process or a lightweight process that is managed by the scheduler independently.
Processes are independent of each other and hence don't share a memory or other resources.	Threads are interdependent and share memory.
Each process is treated as a new process by the operating system.	The operating system takes all the user-level threads as a single process.
If one process gets blocked by the operating system, then the other process can continue the execution.	If any user-level thread gets blocked, all of its peer threads also get blocked because OS takes all of them as a single process.
Context switching between two processes takes much time as they are heavy compared to thread.	Context switching between the threads is fast because they are very lightweight.
The data segment and code segment of each process are independent of the other.	Threads share data segment and code segment with their peer threads; hence are the same for other threads also.

The operating system takes more time to terminate a process.	Threads can be terminated in very little time.
New process creation is more time taking as each new process takes all the resources.	A thread needs less time for creation.

Pthreads() : POSIX Threads are commonly known as **PThreads**. It is an execution model that exists independently from a language and a parallel execution model. It allows a program to control multiple different workflows that overlap in time. Each flow of work is referred to as a **thread**. Creation and controlling these flows is achieved by making calls to the POSIX Threads API. POSIX Threads is an API defined by the standard POSIX.1c, Threads extensions

Code :

```
#include <bits/stdc++.h>
#include<pthread.h>
using namespace std;

#define MAX 3

#define MAX_THREAD 3

//three matrix are created
int matA[MAX][MAX];
int matB[MAX][MAX];
int matC[MAX][MAX];
int step_i = 0;

//accepting matrix's operations in other matrix
void* multi(void* arg)
{
    int i = step_i++;

    for (int j = 0; j < MAX; j++)
        for (int k = 0; k < MAX; k++)
            matC[i][j] += matA[i][k] * matB[k][j];
    return 0;
}

int main()
{
    //rand() size is at least 32767.
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++) {
            matA[i][j] = rand() % 10;
            matB[i][j] = rand() % 10;
        }
    }

    cout << "Matrix A" << endl;
    for (int i = 0; i < MAX; i++) {
```

```

        for (int j = 0; j < MAX; j++)
            cout << matA[i][j] << " ";
        cout << endl;
    }

    cout << "Matrix B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matB[i][j] << " ";
        cout << endl;
    }

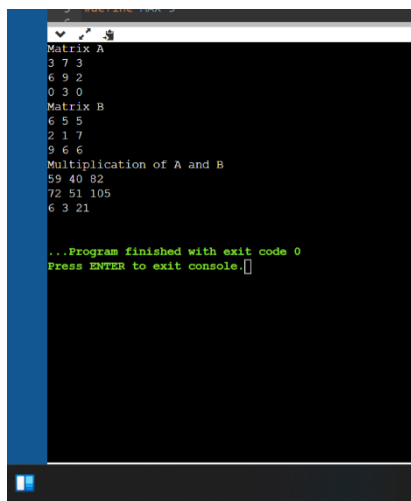
    pthread_t threads[MAX_THREAD];

    for (int i = 0; i < MAX_THREAD; i++) {
        int* p;
        pthread_create(&threads[i], NULL, multi, (void*)(p));
    }

    for (int i = 0; i < MAX_THREAD; i++){
        pthread_join(threads[i], NULL);
    }
    cout << "Multiplication of A and B" << endl;
    for (int i = 0; i < MAX; i++) {
        for (int j = 0; j < MAX; j++)
            cout << matC[i][j] << " ";
        cout << endl;
    }
    return 0;
}

```

Output :



```

Matrix A
3 7 3
6 9 2
0 3 0
Matrix B
6 5 5
2 1 7
9 6 6
Multiplication of A and B
59 40 82
72 51 105
6 3 21

...Program finished with exit code 0
Press ENTER to exit console.

```