**Assignment 05**

**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).

**Program :**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

#define SIZE 10

int A[SIZE][SIZE], B[SIZE][SIZE];

long int C[SIZE][SIZE];

void \*mul\_thread(void \*arg) {

int i, row, col, \*rcArgs;

long int return\_val; //long int, since int cannot be type casted to void

rcArgs = (int \*) arg;

row = rcArgs[0];

col = rcArgs[1];

i = rcArgs[2];

return\_val = A[row][i] \* B[i][col];

//return ((void \*) return\_val);

pthread\_exit((void \*) return\_val);

}

void accept\_matrix(int M[SIZE][SIZE], int rows, int cols) {

int i, j;

printf("\n");

for(i=0;i<rows;i++) {

for(j=0;j<cols;j++) {

printf("Value at [%d][%d]: ",i+1,j+1);

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

}

}

}

void display\_matrix(int M[SIZE][SIZE], int rows, int cols) {

int i, j;

printf("\n");

for(i=0;i<rows;i++){

for(j=0;j<cols;j++){

printf("%2d ",M[i][j]);

}

printf("\n");

}

}

int main() {

int rows\_A, cols\_A, rows\_B, cols\_B;

int rcArgs[3];

int i, j, k, \*status;

pthread\_t P[SIZE][SIZE][SIZE];

printf("\nEnter no. of rows in matrix A: ");

scanf("%d",&rows\_A);

printf("Enter no. of columns in matrix A: ");

scanf("%d",&cols\_A);

accept\_matrix(A, rows\_A, cols\_A);

printf("\nEnter no. of rows in matrix B: ");

scanf("%d",&rows\_B);

printf("Enter no. of columns in matrix B: ");

scanf("%d",&cols\_B);

accept\_matrix(B, rows\_B, cols\_B);

if(cols\_A == rows\_B) {

for(i=0;i<rows\_A;i++) {

for(j=0;j<cols\_B;j++) {

for(k=0;k<cols\_A;k++){

rcArgs[0] = i;

rcArgs[1] = j;

rcArgs[2] = k;

//Creating a new thread for every multiplication operation

if(pthread\_create(&P[i][j][k], NULL, mul\_thread, rcArgs) != 0)

printf("\nCannot create thread.\n");

else

//Inserting delay

sleep(1);

}

}

}

} else {

printf("\n Matrix multiplication not possible.");

exit(1);

}

printf("\nMatrix A:");

display\_matrix(A, rows\_A, cols\_A);

printf("\nMatrix B:");

display\_matrix(B, rows\_B, cols\_B);

for(i=0;i<rows\_A;i++) {

for(j=0;j<cols\_B;j++) {

for(k=0;k<cols\_A;k++){

//joining all the threads and retrieving values in status

if(pthread\_join(P[i][j][k],(void \*\*) &status) != 0)

perror("\nThread join failed.\n");

C[i][j] += (long int)status;

}

}

}

printf("\nResultant Matrix:\n");

for(i=0;i<rows\_A;i++){

for(j=0;j<cols\_B;j++){

printf("%2ld ",C[i][j]);

}

printf("\n");

}

exit(EXIT\_SUCCESS);

}

**Output :-**

