Assignment 1 factorial.c

WAP to print the factorial of a given number using multithreads

Pre-requisites:-

- Knowledge about multi-thread process, How to read and understand ‘man pages’.
- Good knowledge about pthread library functions.

Objective: -

- To understand working and flow of multithread programs.

Requirements: -

1. Modify the factorial template code using multiple threads.
2. Create at-least 3 threads and share the work among threads equally .
3. Create and join threads using separate loops .
4. You may have to change the argument structure.
5. Declare all integer variables as unsigned long int (For max values).

Sample execution: -

```
1. ./factorial
   Enter the number
   10
   factorial of 10 is 3628800
```

Assignment 2 sum_max.c

WAP to sum and maximum of a given array using multiple threads. Synchronize threads using mutex.

Pre-requisites:-

- Knowledge about multi-thread process, How to read and understand ‘man pages’.
- Good knowledge about pthread library functions.
- Thread synchronization (mutex and semaphores).

Objective: -

- To understand need and implementation of thread synchronization.

Requirements: -

1. Create an array[N] (min size 500) with 1 → N values and two global variables global_sum & global_max.
2. Create multiple threads to find sum of array and maximum value in array.
3. Threads will share array equally among them. Means each thread will access different part of array at same time.
4. Create at-least 5 threads. Create and join threads using separate loops.
5. Each thread will update global_sum and global_max.
6. Since we are accessing global variable from threads, do proper synchronization using mutex.

Sample execution: -

```
1. ./sum_max
   max = 100
   Sum = 1020
```

Hints:

- Don't use mutex lock and unlock inside loops
- Don't use loops between mutex lock and unlock .!!!

Assignment 3 matrix_mul.c

WAP to multiply two matrices using multiple threads

Pre-requisites:-

- Knowledge about multi-thread process, How to read and understand 'man pages'.
- Good knowledge about pthread library functions.
- Multiplication of two matrices.
- Dynamic allocation for 2D array.

Objective: -

- To understand working and flow of multithread programs.

Requirements: -

1. Create three local matrices, M1 **MxN** M2 **NxP** and Result **MxP** (M1 columns = M2 rows) where M, N & P values are provided by user.
2. In case M1 columns != M2 rows print error message to user.
3. Create all matrices using dynamic allocation.
4. Use structure to pass arguments to threads
   sample structure.

```
typedef struct thread_data
{
        short m1_row;
        short m1_col;
        short m2_col;
        int **matrix1;
        int **matrix2;
        int **result;
        short cur_row;
}Thread_data_t;
```

5. Each thread will calculate and store single row in result. So number of threads equals number of rows in M1.
   Eg: M1= 1 2 3 M2 = 1 1 1
   1 1 1 2 2 2
   2 2 2 3 3 3

   Thread 1 → M1 row1 x M2 col1, col2, col3

   1x1 + 2x2 + 3x3 1x1 + 2x2 + 3x3 1x1 + 2x2 + 3x3

   14 **14** **14**

Thread 2 → M1 row2 x M2 col1, col2, col3

1x1 + 1x2 + 1x3	1x1 + 1x2 + 1x3	1x1 + 1x2 + 1x3
6	**6**	**6**

Thread 3 → M1 row3 x M2 col1, col2, col3

2x1 + 2x2 + 2x3	2x1 + 2x2 + 2x3	2x1 + 2x2 + 2x3
12	**12**	**12**

6. Don't create any global variables.
7. Create generic function for matrix dynamic allocation and deallocating.

Sample execution: -

1. **./matrix_mul**

```
Enter M1 rows and columns
3  3
Enter M2 rows and columns
3  3
Enter M1 values
1  2  3  1  1  1  2  2  2
Enter M2 values
1  2  3  1  2  3  1  2  3
Result is
14   14   14
6    6    6
12   12   12
```