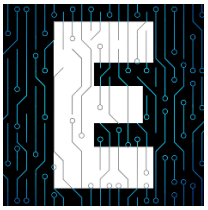
Alexandria University
Faculty of Engineering
Computer & Systems Engineering
CSEx61: Operating Systems



LAB #2

Student Name	Mahmoud Tarek Mahmoud Embaby
Student ID	20011800
Lab Title	Matrix Multiplication (Multi-Threading)
GitHub Link	https://github.com/SajedHassan/Operating-Systems/blob/master/Labs/lab2

1. Code Organization:

The code implements matrix multiplication using **pthreads** library. It defines a struct **thread_data** containing the matrices and their dimensions, the method to be used in matrix multiplication, and the resulting matrix. The method used can be either thread per matrix, thread per row or thread per element.

The **matrix_multiply()** function performs the multiplication of the matrices using the selected method, and the resulting matrix is stored in the **thread_data** struct. The function also measures the elapsed time of the multiplication.

The **write_matrix_to_file()** function writes a given matrix to a file, with the specified name and format.

The **main()** function takes the input and output file names as command-line arguments, reads the matrices from the input files, creates the **thread_data** struct, calls the **matrix_multiply()** function, and writes the resulting matrix to a file.

2. Main Functions:

Method Name	Function
Main()	The main function is divided into three parts. The first part is variable declaration and initialization. The program accepts up to three command-line arguments, which specify the names of the input files and the output file. If no arguments are provided, the program will use the default file names. The program also opens the input files and checks if they can be read. The second part of the main function is responsible for reading the matrices from the input files and storing them in memory. The program reads the dimensions of the matrices from the first line of each input file and then reads the matrix elements from the remaining lines of each file. The matrices are stored as two-dimensional arrays of double precision floating-point numbers. The third part of the main function creates threads to perform matrix multiplication using one of three methods. In the first method, each thread is responsible for calculating the result of one row of the output matrix. In the second method, each thread is responsible for calculating the result of one element of the output matrix. In the third method, each thread is responsible for calculating the result of one complete output matrix. The method used is determined by a command-line argument, or by a default value if no argument is provided.
Matrix_multiply()	The matrix multiplication is performed by calling the function matrix_multiply, which takes a pointer to a struct thread_data as an argument. The struct contains the input matrices and their dimensions, as well as a pointer to the output matrix. The function matrix_multiply checks the method specified in the thread_data struct and performs matrix multiplication accordingly. The result is stored in the output matrix. The function also measures the time taken to perform the calculation and prints it to the console.
Write_matrix_to_file()	The program then writes the output matrix to a file with the specified name or the default name, using the function write_matrix_to_file. The output matrix is also printed to the console. Finally, the program frees the memory allocated for the matrices and closes the input and output files.

3. Run & Compile:

- Open a terminal or command prompt on your computer.
- Navigate to the directory where the C code file is saved.
- Type the following command to compile the code:
  - gcc -o program_name file_name.c *Replace "program_name" with the name you want to give the executable file and "file_name.c" with the name of your code.*
- Press Enter to run the command. If the compilation is successful, you should see a new executable file with the name you specified in the same directory as your C code file.
- To run the program, type the following command:
  - ./program_name *Replace "program_name" with the name you specified in Step 3. Press Enter to run the command.*
- The program should execute and output the result to the terminal.

4. Sample Runs:

A	B	Output	Elapsed time
row=3 col=3 1 2 4 3 4 5 7 8 9	row=3 col=3 7 8 10 10 11 10 1 2 3	row=3 col=3 31 38 42 66 78 85 138 162 177	