CS3523: OS-II - ASSIGNMENT 1

Aim: The aim of this assignment is to understand how matrix multiplication task varies (in terms of speed of execution and sharing matrix data) in single-threaded, multi-threaded and multi-process programming fashion.

Problem: Write a program matmul.c that calculates C=A*B where A is an integer matrix of size ar*ac and B is an integer matrix of size br*bc. Variables ar, ac, br, and bc will be passed as mandatory arguments viz., `--ar`, `--ac`, `--br`, `--bc` respectively from command line and your program should first check feasibility of multiplication and report error if it's not possible. Run the computation in single process/thread, multi-thread, and multi-process variants respectively in a separate sub-routine for modularity. The program should have a routine named **`init_matrix(...)`** to initialize matrices A and B by making use of pseudo random number generator (**`rand(...)`** function) available in standard C library in <stdlib.h>.

Also implement command line option `--interactive` to run the program in interactive mode. For interactive mode, write two more routines named **`input_matrix(...)`** and **`output_matrix(...)`** to input and display the contents of a given matrix in rectangular format. These two routines need to be called only in interactive mode.

Syntax:

```
matmul --ar <rows_in_A>  --ac <cols_in_A>  --br <rows_in_B>  --bc <cols_in_B>
[--interactive]
```

Eg 1 - Non interactive mode.

```
./matmul --ar 3 --ac 4 --br 4 --bc 5
```

Output specification:

```
Time taken for single threaded: 400 us
Time taken for multi process: 300 us
Time taken for multi threaded: 200 us
Speedup for multi process : 1.33 x
Speedup for multi threaded : 2 x
```

Eg 2 - Interactive mode.

```
./matmul --ar 3 --ac 3 --br 3 --bc 3 --interactive
```

Input/Output specification:

```
Enter A:

2 3 4

4 3 4

5 3 7

Enter B:

1 0 0

0 1 0

0 0 1

Result:

2 3 4

4 3 4

5 3 7

Time taken for single threaded: 400 us
Time taken for multi process: 300 us
Time taken for multi threaded: 200 us
Speedup for multi process : 1.33 x
Speedup for multi threaded : 2 x
```

Notes:

1. The timing and speedup shown above examples are fictitious.
2. Do not print any output other than given in the input/output specification as autograder can only check strict outputs as given above. Any debug prints can confuse autograder and award 0 marks for test cases.
3. Do not write any computation logic in main(). Just make calls into single_thread_mm(), multi_thread_mm(), multi_process_mm() functions. These functions must return time taken for matrix multiplication. *Do not account time taken for memory allocation, freeing, creating process/threads.*
4. single-thread and multi-thread functions should use HEAP segment for allocating memory for matrices A, B, C at run time (dynamically)
5. Multi-process function should use SHARED MEMORY segment for keeping matrices A, B, C at run time (dynamically)
6. single_thread_mm() should initialize matrices using init_matrix() and compute C in the same main/default thread.

7. multi_thread_mm() should initialize matrices using init_matrix() and create optimal number of worker threads to compute the elements of C in the fastest manner. Main thread needs to get the work done by the worker threads.
8. multi_process_mm() should initialize matrices using init_matrix() and create optimal number of worker child processes using fork() to compute the elements of C in the fastest manner. Parent process needs to get the work done by the worker child processes.
9. Measure cpu time elapsed for matrix multiplication and return the time in microsecs to main.
7. You need to Strictly follow C coding standard of GNOME project (available at this link) and DON lab C standard written by Prof. Gonsalves for completing this assignment.
8. Create a README which should contain instructions on how to compile, run the program and sample outputs from your system. Do not submit object files, assembler files, or executables.
9. Create a report with your observations on speedup using multiprocessing and multithreading for various sizes of inputs. You need to run the program multiple times to study speedup.
10. *Optionally, tweak thread scheduling parameters and study its effect on the order of thread execution. You can also try assigning different threads to different cores (CPU affinity). Report your observations due to these tuning in the report.*
11. Marks of this assignment are divided for CODE files, documentation, report+README and the speedup shown.

Deliverables:

1. A report describing how you completed above task(s). Measurements of cpu time elapsed and your observations. Make sure that your report is technically sound and readable.
2. README file which helps to know list of files submitted and how to compile and run your program
3. Upload the program, README, report to autograder.

PLAGIARISM STATEMENT <Include it in your report>

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment in any other course, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In

addition, I understand my responsibility to report honour violations by other students if I become aware of it.

Name:

Date:

Signature: <keep your initials here>