Fundamentos de Sistemas de Operação

MIEI 2019/2020

Homework Assignment 1

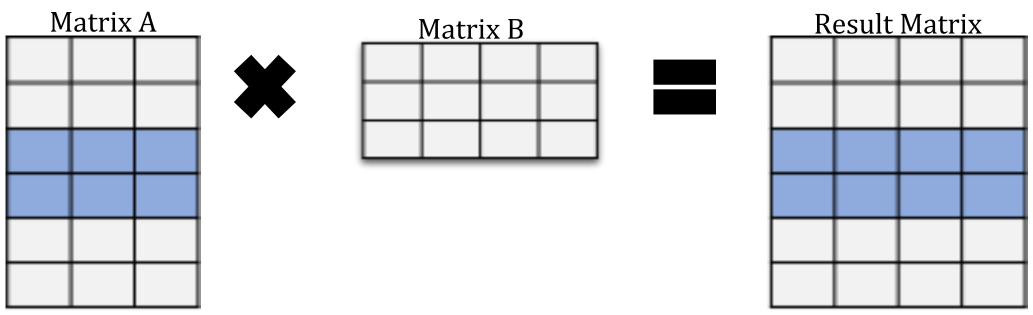
Deadline and Delivery

This assignment is to be performed ***individually*** by each student – any detected frauds will cause failing the discipline. The code has to be submitted for evaluation via the Mooshak system (`http://mooshak.di.fct.unl.pt/~mooshak/`) using each student's individual account -- the deadline is ***23h59, October 9th, 2019***.

Description

The goal of the assignment is to implement the multiplication of two matrices. However, given that matrix multiplication may be computationally heavy, we will use multiple processes to carry out the operation. Accordingly, the initial process (let's call it *master*) will create a given number of child processes (let's call them *workers*) and assign, to each of these workers, the task of computing part of the matrix multiplication.

The algorithm you will use to compute this parallel matrix multiplication is quite simple. Assume that we are computing A x B, each worker will receive a subset of A's rows and the entire matrix B. With this data it will able to compute part of the result matrix, as illustrated in the next figure.

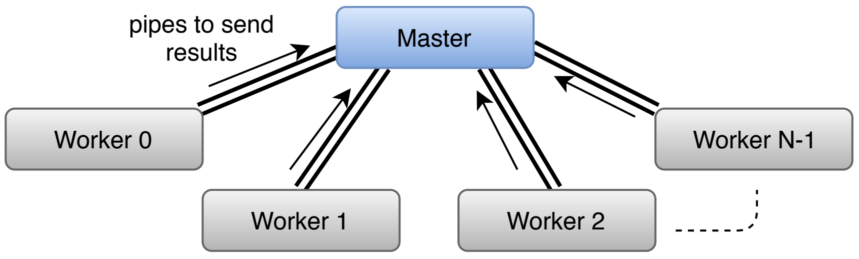


Given that this is an assignment of an Operating Systems' course, you will have to take advantage of your knowledge on process creation to share the data between the master and its workers, namely on the fact that

the *fork* system call creates a clone of the current process, including its memory map.

So, there is no need to send the input matrices to the workers, as they are already in their memory map. You will only have to inform them which are the rows that have been assigned to them.

The same cannot be said about the results. These have to be explicitly sent by the workers to the master. For that purpose, the master must create a pipe for each worker spawned, and each worker is required to write its result to the pipe before terminating.



The behaviour of the master may be, hence, given by the following pseudo-code:

```
Read or generate the input matrices
for w = 0 .. number_workers-1
   create a pipe to receive the result from w
   create worker w
   assign a set of rows to worker w

Allocate space for the final result matrix
for w = 0 .. number_workers-1
   get the result from worker w
   place the result in the final result matrix
```

In turn, the behaviour of a worker is given by:

```
Create a local matrix (PA) comprising only the rows assigned to the worker
Compute a regular matrix multiplication between matrix PA and matrix B
Send the result (a matrix) to the master
```

Work to do

To allow for you to focus your attention on the abovementioned Operating Systems concepts (fork, pipes, etc.), we already supply you with a codebase (downloadable from CLIP) with part of the work done. This codebase comprises 5 files:

- `main.c` is the file that starts the application.
- `matrix.h` and `matrix.c` provide struct `matrix` and a set of functions that you apply over matrices, such as obtaining a submatrix or performing a matrix multiplication.
- `parmatmul.h` and `parmatmul.c` provide struct `worker` and functions to perform the parallel matrix multiplication.

To complete the given implementation, you will only have to work on file `parmatmul.c` and **this is the only file that you must submit to Mooshak**. To complete the assignment, you must:

- Implement function **launch_worker** that creates a process (a worker), creates the pipe for the given worker to return the result and implements the behaviour of the worker.
- Implement function **read_matrix_into** that reads a matrix from a pipe and places it in the final result matrix.

Compile and Run the Application

To compile simply type

```
➢ make
```

To run you must supply the number of rows and columns of the first matrix (matrix A) and the number of columns of the second matrix (matrix B), and the number of workers. The matrices are generated by the application use pseudo-random numbers. You may supply the seed to the pseudo-random number. The command line is thus:

```
➢ parmatmul nrows1 ncols1 ncols2 nworkers [seed]
```

To use 5 workers to multiply a matrix of 10x20 with a matrix of 20x4, you need to type:

```
➢ parmatmul 10 20 4 5
```