Arizona State University
School of Mathematical and Natural Sciences
ACO 350: Systems Programming (Spring 2021)

Prof: Yasin N. Silva

Project # 2: UNIX Multiprocess Programming and Inter-process Communication

Due **Tuesday, March 2nd 2021**, before class starts.

Requirements (submissions that don't follow these instructions won't be graded):

1. You need to work in groups of **four** students.
2. Upload your solution (single file matrix.c) to Canvas. Only one student per team should submit the file.
3. At the top of the file (matrix.c) include the list of participating students and the tasks that each student completed.

Goal: In this project, you will implement a program to multiply two matrices using the C Language and the Virtual Machine environment you created in the previous project. You will apply the concepts of Multiprocess Programming and Inter-process Communication. You are expected to work in groups of **four** students. The guest operating system (the one that was installed in the VM) should be running Linux.

Instructions:

1. Create a program (matrix.c) in C to multiply two matrices in the following fashion:

    a. The main process (parent process) will create multiple child processes. Each child process will solve part of the matrix multiplication problem in the following way:

    Given two matrices A and B:

    A =

1	2	3
4	5	6

    B =

7	8
9	10
11	12

    The result of A x B is:

58	64
139	154

    Where:

    58 = (1 x 7) + (2 x 9) + (3 x 11) //first row of A, first column of B
    64 = (1 x 8) + (2 x 10) + (3 x 12) //first row of A, second column of B
    139 = (4 x 7) + (5 x 9) + (6 x 11) //second row of A, first column of B
    154 = (4 x 8) + (5 x 10) + (6 x 12) //second row of A, second column of B

The work required to perform this multiplication can be divided in 2 parts (1 part per row in the final matrix). Each part will be assigned to a different child process. The first child will compute the result of (1 x 7) + (2 x 9) + (3 x 11) and (1 x 8) + (2 x 10) + (3 x 12), and so on. Note that the final result can be constructed from the partial results. In general, you will create as many child processes as the number of rows in A x B.

b. Each child process will send its partial results to the parent process. The communication should be implemented using pipes.

c. The parent process will construct the final result using the partial results received from all the children.

2. Use the following code as a starting point for your solution (matrix.c).

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h> /*pid_t */
#include <sys/wait.h> /*wait */

#define M 2
#define N 3
#define O 2

int main()
{
	//example matrices
	int matrix1[M][N] = { {1,2,3}, {4,5,6} }; /*M rows x N columns*/
	int matrix2[N][O] = { {1,2}, {3,4}, {5,6} }; /* N x O */
	int Product[M][O]; /* M x O */

	//declaration and creation of pipes
	//...

	//matrix multiplication
	//...

	printf("The matrix product is:\n");
	//Print the result (product)
	//...

	return 0;
}
```

3. Your final solution should support any set of correct values for M, N, and O (number of rows and columns of the matrices). A suggested approach is to solve the problem for the specific values of M, N, and O (number of rows and columns) provided in the initial code. When you have a correct solution for this case, extend your code to support matrices with different values of M, N, and O.

Useful links:

1. Arrays in C: http://www.exforsys.com/tutorials/c-language/c-arrays.html
2. Using pipes and fork in C: http://tldp.org/LDP/lpg/node11.html
3. Language C: http://www.cprogramming.com/tutorial/c-tutorial.html

Project submission and grading

- Upload your solution (single file matrix.c) to Canvas.
- If your code does not compile you will get a grade of 0.
- **Make sure your code compiles and runs when the following lines are changed to specify other matrices to be multiplied.**

```
#define M 2
#define N 3
#define O 2
int matrix1[M][N] = { {1,2,3}, {4,5,6} };
int matrix2[N][O] = { {1,2}, {3,4}, {5,6} };
```