CSC 139, Spring 2015
Department of Computer Science, CSUS
2/2/2015

Programming Assignment 1

- **The due day: February 17.**
- **No credit for a program that does not compile or does not run.**

This assignment requires you to write a multi-processed C program that carries out the matrix multiplication of two simple integer matrices, and displays the results. Make sure that your submitted program compiles and runs on Athena, a Linux machine in ECS.

1. In your program, the parent process takes the input data of *sixteen* integer values from the command line and verifies for correctness (through argc and argv in main(int argc, char *argv[ ])). If the number of input is not correct, or any of the input data is not an integer (e.g., abc, −34a5, or 3.14), the parent process should prompt the user with the message: "Incorrect number of input or incorrect type of input" and then exit. Signed integers such as +25, −38 should be allowed. The sixteen input integer values, $a0$,.., $a7$, $b0$,.., $b7$, given at the command line in that order are elements of the following two input matrices, respectively:
$$\boldsymbol{A} = \begin{bmatrix} a0 & a1 & a2 & a3 \\ a4 & a5 & a6 & a7 \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} b0 & b1 \\ b2 & b3 \\ b4 & b5 \\ b6 & b7 \end{bmatrix}$$
2. After verifying that the input data are correct, the parent process then creates five child processes (**`fork`**) to carry out the matrix multiplication and the result display. The first child calculates $c0$, second child $c1$, third child $c2$, and fourth child $c3$, in the resulting matrix ***C*** below. Four child processes (MC_0, .., MC_3) then send their respective results separately to the fifth child (*Display*) that displays the resulting matrix ***C*** using **`printf`**. Five child processes will exit after finishing their respective tasks (**`exit`**).
$$\begin{bmatrix} a0 & a1 & a2 & a3 \\ a4 & a5 & a6 & a7 \end{bmatrix} \times \begin{bmatrix} b0 & b1 \\ b2 & b3 \\ b4 & b5 \\ b6 & b7 \end{bmatrix} = \begin{bmatrix} c0 & c1 \\ c2 & c3 \end{bmatrix} \qquad \text{where } c0 = a0 * b0 + a1 * b2 + a2 * b4 + a3 * b6$$
3. The *Display* process can display the resulting matrix in the following manner using **`printf`**:
```
The first row of the resulting matrix is:      c0      c1
The second row of the resulting matrix is:     c2      c3
```
4. Four communication channels need to be set up between each of MC_0, .., MC_3 and *Display*. The unnamed pipes will be used for that purpose (use online manual pages **`man`**, or refer to pp.142-145 of the textbook, or pp.124-125 of the reference book for **`pipe`**, **`read`**, **`write`**, **`close`**, if you are not familiar with those system calls). The parent process will set up the pipes before spawning the five child processes.
5. The parent process will wait until after all five child processes finish their tasks (**`waitpid`**), then it exits. Arrange some messages (via **`printf`**) from all processes before they exit. For the parent, display a message before each child process is created, a message before issuing **`waitpid`** calls to start the waiting stage, and a message before it exits. For each of the child processes, display a message including its pid and its parent's pid (**`getpid`**, **`getppid`**) right after being created, a message for its result (computation results, or communication results), and a message before its termination.
6. Be sure to include the following header files at the top of your program. Be sure to include adequate commenting (**point deduction for programs with inadequate comments**).
```
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <sys/uio.h>
#include <string.h>
```

7. Compile your program using gcc and run your program with at least the following four scenarios. Submit your source file to SacCT Assignment1.

```
gcc   prog1.c  -o  prog1
prog1  ……//incorrect number of input, say, 10 or 17 integers.
prog1  ……//incorrect type of input, say, -3a4 in command line.
prog1  ……//16 integers, say, 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
prog1  ……//16 integers, say, 12 −11 10 −9 +8 7 +6 -5 -4 +3 2 1 2 3 4 5
```