CECS 326-01 Assignment 1 (10 points)

Due: 9/21/2023 online on Canvas

This assignment has two parts. The program done in Part 1 will be needed in Part 2.

This assignment is about process creation and coordination using fork(), exec(), and wait() system calls, and commandline arguments on Linux. Syntax of these system calls and their use can be found using the man pages on Linux by typing the following command:

man fork (or any other system call of interest)

**Part 1**

Write a C++ program to be named *child.cpp* and compiled into executable *child*. Run *child* with the command as follows:

./child child\_number name\_of\_child

*child* will behave as follows:

1. receives a child number and the child’s name
2. outputs “I am child number x, and my name is xxxxxx.”

(Note: content of output depends on data received from the command line arguments.)

1. Exits

For example, the command below

./*child* 2 Mary

should produce an output line

I am child number 2, and my name is Mary.

**Part 2**

Write a second C++ program to be named *parent.cpp* and compiled into executable *parent*. Run *parent* as follows:

./*parent* name\_1 name\_2 … name\_k

*parent* will behave as follows:

1. takes in the list of names from the commandline arguments
2. creates as many child processes as there are in the name list and have the child process execute the *child* executable from Part 1, and passes to each child process a child number and a name of that child
3. waits for all child processes to terminate
4. outputs “All child processes terminated. Parent exits.” And terminates.

Each child process invoked by *parent* will behave as described in Part 1.

**Sample run**

To invoke the execution:

./*parent* Nancy Roberto Joseph

*parent* process does the following:

1. outputs “I have 3 children.” -- Note: the number 3 comes from the number of names in the commandline arguments
2. creates 3 child processes, and have each execute *child* and passes to it an integer that represents the child number and the next name from the name list in the commandline arguments
3. waits for all child processes to terminate, then
4. outputs “All child processes terminated. Parent exits.”

Output from *child* processes (It should be noted that the order of child’s output may vary.)

From first child process:

I am child number 1, and my name is Nancy.

From second child process:

I am child number 2, and my name is Roberto.

From third child process:

I am child number 3, and my name is Joseph.

Output from *parent* process

I have 3 children.

All child processes terminated. Parent exits.

**Submit on Convas** the following:

1. The source programs *parent.cpp* and *child.cpp*;
2. A screenshot that shows successful compilation of *child.cpp* to *child* and a successful run of *child* for Part 1;
3. A screenshot that shows successful compilation of *parent.cpp* to *parent* and a successful run of *parent* for Part 2; and
4. A cover page that provides your name, your student ID, course # and section, assignment #, due date, submission date, and a clear program description detailing what the programs are about. Format of the cover page should follow the cover page template posted on Canvas.
5. The programs must be properly formatted and adequately commented to enhance readability and understanding. Detailed documentation on all system calls are especially needed.