Oregon State University
College of Engineering

All Parts of this programming project are October 2nd by midnight. Submit a single gzipped `tar` file to **TEACH**. The single gzipped `tar` file should contain the all source files (C source [`*.c` and `*.h`] and the `Makefile`). Submitting your solutions before October 2nd will earn you a 10% bonus. **You must have a single `Makefile` to build all portions of this assignment.** If you don't know how to create a gzipped `tar` file, you need to learn before you submit this assignment. This assignment *should* be an easy refresher for you. it is not intended to be a long arduous assignment.

1. Create a single `Makefile` that will build all of the below C programs. Place them all in a single directory (no sub-directories). If you don't have a `Makefile`, your submission will not be graded. If your `Makefile` does not build a portion of your assignment, that portion will receive a zero grade. I expect to build all the programs below by using the following two commands:
   ```
   make clean
   make all
   ```
   If this does not work, it will be bad for your grade on this assignment.

2. Write a C program that calls `fork()`. Before calling `fork()`, have the parent process assign the value 100 to `int` called `xx`. What value is the value of `xx` in the child process? Have the parent process assign the value 999 to `xx` and have the child process assign the value 777 to `xx`. What happens to `xx` variable when both the child and parent change the value of `xx`? Have both processes print the value of `xx`.

3. Write a C program that opens a file name "`JUNK.txt`" (with the `fopen()` function call). Have the parent process print "`before fork`" into the file. The parent process should call `fork()` to create a new process. Have the parent process perform a for loop that prints "`parent`" into the open file 10 times. Have the child process perform a for loop that prints "`child`" into the open file 10 times. What happens when they are writing to the file concurrently, i.e., at the same time (answer this in the comments in your code)?

4. Write a C program using `fork()`. The child process should print "`hello`" to stdout; the parent process should print "`goodbye`" to stdout. **Make sure that the child process always prints first**. Can you do this without the parent calling `wait()` in the parent (and NOT using some big loop in the parent)?

5. Write a C program that calls `fork()` and then the child process calls each of the following `exec()` functions to run the program "`ls -l -F -h`": `execl()`, `execlp()`, `execv()`, and `execvp()`. The parent process must wait until the child process is complete before it exits. Describe the differences of the exec functions in your code as comments.

Final note

The programming projects in this course are intended to give you basic skills. In later programming projects, we will ***assume*** that you have mastered the skills introduced in earlier programming projects. **If you don't understand, ask questions.**