**CSCI 4727 Homework 2**

**Due: Thursday, February 11, 5:00 pm**

This assignment is intended to give you some Linux scripting practice and also some experience with creating processes. Note that you may have to re-do your makefile from Homework 1 a bit: your executable \*must\* be named hw1; the input file name is in argv[1], and the output file name is in argv[2]. ***This is an individual assignment.***

**Part 1:** Write a ***bash shell script*** named part1 that does the following: First, check for the existence of a makefile; if it’s *not* there, display an error message and quit, but if it exists, run the makefile. Second, check for the existence of an executable called hw1; if there, run it using the same command line arguments that part1 has (which should be the same kind of command line arguments from Homework 1); if not, display an error message, then quit. See the bash shell example “bashexample” in ~barrettm/4727/examples.

**Part 2:** First, fix any problems that you had in Homework 1, including refactoring the code to use functions – make the program reasonably modular, not just one large main program.

Next, write a short C/C++ program (name the executable part2) that calls fork() to create a child process. The child should print its pid and its parent pid using the function calls getpid()and getppid(). Then the child should use the execv (in the exec family under man; see below) to run your hw1 program.

The main program will call the fork() system call. As discussed in class, this function clones the process; the only difference is the return value of the function: 0 is returned to the child process, and the child’s process id is returned to the parent (see the handout about Unix functions). The usual method is to use this code pattern:

value = fork();

if ( value == 0) {

child\_function( );

} else {

more parent code ...

};

Note that child\_function( ) should ***not*** return – it should call exit(0) when it finishes its job. If the child's job is just a few lines of code, a function is not needed (but the exit(0) call still is), but the function style is recommended. If the parent has more than a few lines of code, then the section "more parent code" should be replaced by a function call, too.

The syntax is execv(filename, argv). The arguments to the command are often optional. For example, the command

execv("cal", argv )

will execute the cal command (it displays a calendar), where the argv argument will pass along any command-line parameters of the original process. Instead of cal, you'll substitute hw1.

After creating the child, the parent should use wait() before exiting. The wait() command takes an integer pointer as a parameter (for example, wait(&i) ). Also, you need to use #include <sys/wait.h> and #include <stdlib.h>. Recall that fork() returns the child's pid to the parent and 0 to the child, or -1 if there was an error on creation. After the child dies, the parent should display the output file created by hw1 using the system() command – that is, don’t write code to display the file, have the system do it. Then the parent should exit.

**Deliverables** Send me your part1 bash script; the source code \*only\* (***no .o files or executables!***) and makefile for part2, zipping them up, and mailing them to me at [barrettm@etsu.edu](mailto:barrettm@etsu.edu). Make sure that you name your executables “part1” and “part2”, that your homework 1 executable gets named “hw1” in the makefile, and that the subject line of your email has your last name and hw2 in it. No hard copy is required.