Chapter 2 lecture notes (pg. 10-19)

Purpose: This assignment covers processes. The programming portion familiarizes you with fork and wait.

You may start with a copy of the program `/net/326/fork.c`.

Modify the fork so that:

1) If the process is the child, it prints "C" in front of the number. If the process is the parent, it prints "P" in front of the number. (Note: you will have to use the if statement from the second fork example.)

2) The loop only goes to 10.

3) If it is the child, just after it prints the number, it sleeps for 2 seconds, if it is the parent , just after it prints the number, it sleeps for 1 second. (See `man 3 sleep` for usage.)

4) After the loop, check the command line arguments (see the lecture notes for how to use argc/argv). If there is an argument (other than the name of the file being executed) and you are the parent you should wait for the child to complete. If there is no argument (or you are the child) execute the return immediately. You will know this works if your shell doesn't return until the child is finished (IE, if you can type commands before it's done, the parent isn't waiting).

Note: The compiler will warn about implicit declaration of wait/wait3. You will need an additional include to stop this warning (and is good coding practice to do so). See the man page for wait/wait3 to know which include has it.

Chapter 3

This next part has no demo or submission, but you need to familiarize yourself with getting process IDs for future assignments. Take notes.

On your Linux box run the `ps aux` command. The `ps` command lists information about processes running on the box. Unfortunately there are often too many of them to fit on so you may have to add a `| less` to get a page at a time. The `aux` form of the command gives all processes (`a` and `x`) and a long format with user names `u`. Notice there are columns that tell you how much cpu and memory the process is using and when the process was started. Alot of system processes are just sitting around waiting for something to do; these are not using cpu or a measurable amount of memory. Notice also the column that gives the PIDs.

Scroll to the bottom of the ps listing and you will see the processes you are running. Your processes are using memory and CPU, although with the speed of our processors, the amount of cpu might not be enough to show up. Notice also you now know the process IDs of your processes. For some of your later assignments you will need to know these.

Demo: Your fork.c program. The instructor would also like to see the code during the demo. And, again, leave the code on the server so the instructor may check it.

Do: While there is no submission for the chapter 3 part, practice getting process IDs for things. For example, your user has a `bash` process running that gives you a shell. What is the pid for it, and how did you find it? Note that. It is important later.