Chapter 2 (last part) and Chapter 3

Purpose: The programming portion of this assignment covers exec.

Build a minishell. Start with a copy of `/net/326/minishell.c`. This contains code that parses a line into a `char* narg[]` array.

The program has an infinite loop. You will exit with a `ctrl+c`.

Inside the loop:

Remove the part that prints the parsed line.

Insert the following.

1) `fork` a child

2 parent) `wait3` for the child to complete After the child completes the parent goes back and prints another prompt. (Note: you will need to include sys/wait.h as in homework 3 to suppress a compiler warning).

2 child) exec what was entered. If you examine what the parser generates you will notice that you want to use the `execvp`. (Note: execvp may also require an include to suppress a warning, see its man page to know which).

Discussion:

You get the prompt "Your command please: "

You type "ls -la"

You see the output of the ls.

When the ls is done, you see the prompt ("Your command please").

Examination of existing code:

Read the manual entry on strtok (string tokenize). Make sure you understand how that part of the code works.

The exploration portion of this assignment has you look at the Linux kernel.

Which modules are loaded by the kernel of the machine you are using?

You must use `man -k` to find out the commands which pertain to modules. Then use the correct command, of course it wouldn't hurt to read the entry for the command before you use it. Hint: the command shows the status of the modules in the Linux Kernel.

To simplify, if there are a lot of sound modules loaded, you can just report how many of those there are rather than listing them all.

The Linux kernel. We mentioned that Linux makes everything look like a file, including information going to and from the kernel. We stated that Linux presents this system information in the directory called `/proc`. Remember `/proc` isn't really a directory, it just looks like one. The files aren't really files either. They are just information coming from (and going to) the kernel.

Now for the questions. Go to the `/proc` directory and see if you can find the file that has information on interrupts. Examine that file and answer the following questions:
What interrupt number is used by the timer?
How many timer interrupts have there been? (The number you see is how many since the last reboot.)
What interrupt number is used by the first usb controller (usb1)?
How many usb1 interrupts have there been?

Demo: Your minishell and the code for your minishell.

Submit: Answers to the Linux Kernel and Module questsions. They may be printed or handwritten.