CS 460 Programming Assignment 2 Your Own Unix Shell

Due Feb 29, 2016 11:59pm Milestone 1
Due March 7, 2016 11:59pm Milestone 2
Due March 14, 2016 4:45pm (DEMO) Milestone 3 65 points

Goal:
Learn about Unix processes, interprocess communication, fork(), exec(), pipe() dup2(), waitpid().

Description:
You will write a Unix shell, a small command line interface to Unix. The shell will need the ability to launch applications (spawn other processes), do input and output redirection, allow an application's output to be piped to another application's input, and allow an application to be launched in the background.

Your Program:
Your program will need to parse the command line input from the user and launch applications as required. You need to handle the redirection symbols which will redirect stdout to a file (>) or read stdin from a file (<). Users also need to be able to use the | to send the output of one application to the input of another. Finally, the & symbol at the end of a line will launch an application in the background and present the user with a shell prompt again before that application terminates. Simple examples follow. Note the prompt displays the process ID of the shell.

Sample Output:

```
zeus$ ./CS460_Shell
21364> ls
CS460_3H.pdf  CS460_3.pdf  hint.txt
21364> cat hint.txt
you should look at man -s 2 pipe
recursion is fun!
21364> cat hint.txt | grep fun
recursion is fun!
21364> cat hint.txt > newFile.txt
21364> ls newFile.txt
newFile.txt
21364> cat < hint.txt | grep man
you should look at man -s 2 pipe
21364> firefox &
21364> ps | grep firefox
 2515 pts/1    00:00:01 firefox
21364> exit
zeus$
```

►Implement the ; symbol to separate commands. The maximum number of characters before pressing return is still 2048.

```
21346> ls ; cat hint.txt
hint.txt newFile.txt
you should look at man -s 2 pipe
recursion is fun!
```

Choose ONE:

Choose one of the following functions and implement it before the final milestone. Notify me via email which function you have chosen to implement.

►Add the alias command to your shell. The alias command allows you to set an alias for a command or set of commands:

```
21346> ALIAS ls = 'ls -al'
21346> ls
-rwx------ 1 chadd staff 8498 Sep  9 10:51 hint.txt
-rwx------ 1 chadd staff  178 Feb 15 21:17 newFile.txt
```

►Allow the user to set an environment variable which is passed to the launched application's environment. I will give you the executable printEnvVariable. Allow the user to show all the environment variables currently set (including inherited ones).

```
21346> SET MyVariable = '999'
21346> printEnvVariable
     MyVariable = 999

21346> SHOW
MyVariable = 999
PATH = ....
....
```

Constraints:

Each command line will be no more than 2048 characters. Each symbol (< > |) will be surrounded on either side by at least one space (**cmd1<in|grep** is not allowed). The & will be preceded by at least one space. You do NOT need to support wild cards (**ls *.txt**).

The only *builtin* functions you need to create are **exit** and **cd**.

Functions (and such) you will (probably) need:
fork(), exec??(), strtok_r(), dup2(), pipe(), waitpid(), STDOUT_FILENO, STDIN_FILENO

Subversion (or how do I submit my work?):

You must store your source code in a Subversion repository on zeus. Name your project **CS460_Shell_PUNetID**.

```
zeus$ ./CS460_Shell -d
21364> ls
command: ls
     arguments: none
     redirection:
          stdin: none
          stdout: none
     pipe: none
background: no
21364> cat hint.txt > newFile.txt
command: cat
     arguments: hint.txt
     redirection:
          stdin: none
          stdout: newFile.txt
     pipe: none
background: no
21364> cat hint.txt newFile.txt | grep fun &
command: cat
     arguments: hint.txt newFile.txt
     redirection:
          stdin: none
          stdout: PIPE
     pipe: YES
command: grep
     arguments: fun
     redirection:
          stdin: PIPE
          stdout: none
     pipe: none
background: YES
21364> exit
zeus$
```

Make Targets

CS460_Shell: build the executable.
valgrind: start the executable with valgrind
debug: start the executable with the -d command line option
valgrind_debug: start the executable with the -d command line option with valgrind

Executable Location

You should build the executable (CS460_Shell) at the root of your Eclipse Project.

Milestones (all milestones are due at 11:59 pm):

(10 pts) **1:** Feb 29: Parsing the command line*****

(10 pts) **2:** Mar 7: Launch an application (with arguments, with backgrounding, no redirection)

(45 pts) **3:** March 14: Final Milestone!

***** For the first milestone, you need to be able to display the parse of the command line as shown. Only display this parse if the user starts your shell with the **-d** command line option. This command line option needs to be present in every version you submit. When this command line option is present (even in later milestones) you should parse and display the command *only* and not launch any applications.

Notes:

You may need to use the man pages (or Google) to determine what header files need to be used to give you access to the functions mentioned above.

Good online set of man pages: **`http://linux.die.net/man/`**

Lookup the parameters that exec() takes to guide you in building a data structure to represent the parsed command line.