Operating Systems: CS5460 Assignment 1

Due Janurary 16th 2014 11:59PM

Step 1:
Read the pdf about setting up git. Once that is complete come back to this to learn *what* we expect you to do in the shell lab.

Step 2:
This assignment will make you more familiar with the Unix system call interface and the shell by implementing several features in a small shell. You can do this assignment on any operating system that supports the Unix API (CADE Lab, Your personal Linux/MacOSX computer). At the end of class commit and push your changes to the grading repository (Read the pdf about the git setup).

The skeleton shell contains two main parts: parsing shell commands and implementing them. The parser recognizes only simple shell commands such as the following:

```
ls < y
cat < y | sort | uniq | wc > y1
cat y1
rm y1
ls | sort | uniq | wc
rm y
```

Cut and paste these commands into a file t.sh You can compile the skeleton shell as follows:

```
$ gcc sh.c
```

which produce an a.out file, which you can run:

```
$ ./a.out < t.sh
```

This execution will panic because you have not implemented several features. In the rest of this assignment you will implement those features.

Executing simple commands
Implement simple commands, such as:

```
$ ls
```

The parser already builds an `execcmd` for you, so the only code you have to write is for the ' ' case in `runcmd` . To test that you can run "ls". You might find it useful to look at the manual page for exec; type "man 3 exec".

I/O redirection
Implement I/O redirection commands so that you can run:

```
echo "CS5460 is cool" > x.txt
cat < x.txt
```

The parser already recognizes ">" and "<", and builds a redircmd for you, so your job is just filling out the missing code in runcmd for those symbols. Make sure your implementation runs correctly with the above test input. You might find the man pages for open and close useful.

Implement pipes
Implement pipes so that you can run command pipelines such as:

```
$ ls | sort | uniq | wc
```

The parser already recognizes "|", and builds a pipecmd for you, so the only code you must write is for the '|' case in runcmd. Test that you can run the above pipeline. You might find the man pages for pipe, fork, close, and dup useful.

Now you should be able the following command correctly:

```
$ a.out < t.sh
```

Don't forget to regularly commit/push your soltution to *Your* forked Assigment 1 repo. When complete with the *entire* assignment commit, and push your solution to the *grading* repo, with or without challenge solutions.

Challenge exercises

You can add any feature of your choice to your shell. But, you may want consider the following as a start:

Implement lists of commands, separated by ";"
Implement sub shells by implementing "(" and ")"
Implement running commands in the background by supporting "&" and "wait"
All of these require making changing to the parser and the runcmd function.