In-class assignment: shell

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 (a Linux Athena machine, your laptop with Linux or MacOS, etc.).

Download the skeleton of the xv6 shell, and look it over. 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 "6.828 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:

This course makes use of Athena, MIT's UNIX-based computing environment. OCW does not provide access to this environment.

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

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.

MIT OpenCourseWare
http://ocw.mit.edu

6.828 Operating System Engineering

Fall 2012

For information about citing these materials or our Terms of Use, visit: http://ocw.mit.edu/terms.