CSCI-P 536

Advanced Operating Systems

Fall 2024

LAB 1

Due: 11.59 PM Sunday, October 6, 2024

DESCRIPTION

In this assignment, you are asked to implement a Command Line Interface (CLI).

PURPOSE

The purpose of this assignment is that you will have a more concrete understanding of how CLIs work, especially parse the given command, multiple parameters, and various operators as well as how to access and use system calls such as `fork`, `wait`, `execvp`, `open`, `close`, `dup`, `dup2`, and `pipe`.

BACKGROUND

A shell is an application that provides a text-based interface to an operating system. By logging into a terminal or console, you can enter commands to navigate directories, copy files, and list contents. These commands are processed by the shell. There are various types of shells, such as Bourne, Csh, Tcsh, and Bash. In this lab, you'll create your own shell.

Your shell will need a prompt string, like this:

```
WHATTHESHELL> cat file.txt
WHATTHESHELL> ls -l
```

In this example, `WHATTHESHELL>` is the prompt, `ls` is the command, and `-l` is an argument. The command typically corresponds to the executable filename.

GRAMMAR

Below is the grammar for valid shell commands. Your shell must parse this to determine if user input is valid. In the grammar, `[sth]` indicates optional elements, and ∗ represents zero or more occurrences.

```
command line: cmd [< fn] [| cmd]∗ [> fn] [&] EOL
cmd: cn [ar]∗
cn: <string> // command name
fn: <string> // file name
```

```
ar: <string> // argument
&: // run in background
```

PROBLEM

Your shell should support the below functionalities, not necessarily in separate lines, and you'll receive the corresponding points.

- Differentiate invalid commands by printing error [40]
- File redirection using < and > operators [20]
  - `cmd < file`
  - `cmd > file`
- Multiple pipes [30]
  - `cmd1 | cmd2 | … | cmdn`
- Background process [10]
  - `cmd &`

Your submission will be tested on several test cases from simple to more complex ones. An example of a complex command is given below.

```
cat < file1 | more | more | grep 3 | sort | head | wc > file2 &
```

NOTES

Submit source code in C with a Makefile and Readme, preferrably in a tar ball. You may want to test your code in the server, below, which you are given access to for this course. In disputes, the results from the execution of the submitted code in the given server will be valid.

```
courserv01.luddy.iupui.edu
```