Lab 2 – A Simple Shell

Part 2: Supporting Execution of Shell Commands

IUPUI CSCI 503 – Fall 2019
Assigned: Thursday, September 19, 2019
Due time: 11:59PM, Thursday, October 3, 2019

1. Objectives

This Lab 2, which follows the previous Lab 1, is designed to achieve the following two goals:

i) To learn how to use a collection of system calls: fork, wait, execvp, open, close, dup, dup2, and pipe.
ii) To understand how an OS's Command Line Interpreter (**CLI**) actually works.

2. Problem

As already described in Lab 1, your own shell should be able to support the following 4 features:

- A command line parser to process your text input, then print the error information if the command line is not valid (already done by your Lab 1).

  New: If the command line is valid, your Lab 2 will execute the command line correctly!

- File redirection: first cmd **<** file, and last cmd **> or >>** file
- Many pipes: cmd1 **|** cmd2 **|** cmd3 **|** … **|** cmdn
- Background: cmdn **&**

Your Lab 2 program may look like the following pseudocode:

```
while (1) {
  printf("your prompt"); //  done in Lab 1
  Read one line from the user; // done in Lab 1
  Parse the line into an array of commands; // done in Lab 1
  Finally, execute the array of parsed commands if the input is a valid command line; //System calls will be used in Lab 2
}
```

3. Score distribution

- Coding style: 10% (as described in the "Labs Grading Policy")
- Can support < and >, >> correctly: 30%
- Can support many pipes (|) correctly: 50%
- Can support background process (&) correctly: 10%

The total score is 100 points.

TA will prepare a number of test cases (from simple to complex) to stress test the correctness of your simple shell. For example, you might want to support the following command line.

```
cat < aaa | more | more | grep 2 | sort | head | wc > bbb &
```

Hint 1: you can input any command line on a Linux shell to see what the expected behavior should be. It may print an error message, or execute then produce an output. Your simple shell should do the same thing as the shell on your Linux machine.

Hint 2: you must understand **pipe4.c** thoroughly in order to complete Lab 2. It is already posted on Canvas.

Hint 3: Test your Lab 2 on Linux first before you submit it. Reserve enough time to test it.

Hint 4: You will need gdb.

Last note: All system calls need error checking.

4. Deliverables

Source codes in C, a Makefile, and a README in a plain text file.

Make a tar ball of the above files and send it to TA via Canvas. The tar ball name should be "Lab2_name.tar".