Lab 1 – A Super Simple Shell

IUPUI CSCI 503 – Spring 2023
Assigned: Feb 2, 2023
Due time: Feb 17 11:59pm, 2023

1. Objectives

This lab is designed to achieve the following goals:

i) To learn how to use a set of system calls: fork, wait, execvp, open, close, dup, dup2, and pipe;
ii) To better understand how a Command Line Interpreter (**CLI**) works.

2. Background

A "shell" is also known as command line interpreter. It is an application that "wraps" around OSes, and provides a text-based human-computer interface. When logging onto a *terminal* or a *console*, you type in various commands to change directories, copy files around, and list files. The commands you entered will be parsed and processed by the "shell" application. There are many types of shells out there (e.g., bourne, csh, tcsh, bash). In this lab you will write your own IUPUI shell.

First, your shell must have a prompt string. You can choose any prompt you like. It may look like the following:

```
IUPUI>  ls –l
```

In the above example, "IUPUI>" is the prompt string, "ls" is a command name, and "-l" is an argument of the command. The command name is usually the filename of the executable.

3. Shell Language Grammar

The following shows the grammar for your users to compose a valid shell command. Lab 1 should be able to parse an input line, and decide if the user's input is valid or not.

In the grammar below, [obj] denotes obj inside [ ] is optional; "|" denotes Unix pipe; "*" denotes 0 or >=1 occurrences..

command line → cmd [< fn] [| cmd]* [> fn] [&] EOL

cmd → cn [arg]*

cn → a string //a command name

fn → a string //a file name

arg → a string //an argument

About "&": The input command line will be running as a background process. Note that a shell must wait until the program completes unless the user runs a background process (i.e., using &).

Note: If you are not familiar with shell, please read the following article:
http://linuxcommand.org/lc3_lts0070.php

4. Lab Requirement:

Your own shell should support the following features:

1) Your program will keep accepting new commands lines until users input "exit". Your shell should have a prompt string.
2) A user's command line may or may not have "<" at the beginning.
3) A user's command line may or may not have ">" in the end.
4) There may be up to 9 pipes in a command line (it's OK if you want to support more than 9 pipes).
5) A user may run a command line as a background process (i.e., the command line ends with "&").

Here are two example command lines that are valid:

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

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

Here are two example command lines that are not valid:

IUPUI> ls | cat < tmp.txt

IUPUI> ls > tmp.txt | more

5. Score distribution

- Coding style: 10% (as described in the "Labs Grading Policy"). Notice: All System Calls must have error checking!
- Your shell should recognize whether a user input is valid or not: 30%
  E.g., If I input "|" or "<" only, you should print "Invalid input".
  Note that there are many possibilities of invalid command lines.
- Support < and >: 20%
- Support from 0 to 9 pipes (|): 30%
- Support background process (&): 10%

The total score is 100 points.

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

Hint 2: Test your program on Linux first before you submit it. Leave at least a week to test it.

6. Deliverables

Source codes in C, a Makefile, and a README.

Make a tar ball of the above files and send it to the TA via Canvas. The tar ball name should be "Lab1_name.tar".

README shall be written in a PLAIN text file (.txt, instead of MS .doc or PDF), and include information such as:

1. How to build your shell (TA will run "make" to build everything).

2. How to launch, then how to quit your shell correctly (normally, users will type in "exit" to quit a shell).

3. Several example inputs, and your expected outputs based on your own tests.

Make a tar ball of the above files and send it to the TA via Canvas. The tar ball name should be "Lab1_Yourname.tar".