**CSCE3613 Operating Systems**
**Programming Assignment One**
**Part one of implementing our simple Shell**

By Wing Ning Li

# 1 Problem Description

The following is quoted from a book by William Shotts.

> What Is "The Shell"?
>
> Simply put, the shell is a program that takes commands from the keyboard and gives them to the operating system to perform. In the old days, it was the only user interface available on a Unix-like system such as Linux. Nowadays, we have graphical user interfaces (GUIs) in addition to command line interfaces (CLIs) such as the shell.
>
> On most Linux systems a program called bash (which stands for Bourne Again SHell, an enhanced version of the original Unix shell program, sh, written by Steve Bourne) acts as the shell program. Besides bash, there are other shell programs that can be installed in a Linux system. These include: ksh, tcsh and zsh.
>
> What's A "Terminal?"
>
> It's a program called a terminal emulator. This is a program that opens a window and lets you interact with the shell. There are a bunch of different terminal emulators you can use. Most Linux distributions supply several, such as: gnome-terminal, konsole, xterm, rxvt, kvt, nxterm, and eterm.

Very often when someone asks us whether we know Unix or Linux what it means is whether we are effective to use a Shell to accomplish IT tasks. To be effective, we must know various commands, understand how the Shell allows interactions between commands. This is the user perspective of learning the Shell.

On the other hand, very likely the Shell is implemented in C/C++. As a program, it must be started. Once it starts, the user enters the input to the program (command line input) to tell it what to do and the program gives feedback to the user. This is done through the input and output statements of a program. As a student studying computing, we know how this part works already. But how does the Shell, after reading the name of our executable file, actually run our program? How does the Shell, which is a program that could be written by us, carry out input/output redirection or pipe? This is the system programmer's perspective of learning the shell and operating system kernel (system calls and code that supports system calls).

Both perspectives are needed so we may develop a simple Shell. We will accomplish it over three programming assignments representing the three stages

of our development. Our program will be written in C, which is a subset of C++ as most lower level system calls are written in C.

To be a bit more specific, our project (to be done in 3 assignments) is to write a C program that runs in Turing (Linux) that mimics the behaviors of the bash Shell program. Our program will use system calls. As a first step, in this assignment, the program will simply parse user input. There are two ways to get user input: interactively from the console (terminal) or as a command line argument. For now, we will process user input as a command line argument. The entire user input is within a double quote (the reason for that is discussed in class and you should know why). In the future, if no input is given, our program reads user input interactively and no double quoting is needed. We want to design our program nicely so that both cases can be handled easily.

A user input has the following:

- A Linux Command that is not internal or a user executable program
- Command line arguments to the Linux command or user executable program
- Redirections >>(output append), > (output overwritten), <(input)
- Pipes connecting two commands or programs |
- A special internal command "quit" to terminate our shell.

Our program will simply output the parsed results. Let a.out be our program. Here are some examples:

```
$ ./a.out   "ls"
The user command or program is: ls

$ ./a.out   "ls -a"
The user command or program is: ls
The command line argument to the user command and program is: -a

$ ./a.out  " ls -a >> myfile"
The user command or program is: ls
The command line argument to the user command and program is: -a
Output redirection: >>
Output file: myfile

$ ./a.out  " ls -l | grep project"
The user command or program is: ls
The command line argument to the user command and program is: -l
Pipe: yes
The user command or program is: grep
The command line argument to the user command and program is: project
```

```
$ ./a.out  " ls -l -a -R| grep project"
The user command or program is: ls
The command line argument to the user command and program is: -l
The command line argument to the user command and program is: -a
The command line argument to the user command and program is: -R
Pipe: yes
The user command or program is: grep
The command line argument to the user command and program is: project

$ ./a.out   "quit"
Program terminates successfully by the user
```

## 2 Purpose

Review C/C++ programming. Review using Linux/Unix. Review programming development under Linux/Unix. Learn or review input parsing.

## 3 Design

The behaviorial of the simple Shell is designed for us by the bash Shell that we want the simple Shell to mimic. As for the user input to our program, there are two ways: 1) statically where the input is provided as part of the command line that runs the simple Shell; 2) interactively.

## 4 Implementation

Using C programming language, array of characters (C style string) and pointers, and C library function.

## 5 Test and evaluation

Try different combinations and scenarios to make sure our program works.

## 6 Report and documentation

A short report about things observed and things learned and understood. The report should also describe the test cases used and the reasons for each test case selected. Properly document and indent the source code. The source code must include the author name and as well as a synopsis of the file.

# 7 Homework submission

To have a better organization, for each of our programming assignments we should have a directory (folder), under which we should have source code (.c .h files), report, and makefile if you use make, etc. This is the directory where we compile and test run our program. In **this directory**, type the following:

```
submit csce3613 wingninggrader assign1
```

Also upload your report (file format pdf or word) in blackboard.