CSCI 3453: Operating System Concepts
Lab Assignment # 2
Due Date: March 8, 2020 @ 11:55 PM

Goal of this programming assignment

The primary goal of this assignment is to understand and gain some familiarity with the system call interface in Unix Environment. In this assignment you will be implementing your own command line interpreter like a Unix shell program. **YOU ARE NOT IMPLEMENTING COMMANDS! YOUR program should just FORK off programs and EXECUTE them.**

Requirements

1) Programming language: You must use either C or C++ to develop your program.
2) Running Environment: Your program should be compiled at CSEGRID and be able to be tested without errors.

The implementation details

Backgrounds: The command line interpreter, called a shell, is an application program that gets commands from a keyboard and uses the system call interfaces to invoke OS functions.

Your shell should support followings:
1) Support exit command which will terminate your shell.
   Example: exit
2) Support commands with no arguments.
   Examples: “ls”, “pwd”, “whoami” and “hostname”
3) Support commands with arguments:
   Examples: “ls –l”, “ls –fx”, “ps –aux”,“date –u”, “cd directory”, etc.
4) A command, with or without arguments, whose output is redirected to a file.
   Examples: ls > foo, ls -Fx > foob
5) A command, with or without arguments, whose input is redirected from a file.
   Examples: sort < testfile
6) Display error messages if the typed command has errors or can’t be found

Hint: Your shell should isolate itself from program failures by creating a child process to execute each command specified by the user.

You might consider the following functions to build your shell:

1) Printing a prompt – when myshell is begun, it display its own string as its own prompt. For example, “myshell> ”.
2) Getting command line – To get a command line, the shell performs a blocking read operation so that the process that executes the shell will be blocked until the user types a command line in response to the prompt. When the user enters the command, the command line string should be returned to the myshell.
3) Parsing the command: Take the user command and divide it into command and arguments and check for errors.
4) Finding the file – The shell provides a set of “environment variables” for each user such as “PATH”. The PATH environment variable is an ordered list of absolute pathnames that specifies where the shell should search for command files. Use getenv function to find the PATH environment variable (http://man7.org/linux/man-pages/man3/getenv.3.html)
5) Create a child process and launch the command – Once the command and arguments are prepared, create a child process to execute the command with its arguments and wait until the command is completed. Use execve function to execute the given command (http://man7.org/linux/man-pages/man2/execve.2.html)

Deliverables

1. Program source codes includes all source program files, Makefile, and Readme
2. Sample Output (Screen shot).
3. Please include your name and due date in comments at the top of each of your C/C++ code file.

How to turn in my work

Please do the followings when you submit your programming assignment.

- Create a tar or zip file that contains your screen shot, source code, makefile and readme. DO NOT INCLUDE EXECUTABLES AND OBJECT FILES.
- Please use the following convention when you create a tar file FirstName_LastName_Lab2.tar or FirstName_LastName_Lab2.zip
- Upload your tar file into Canvas.

Rubric for Grading:

Description	**Points**
Display appropriate prompt	10
Program terminates when exit command is typed	10
Support command with no arguments	20
Support command with arguments	40
Support command with output redirection	40
Support command with input redirection	40
Display error message if command is not found or command has errors	10
Code is well formatted	10
Code is well documented.	10
Total Points	**190**