CS39002: Operating Systems Lab
Spring 2015
Assignment 2(a): Designing Shell
Due on: February 2, 2015, 1 PM

The shell is a program that interprets commands and acts as an intermediary between the user and the inner workings of the operating system - and as such is arguably one of the most important parts of a Unix system.

In this assignment, we shall start making our very own version of a Unix shell (and will complete it in a future assignment).

The shell will give a prompt for the user to type in a command, take the command, execute it, and then give the prompt back for the next command (i.e., actually give the functionality of a shell). Your program for the assignment should do the following:

- Give a prompt (like the $ prompt you get in Linux) for the user to type in a command. The prompt should have the current working directory name (full path) followed by the ">" sign (for ex., /usr/home/me/Desktop>)

- Implement the following commands as builtin commands:
  - **cd <dir>:** changes the directory to "dir".
    - Print an error if the directory doesn't exist.
  - **pwd:** prints the current directory.
  - **mkdir <dir>:** creates a directory called "dir".
    - Assume that the user has write permissions in the current directory.
  - **rmdir <dir>:** removes the directory called "dir".
    - No options need to be supported.
    - Print an message error if the directory is non-empty or if the directory doesn't exist.
  - **ls:** lists the files in the current directory.
    - It should support *ls* both without any option and with the option ***"-l"***. No other options need to be supported.
  - **cp <file1> <file2>:** copies the content of "file1" into "file2".
    - Check if *file1* exists. Print an error message if not.
    - Check if the user has read permission for *file1*. Print an error message if not.
    - Check if the user has write permissions for *file2*. Print an error message if not.
    - If *file2* doesn't exist, copy all bytes from *file1* into a newly created *file2*.

        - If *file2* exists, check the last modification time of *file1* and *file2.* Print an error message if the modification time for *file2* is more recent as compared to *file1.* If not, overwrite *file2*. Note that this requirement isn't a standard bash feature.
        - The file names may contain a full pathname.
        - You can assume "file1" and "file2" are simple files and not directories.
        - No option of cp needs to be supported.
    - **exit:** exits the shell

The commands are the same as the corresponding Linux commands by the same name. Do "man" to see the descriptions. You can use the standard C library functions **chdir, getcwd, mkdir, rmdir, readdir, stat** etc. to implement the calls.

All calls should handle errors properly, with an informative error message. Look up the **perror** call.

These commands are called builtin commands since your shell program will have a function corresponding to each of these commands to execute them; no new process will be created to execute them. (Note that all these commands are not builtin commands in the bash shell, but we will make them so in our shell).

- Any other command typed at the prompt should be executed as if it is the name of an executable file. For example, typing *"a.out"* should execute the file a.out. The file can be in the current directory or in any of the directories specified by the PATH environment variable (use **getenv** to get the value of PATH). The file should be executed after creating a new process and then **exec**'ing the file onto it. Look up **execvp**. The parent process should wait for the file to finish execution and then go on to read the next command from the user. The command typed can have any number of command line arguments.
- Support background execution of commands. Normally when you type a command at the shell prompt, the prompt does not return until the command is finished. For background executions, the prompt returns immediately, the command continues execution in the background. Typing an "&" at the end of a command (for ex., *a.out&*) should make it execute in the background. (Note: Background execution needn't be supported for builtin commands).

Give your program the name *myshell.c*.

Cases of plagiarism will be penalized severely.

Hints and Resources:

- **GNU C Library:** http://www.delorie.com/gnu/docs/glibc/libc_toc.html
- *Different versions of exec:* http://stackoverflow.com/questions/5769734/what-are-the-different-versions-of-exec-used-for-in-c

Contact respective TAs for any further clarifications.