CS 320: Assign 1: Basic Shell

Objective:
Create a basic shell user interface that allows the user to work with process-oriented system calls, execute existing system commands, invoke user-defined executables, manipulate/utilize shell variables and create built-in shell macros.

Note – the following specifications will help you to *incrementally* build a useful tool. Some steps will make it easier for subsequent steps to be accomplished but will ultimately vanish in the final product. It is strongly advised that you complete each step before moving to the next. (It is also strongly advised that you make a commit to Git at each step (at least!).

Specifications:

1. Create a shell that displays a basic prompt and reads input from the user[1]. The shell should spawn a separate process to simply echo the command back to the user and then exit. Once the child process is complete, the parent should reset the prompt collect additional information from the user.
2. Refactor the child block so that it is capable of executing simple system commands. For now, you can/should assume:
    a. Commands are found in the /bin/ directory.
    b. "Simple" commands are ones that do not take any additional parameters. (i.e. ls, pwd, ps)
3. Add a built-in "exit" command to your shell. When the user types "exit" at the command prompt, the shell program is allowed to terminate.
4. Refactor the child block so that it is capable of executing some parameterized system commands. These are commands that take additional command-line arguments i.e.
    - ls –l
    - echo *<sometext>*
    - more *<somefile>*
    - cp *<src> <dest>*
    - mv *<src> <dest>*

    (Hint – understand how the following commands will help you)
    ```
    token = strtok(str, s);
    token = strtok(NULL, s);
    ```
5. Refactor the child block to allow for multiple locations in the executable path (see 2.a above). You should separate entries in the path with a semicolon ; For example, you might specify "/bin/;/home/ubuntu/myShell/bin
   For now, this path should be hardcoded into your shell program.
6. Add built-in shell commands that let you work with shell variables. Specifically:
    a. setPATH updates the path variable
    b. getPATH displays the path variable
    c. Command History:
        i. history: displays the previous 20 commands
        ii. !<num>: executes the command in slot *num* in the history list. (i.e. !2 executes the 2nd most recent command)
        iii. !!: Re-executes the last command entered.

[1] It is understood that you will develop this program using C for use in your Linux environment

7. Create a customized system-command “lls” – this behaves like “ls –l”.
    a. This should be a program that is external to your shell; you will create and compile a separate c program.
    b. The executable should be stored in a directory other than /bin/ (don’t mess with that directory).
    c. This program can (and probably should) use the execv system call.

    Verify that it can run by adjusting the path to include the location of this file.
8. Create a customized system command cpy that behaves like the built in cp command.
    a. Should allow command line arguments to specify the *src* and *dest*
    b. You are only responsible for the base command; implementation of additional flags associated with the cp command is not required.
    c. You may NOT use the exec system call or any variants in your command.