CSC 501 Program: *Shell*

Assignment:

Implement the micro-shell, *ush* (ie, µ-shell).

Description:
Implement the Unix shell described in the accompanying man page. **Do not** to implement job control. Therefore, do **not** implement the following:

- the `fg` command,
- the `bg` command,
- the `kill` command,
- the `jobs` command, or
- the `&` operator.

Resources:

We provide code that performs command-line parsing for the shell. The code is reasonably well-commented and tested; however, it comes with no explicit or implied warranty. Unpack this code with the following command:
`tar xzf ush.tar.gz`

Notes:

1. A parsing routine is provided (code). It comes with no warranty. However, the syntax accepted by the given parser is the *de facto* syntax for the assignment. Therefore, any program that uses the provided parser *unmodified*, will (by definition) not fail any syntax tests.
2. Become familiar with the Unix system calls in Section 2 of the Unix programmer's manual. There are several library routines (Section 3) that offer convenient interfaces to some of the more cryptic system calls. However, **do not** use the *system* library routine. (I.e, do not use the library routine called “system,” see system(3) manual page.)
3. The shell will be forking other processes, which may in turn fork off other processes, etc. Forking too many process will cause Unix to run out of process descriptors, making everyone on the machine very

unhappy. Therefore, be very careful when testing fork. To kill a horde of runaway ush processes, use the `killall` console command, which will kill every process of a given name.

4. Summary of details enumerated in the manual page (in order of appearance):
    - rc file (~/.ushrc)
    - hostname%
    - special chars:
      `& | ; < > >> |& >& >>&`
    - backslash
    - strings
    - commands:
      cd echo logout nice pwd setenv unsetenv where
5. Major tasks
    - Command line parsing
        - ush.tar.gz
    - Command execution
        - fork(2)
        - exec family: execvp(2) might be the best
    - I/O redirection
        - open(2), close(2) (not fopen, fclose)
        - dup(2), dup2(3c)
        - stdin, stdout, stderr
    - Environment variable
        - putenv(3c), getenv(3c)
    - Signal handling
        - signal(5)
        - signal(3c)
        - getpid(2), getpgrp(2), getppid(2), getpgid(2)
        - setpgid(2), setpgrp(2)