Assignment 3

CS392 Spring 2019

Task Details

Implement a shell program, which can run various shell commands (with options) from users.

Requirement

1. Your submission should produce a program “cs392_shell”:
    a. When the program initially starts or it finishes one command, prints “cs392_shell $: ” and then waits for another command
    b. It interactively takes shell commands with options (e.g., “ls -l -a”) from user. By “interactively”, I mean it can continuously take and run commands from user.
    c. For a command it receives, it should fork a child process to execute that command, wait for finish of the child process, and then go back to the state that waits for a new command.
    d. It should support multiple commands concatenated through IO/redirected PIPE
    e. If it receives an “exit” command, it should terminate itself.
    f. For every command it receives, it should append the command (with the options) into a log file called “cs392_shell.log”.
    g. It should register handlers for signals “SIGINT” (CTRL + C) and “SIGTSTP” (CTRL + Z). The handlers should print notification messages and then return to normal execution.

2. Your project should include the following files:
    a. “cs392_shell.c”
    b. “cs392_exec.c” and “cs392_exec.h”
    c. “cs392_log.c” and “cs392_log.h”
    d. “cs392_signal.c” and “cs392_signal.h”
    e. “Makefile”

3. The file “cs392_shell.c” should include the main function.
    a. It will use functions from “cs392_exec.c” to run each command from user
    b. It will use functions from “cs392_log.c” to do the logging
    c. It will use functions from “cs392_signal.c” to register signal handler
    d. It will include “cs392_exec.h”, “cs392_log.h”, and “cs392_signal.h” for prototypes of the above mentioned functions.

4. The “cs392_exec.c” should define a function which forks a child process to run commands from the user. The “cs392_exec.h” should provide a prototype for that function.
    a. Hint 1: combine a “fork” and an exec-family function
        i. Use the parent process to continue working as “cs392_shell”
        ii. Use exec-family functions in the child process to run the received command
    b. Hint 2: the execlp(), execvp(), and execvpe() functions automatically search shell command programs in system directories (for more details: https://linux.die.net/man/3/execlp). Note all the testcases will be shell commands. They can all be found in system folders.
    c. Hint 3: use “wait” in the parent to wait for termination of the child process.

5. The “cs392_log.c” should define a function which writes each command (and the options) to “cs392_shell.log” in the manner of **appending**. If “cs392_shell.log” does not exist, this function should create that file in the current working directory. The “cs392_log.h” should provide a prototype for that function.
    a. Hint: “cs392_shell.log” is just a normal file. You will be able to accomplish the above requirements using the data stream operations that we have learned.

6. The “cs392_signal.c” should define handlers for the SIGINT and SIGTSTP signals and define a function to register those handlers. The “cs392_signal.h” should provide a prototype for that registration function.
    a. About the interfaces to register the handlers, you should use the sigaction interface and use the sa_sigaction field in the second argument to point to the handlers.
    b. You are free to design the handlers, such as what kinds of message to print.

7. The “Makefile” builds the above source code files into “cs392_shell”
    a. You are expected to include explicit rules to compile the “.c” file into object files (“.o” files)
    b. You are expected to include explicit rule to link the objects files to generate “cs392_shell”
    c. You need to include a “clean” target in your Makefile, which, if executed, can remove the object files and the ”cs392_shell”.

8. Zip all the files into a cs392_ass3.zip and submit the .zip file only

Testcase:

-- Start your program cs392_shell with “./cs392_shell”
-- Run the following commands:

- pwd

- ls -l -a
- mkdir temp
- ps -h
- man ls

-- Type CTRL + C and CTRL + Z multiple times, and see if your signal handlers work properly (if they print out your messages and if they can return back to the normal execution)
-- Run the command “exit” and see if your “cs392_shell” exits
-- After you exit from cs392_shell, check if the “cs392_shell.log” file contains your command history

Grading Policy: TBD