TSH: A Tiny Shell

Program Four
CS 3411 Spring 2017
Due: Thursday, Feb. 23, 11:05am

Motivation

Pipes are a useful mechanism for interprocess communication (IPC). Data is delivered in FIFO order and the communication mechanism provides for synchronization around the delivery of data between processes (e.g., a read blocks until data is available). Shells take full advantage of this mechanism by using pipes to allow a user to specify that the output of one command be provided as input to a following command. The shell uses the pipe IPC mechanism to implement the required communication.

In this project, you will get experience with the use of pipes for IPC. You will write a tiny shell (TSH).

Requirements

The basic ground rules are:

- The shell must handle (any number of) pipes. For example, it must handle the command: `ls -lt | grep root | head -6 | tail -3 | head -1`.
- You must use the `execvp` variant of exec to execute commands.
- The prompt for `tsh` should (initially) be set to `tsh#`. (Note that there is no '.' in the prompt.)
- Your shell must execute the user commands directly; you may not exec any variant of `sh` nor can you use the `system()` or `popen()` system calls.
- The command `q` should cause your shell to terminate. The shell must support one additional built-in command: `prompt <string>`. The `prompt <string>` command makes `<string>` the new prompt. The string is a contiguous sequence of characters between ASCII value 33 and 126.
- Your shell must support output (stdout) redirection. For simplicity, the shell does not support both pipes and output redirection in the same command. Addtionally, the output redirection operator must appear after the command. The following commands are valid:

  ```
  ls | grep tip
  grep sshd /etc/passwd > output
  ```

  The following commands are invalid.

  ```
  cat /etc/passwd | grep tip > grepout
  > outfile wc myfile
  ```

There are standard routines that will reduce the work required to parse the command line. Browse the man pages.

Notes

Please note the following.

- The directory `/classes/cs3411/projects/4.tsh` contains skeleton code that parses an input command line. Note that this code does not support input or output redirection and ignores certain coding guidelines required of project submissions. It is your responsibility to ensure that the code you submit meets the project requirements.
- The shell need not support background processing.
- The shell need not support multiple commands on the same command line (e.g. `ls;ps` or `ls | grep ps; last | tail -3`).
- The shell must gracefully handle failure of a command within a command pipeline. Specifically, this should not cause the shell to terminate.
- The shell does not support redirection of stderr.

Submission

Submit all code (using the standard course submission procedure) in a file named `tsh.tgz`. Include a makefile so that when the grader types `gtar zxf tsh.tgz; make` a binary file named `tsh` is created. Typing `make clean` should remove all object files and the created binary `tsh`. Also submit hard copy of your code.