Keawa Rozet

916796184

CSC 415

Project 3 – Simple Bash Shell

Command to build and run the project:

My program involves the use of readline, which requires extra compilation. So to avoid all that, I recommend using the makefile I altered to build the program.

make -f Makefile

./myshell

What my code does:

This simple shell is capable of handling redirection and pipes, along with other standard linux commands. I have not implemented the capability of multiple pipes and redirections, but I plan to do that in the future turning in this project.

The shell runs in a loop until the user presses CTRL+C or types the command ‘exit’. The user is prompted for a string, and this string gets check whether or not it is empty. If it is empty, the program skips everything following and returns to the initial prompt. Once a valid string has been given from the user, the string immediately gets parsed.

getArguments uses strtok\_r to tokenize the string using spaces. The arguments are stored in the pointer array myargv and the number of arguments are stored in myargc. Two important things to note are the addition of flagArray and pipeArgCounter. flagArray is responisble for keeping track of the type of redirection (<, >, >>) that appears, and stores it in chronological order by index. pipeArgCounter works differently, however, by storing the number of arguments between each pipe. If no pipe is detected though, then the first element of pipeArgCounter never gets initialized and that is used to signal to the main loop that there are no pipes found in the string.

Continuing in the main loop, the first check is if there are any pipes, then it checks for flags, then it checks if myargv contains any built in commands (cd, pwd, help, exit), and finally it assumes that myargv contains a system command. Once processed, reset() is called and all the buffers and counters are reset to their initial states.

handleBuiltinCommands is a simple method that takes the first index of myargv and compares it to the builtinCommands array. If there is a match, then the switch statement handles the specific built in command. If no built in commands were found, then 0 is returned which signals that the command must by a system command.

handleSystemCommands takes myargv, myargc, and the flags. First a fork is created to handle the system command. Next, it checks if ‘&’ is present at the end of the myargv. If so, then the runBackground signal is set to true and ‘&’ is set to the value of NULL. Since it is the child process that handles the execution of the command, the flag is checked inside the block ‘pid == 0’. If a flag is found, then the redirect is set to NULL and the switch statement handles the appropriate file discriptors for the redirect. After which, the command is executed. If the runBackground signal was not set, then the parent process will wait for the child process to finish executing. If the runBackgroud signal was set, then the method will return without waiting for the child process.

HandlePipedCommands opporates closely to handleSystemCommands but uses two forks to handle the piping. First, the runBackground signal handled to see if the program needs to run in the background. pipe() is called on pipefd and the first fork() is called. It runs dup2 on STDOUT so that it can communicate via piping. The first command is executed and the second fork() gets executed as well. The child process of this second fork() closes the STDOUT fd and runs dup2 on STDIN. The second command is executed, and finally both pipefd[0] and pipefd[1] are closed. If the parent process was signaled to wait, then it will wait for both child processes.

This code largely revolves around the initial parse into the myargv pointer array. Certain counters are created to easily traverse myargv so to achieve the specific outcome specified. This was more ideal instead of iterating through the user’s initial input multiple times depend on if a pipe or redirect was found on the initial scan.

Example Execution

,adPPYba, ,adPPYb,d8 ,adPPYb,d8

a8P\_\_\_\_\_88 a8" `Y88 a8" `Y88

8PP""""""" 8b 88 8b 88

"8b, ,aa "8a, ,d88 "8a, ,d88

`"Ybbd8"' `"YbbdP"Y8 `"YbbdP"Y8 .-~-.

aa, ,88 aa, ,88 .' '.

"Y8bbdP" "Y8bbdP" / \

.-~-. : ;

.' '.| |

/ \ :

: ; .-~""~-,/

| /` `'.

: | \

\ | /

`. .' \ .'

`~~~` '-.\_\_\_\_.-'

@ myEggShell ~/krozet-git/csc415-p3-krozet -> ls -al

total 256

drwxr-xr-x 3 kvothe kvothe 4096 Mar 16 11:54 .

drwxr-xr-x 16 kvothe root 4096 Mar 5 14:31 ..

-rw-r--r-- 1 kvothe kvothe 175737 Feb 26 08:30 415\_HW3.pdf

-rw-r--r-- 1 kvothe kvothe 157 Feb 26 08:30 CMakeLists.txt

-rw-r--r-- 1 kvothe kvothe 8666 Mar 16 11:53 documentation.docx

drwxr-xr-x 8 kvothe kvothe 4096 Mar 16 10:26 .git

-rw-r--r-- 1 kvothe kvothe 430 Feb 26 08:30 .gitignore

-rw-r--r-- 1 kvothe kvothe 1074 Feb 26 08:30 LICENSE

-rw-r--r-- 1 kvothe kvothe 79 Mar 16 11:53 .~lock.documentation.docx#

-rw-r--r-- 1 kvothe kvothe 85 Feb 26 17:43 Makefile

-rwxr-xr-x 1 kvothe kvothe 19696 Mar 16 11:54 myshell

-rw-r--r-- 1 kvothe kvothe 9881 Mar 16 11:44 myshell.c

-rw-r--r-- 1 kvothe kvothe 11 Feb 26 08:30 README.md

-rw-r--r-- 1 kvothe kvothe 4 Feb 26 08:30 readme.txt

@ myEggShell ~/krozet-git/csc415-p3-krozet -> ls -l | grep myshell

-rwxr-xr-x 1 kvothe kvothe 19696 Mar 16 11:54 myshell

-rw-r--r-- 1 kvothe kvothe 9881 Mar 16 11:44 myshell.c

@ myEggShell ~/krozet-git/csc415-p3-krozet -> help

@ Simple Linux EggShell @

Standard linux commands as well as '&' and '|' will work.

cd Changes directory.

cwd Path to current working directory.

exit To quit the shell.

@ myEggShell ~/krozet-git/csc415-p3-krozet -> cd ..

@ myEggShell ~/krozet-git -> cd /

@ myEggShell / -> ls

bin boot dev etc home i3 lib lib64 lost+found mnt opt proc root run sbin srv sys tmp usr var

@ myEggShell / -> cd mnt

@ myEggShell /mnt ->