Operating Systems - CSE231
Assignment 0

Brihi Joshi
2016142

January 23, 2018

1. About Pretty Okayish SHell (POSH)

POSH is an ultra simple shell that acts as an interface between the user and the Operating System kernel. It can process two kinds of commands – "Built-in" and "External". Built-in commands are those which are interpreted by the shell program itself, without requiring a different program to handle the expected operations (of the said command). External commands on the other hand relate to commands which are not handled directly by the shell program but by an external program.
In the code, the Built-in commands are handled by the POSH itself, in conditional statements. For external commands, separate files are made, which are then executed in a child process. POSH typically creates a new process, using the fork() system call and within each process it uses the execvp() system call to run the individual program executables. The parent program must waits for the child program to terminate using the wait() system call.

2. How to run POSH?

After downloading the files, the following directory structure is present

```
-2016142
        -cases.txt
        -report.pdf
        -src/ (this folder contains all C code and the Makefile)
```

You would need to change the path inside /src/posh.c (comments are written where to do that) to where your folder is present.

Run 'make all' in order to build all executables and then type 'posh' to run the shell.

3. Commands supported (their variations and corner cases handled)

3.1 Built-Ins

cd

It is used to change the present directory of the shell.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. .. - Changes the directory to the parent.
4. ~ - Changes the directory to the $HOME directory.

Corner cases/Bugs handled -

1. If no argument is given, it changes the directory to $HOME.
2. Handles invalid flag options.
3. Handles invalid directory names.

echo

It is used to print the argument given as a parameter into stdout. Assumption - An exact replica of the argument is printed into stdout and the -e flag only takes in a single non-space separated text.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -n - Ignores the newline character
4. -e - replaces all escape characters in the text and converts them.

Corner cases/Bugs handled -

1. Handles the case when no argument is given.
2. Handles echo command written in quotes ("echo",'echo').

exit

It is used to exit the main shell process.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.

Corner cases/Bugs handled -

1. The shell terminates even if arguments are provided.
2. Handles echo command written in quotes ("exit",'exit').

history

It displays the history of all the commands run into the shell.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -c - clears the history
4. -w <filename> - Copies all the history into a file.

Corner cases/Bugs handled -

1. Even if the shell is killed and a new shell is opened, the previous history can be seen (with a tally or a counter).
2. Handles invalid flag options.
3. Handles invalid argument names.

pwd

It displays the current working directory of the shell.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -L - Displays logical current working directory.

Corner cases/Bugs handled -

1. Prints error messages in case it is unable to extract the current working directory.
2. Handles invalid flag options.

3.2 Externals

cat
Reads the contents of the file given as an argument.
Flags supported -
1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -n - Numbers the lines read
4. -e - Marks every newline character with a $ sign.

Corner cases/Bugs handled -
1. Doesn't open file if we don't have read access/file doesn't exist
2. Handles invalid flag options.
3. Handles invalid arguments.

date
Prints the IST date in a given format.
Flags supported -
1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -u - Prints the UTC date in the given format
4. -r <seconds> - Given the seconds are the number of seconds from epoch, it calculates the date in the given format.

Corner cases/Bugs handled -
1. Prints error message in case a non numerical argument is given for -r.
2. Handles invalid flag options.

ls
Lists all the files and directories present in the current directory.
Flags supported -
1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -a - Lists all files, including hidden files
4. -i - displays the inode numbers of the files along with their names.

Corner cases/Bugs handled -
1. If the argument given is a directory that doesn't exist, an error message is printed
2. Handles invalid flag options.

mkdir
Creates a new directory with the name as that of the argument.
Flags supported -
1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -v - (verbose) displays which file is created
4. -p - given a path as an argument, recursively creates the files inside one another.

Corner cases/Bugs handled -

1. Error message is displayed if the file exists.
2. Handles invalid flag options.

rm

Removes a given file/directory.

Flags supported -

1. –help - Prints the usage of the command.
2. –version - Prints the version of the command.
3. -i - Asks during deletion whether the user wants to delete the file or not
4. -v - (verbose) displays the name of the file that is deleted.

Corner cases/Bugs handled -

1. Error message displayed if the file doesn't exist
2. Handles invalid flag options.
3. Doesn't delete non-empty directories