Grand Canyon University

Project 1: Improved Unix/Linux Command Line Interpreter

Evan Lee and Ryan Scott

CST-315: Operating Systems Lecture and Lab

Dr. Ricardo Citro

February 20, 2022

**Table of Contents**

**Project 1: Unix/Linux Command Line Interpreter**………………………………………… 3-6

**Project 2: Improved Unix/Linux Command Line Interpreter**……...……………………..7-11

**Project 1: Unix/Linux Command Line Interpreter**

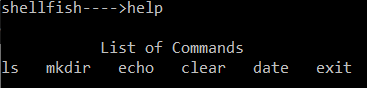
**Introduction:**

For this project we made a simple Linux command line interpreter where the user is given a prompt and is able to type in multiple commands that are used in a Linux terminal. The name of our prompt is “shellfish—->” and will be printed out each time a command has been executed to show that it is ready to accept the next command. For the first implementation of this Linux command line interpreter we have implemented seven commands: ls, mkdir, echo, clear, date, help, and exit. The help command prints out all of the possible commands that Shellfish can run and exit ends the command line interpreter. The following document shows the commands running in a terminal with pictures of the input and output and a flowchart showing the process of how our system operates.

**Commands:**

**help (Prints Out Available Commands):**

Displays a list of Shellfish’s available commands. This command takes in no additional arguments.



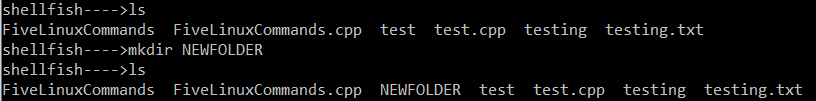
**ls (List Files):**

Used to view the list of files and folders in the current directory. This command takes in no additional arguments.



**mkdir (Make Directory):**

Used to create a new folder (directory) within the current directory. This command requires an additional argument that will serve as the name of the new folder.



**echo (Print Out):**

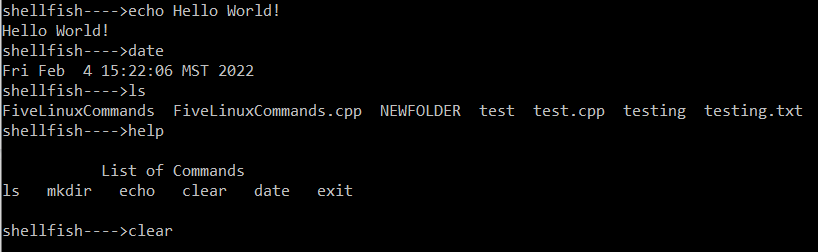
Used to display a string. This command requires an additional argument that will serve as the string to be displayed.

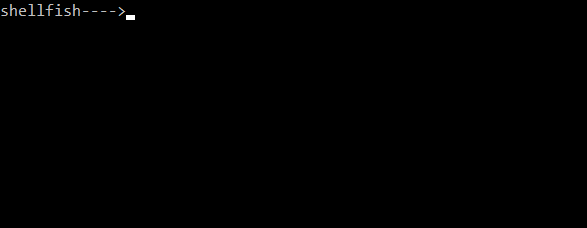


**clear (Clears Console):**

Used to clear the terminal screen. This command takes in no additional arguments.

Before “clear” command:



After “clear” command:

**date (Print Out Current Date):**

Used to display the system date and time. This command takes in no additional arguments.

 ← From Shellfish terminal

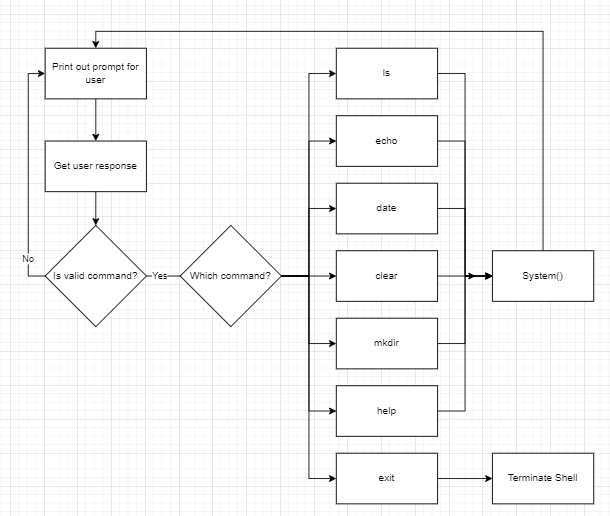
 ← From windows operating system, the two times and dates match.

**exit (Exits the Shell):**

Used to exit Shellfish. This command takes in no additional arguments.

****

**Flowchart and Testing:**



**Design:**

The Linux command line interpreters design is formed around getting a user input and parsing through it to find a keyword for a valid command and an argument if needed. Once a valid command is found, it is run through a series of logic to determine which function will run based on the specific keyword. When the function is run it will create a character array with the command keyword appended into it. Lastly, the character array is sent through a system() call which will execute the command.

**Testing:**

The approach to testing our shell was to first create the C++ script on repl.it, a free collaborative coding space. From there, we were able to compile and run the code on their own Linux shell, testing each command and its subsequent effects. After we were 100% certain the code was working on repl.it, we compiled the C++ script in Ubuntu and ran it, generating the exact same results. This tells us that our solution is correct and fully functional on Linux.

**Troubleshooting:**

The main problem we ran into when creating our shell was the system calls requiring character arrays as a parameter (legacy C functionality). Since we were making our shell in C++, we were actually using strings as the primary way of storing and receiving commands. This required us to use the functions strcpy() and c\_str() to copy the string into a character array. From there, we were finally able to make the system calls we were after.

We also ran into the issue of our first cout call not displaying properly. This was fixed by calling cout.flush() before our getline while loop.

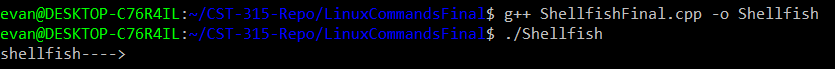
**Project 2: Improved Unix/Linux Command Line Interpreter**

**Introduction:**

For the improved version of the Unix/Linux Command Line Interpreter our main focus was to implement five more commands as well as execute multiple commands in one line. When reformatting our code to add more commands, we noticed that each of the functions in charge of running each command was almost identical, meaning we could convert all ten command functions into one master command function that would accept all commands and run them through system(). We implemented this idea and now the code will execute many more commands without having to manually parse through the string looking for key function command words while significantly reducing the amount of code needed. This implementation does have its drawbacks though, some commands need special permissions and some commands like “cd” don’t work yet. For example, cd does not use system() but instead uses chdir(). For the scope of this project though we will just pick five more commands that will work in our interpreter but there are more than the five extra listed that will work.

**Compiling the ShellfishFinal.cpp file:**

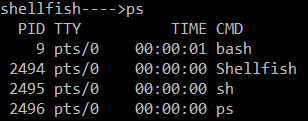
“g++ ShellfishFinal.cpp -o Shellfish” is used to compile the .cpp file, while ./Shellfish is used to execute it.



**New Commands:**

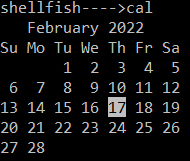
**ps:**

The “ps” command will display the currently active processes. The PID column shows the process ID and can allow you to kill a process if necessary. TTY will show the name of the controlling terminal for the process. TIME will show how long the CPU has worked on the process. Lastly CMD will show what command was used to start the specific process.



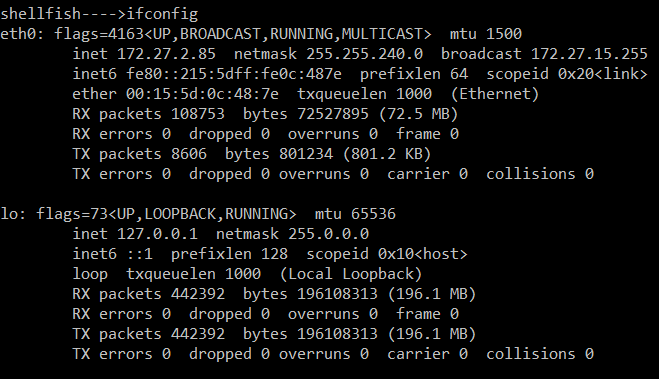
**cal (calendar):**

Will display the current month's calendar with today’s date highlighted.



**ifconfig:**

Will display the IP addresses of all network interfaces.



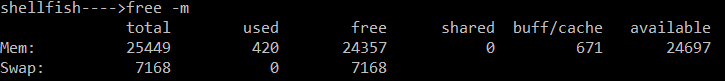
**whoami:**

Displays the user that is currently logged in.



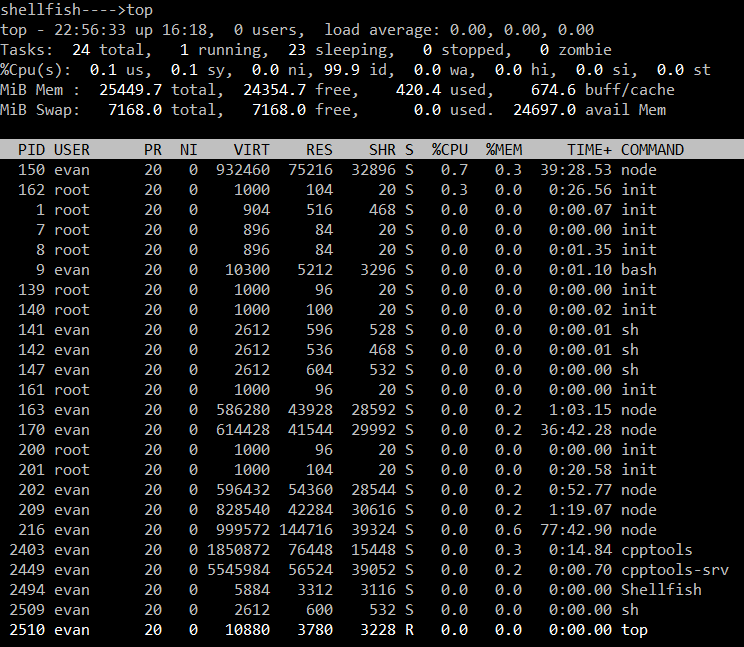
**free -m:**

Displays the memory that is free as well as memory that is allocated already.



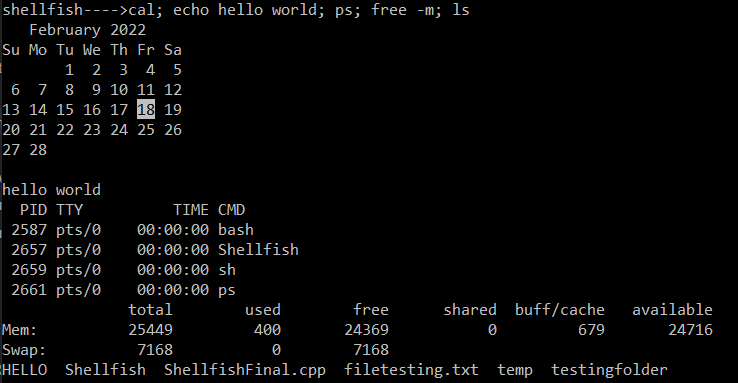
**top:**

Shows the current number of tasks and the resources used. This command is similar to opening task manager in windows. There is a column for process id (PID), percent CPU allocation, percent memory allocation, execution time, as well as many more features.

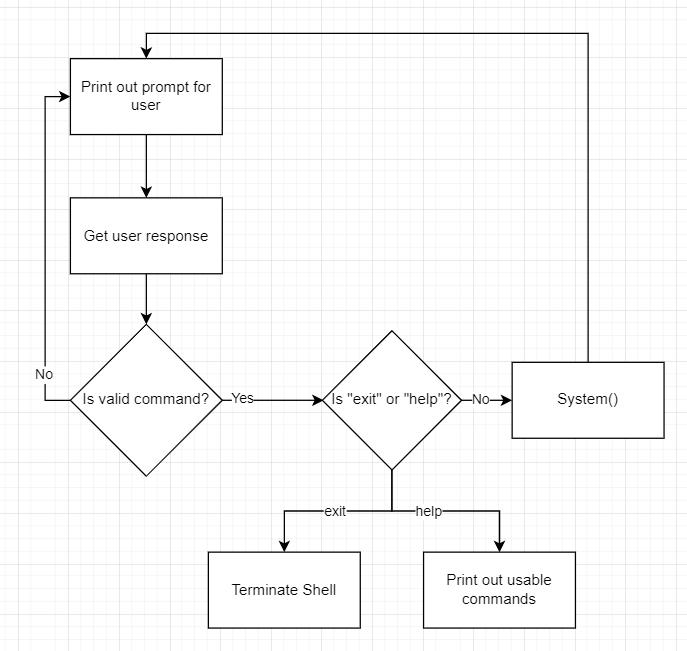


**Running Five Commands at Once:**

By separating commands with a semicolon, the shell is able to perform multiple commands at once.



**Flowchart:**

****

**Design:**

The design of the improved command line interpreter was changed from the first version. Now instead of multiple functions being created for each command, one function is able to replace them to simplify the code much more. Now the only commands that require separate functions are exit and help which we can check for manually. Any other call gets passed into the System() function.

**Testing:**

The approach to testing our shell for the improved Unix/Linux Command Line Interpreter was the same as the first command line interpreter. First create the C++ script on repl.it, a free collaborative coding space. From there, we were able to compile and run the code on their own Linux shell, testing each command and its subsequent effects. After we were 100% certain the code was working on repl.it, we compiled the C++ script in Ubuntu and ran it, generating the exact same results. This tells us that our solution is correct and fully functional on Linux.

**Troubleshooting:**

When working on the improved command Unix/Linux Command Line Interpreter, we initially copied and pasted previous working code, changed the function names and added on to the if-statement chain. That is when we realized that all 10 of our command functions could really be refactored into one simple function where any parameter is used in a system call. Not only did this significantly cut down on the amount of data we had to parse, but it automatically works with multiple commands on the same line too!