Grand Canyon University

Project 4: File System Manager

Evan Lee and Ryan Scott

CST-315: Operating Systems Lecture and Lab

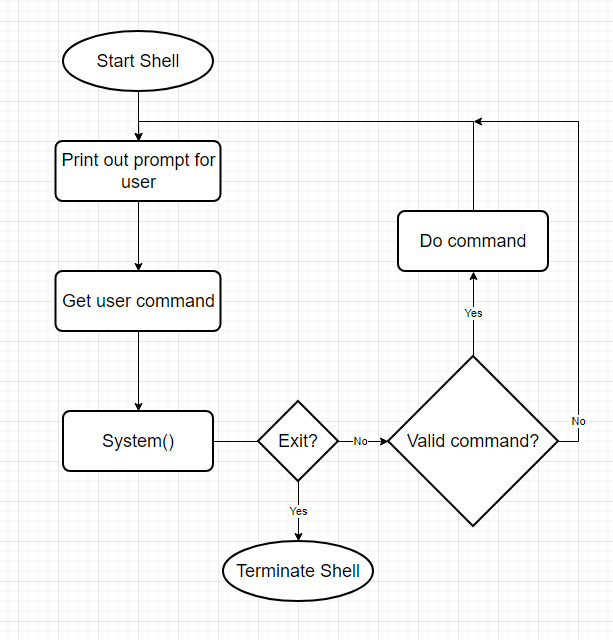
Dr. Ricardo Citro

April 10, 2022

**Introduction:**

For this project, our focus was on implementing a robust file system manager into our Linux command line interpreter. Due to the impressive advancements made in the previous improved command line interpreter project, this required no additional code. It only required additional documentation, expanding upon what could already be achieved in our current Shellfish Linux shell.

**Explanation of Design-Flowchart:**



**Explanation of Design:**

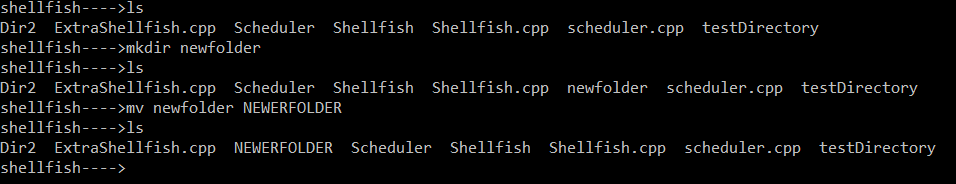
The design for our shell allows us to execute almost any Linux command with any valid flags or additional parameters due to the nature of how System calls work. All we have to do is input the string into a function that will send it through a System call and it will even handle mistyped commands as well. With this working function all we need to do is run a while loop that will ask the user for input until the user either types in the exit command or presses ctrl+c.

**Concept and Analysis:**

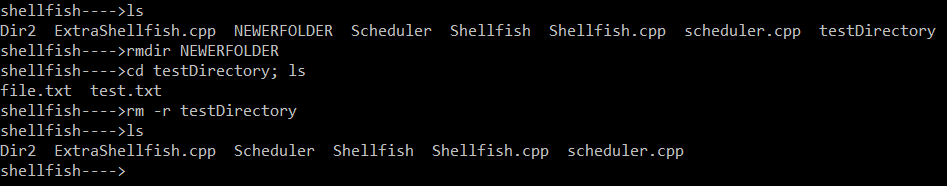
The file system manager is at the core of the Linux command line interpreter because this is what will allow the user to create, read, update, and delete files as well as directories. Without a file system manager there are not any meaningful ways to get user input. By implementing all of these file system commands, it will allow the user to use our shell quickly, reliably, and with much more usability than the previous shell.

**Test and Validate:**

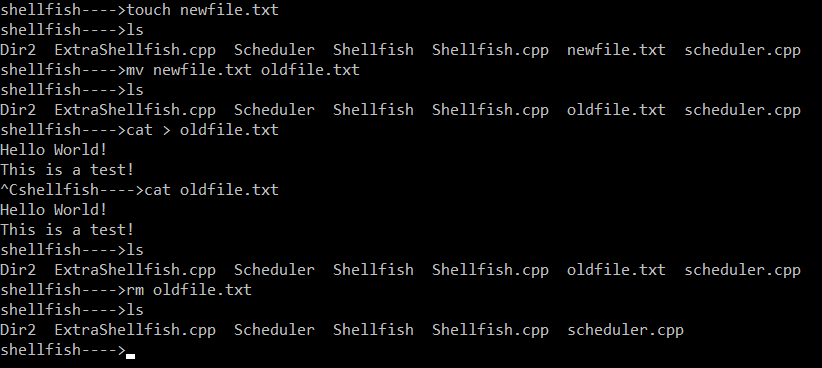
1. **Create and rename directories**
   1. mkdir: used to make a new directory, taking in the name as a parameter.
   2. mv: used to rename an existing directory, taking in the name and a replacement name as parameters.



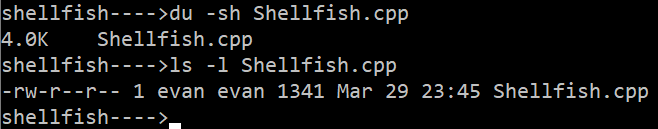
1. **Delete directories, with special handling of non-empty directories (e.g. special flag to delete command)**
   1. rm/rmdir: used to remove an empty directory, taking in the name as a parameter.
   2. rm -r: used to remove a populated directory, taking in the name as a parameter. This uses the special flag -r to do so.

****

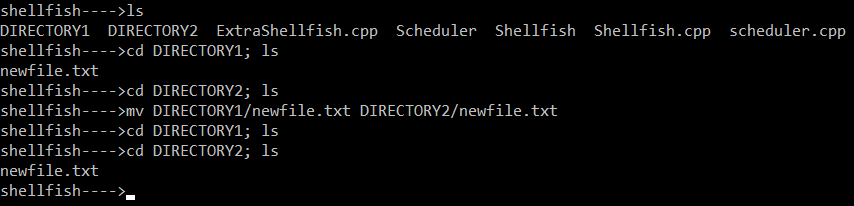
1. **Create, rename, edit, and delete files**
   1. touch: used to create a new file, taking in the name as a parameter.
   2. mv: used to rename an existing file, taking in the name and a replacement name as parameters.
   3. cat >: redirects input from the terminal into a file, editing it. To finish appending, press enter and then ‘ctrl + C’ to return to the terminal.
   4. rm: used to remove an existing file, taking in the name as a parameter.

****

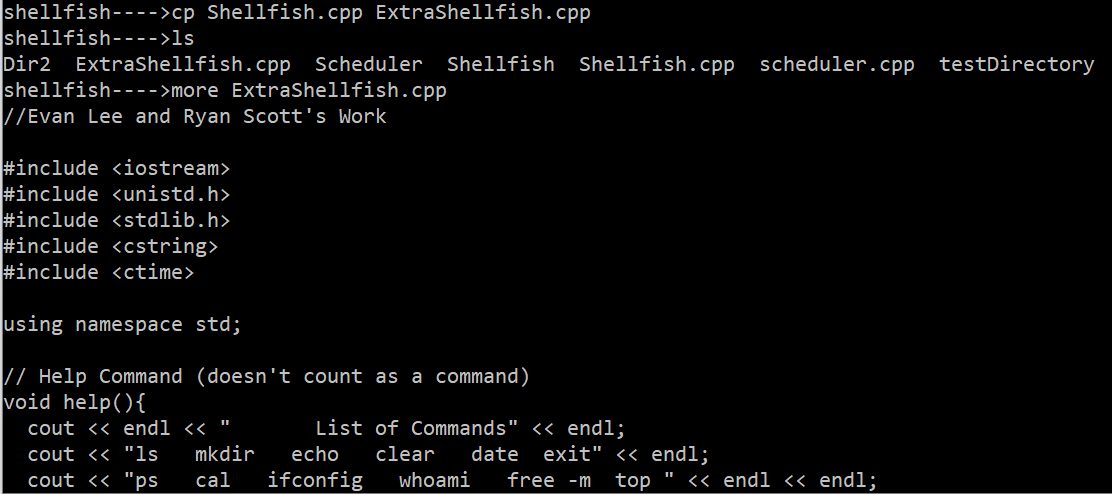
1. **Size of file**
   1. du: estimates file space usage, taking in the name as a parameter.
   2. ls -l: displays basic information of a file, including the size in bytes. Takes the name as a parameter.

****

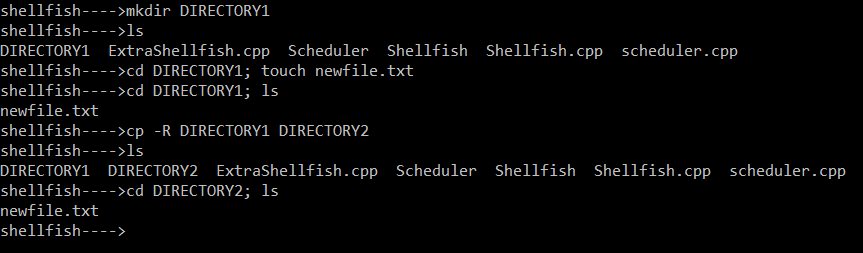
1. **Move files across directories**
   1. mv: moves a file from one place to another, taking in the name of the file and the name of the new directory as parameters.

****

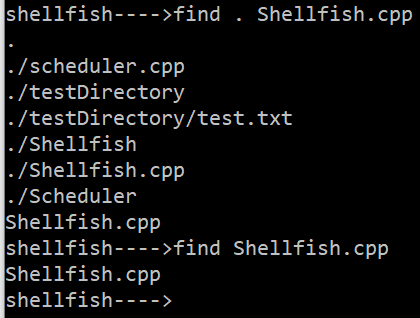
1. **Duplicate files**
   1. cp: copies a file, taking in the name and duplicate’s new name as parameters.

****

1. **Duplicate directories**
   1. cp -R: copies a directory (including all of its contents), taking in the name and duplicate’s new name as parameters.

****

1. **Search for a file in a directory tree**
   1. find: locates files based on the parameters passed in.

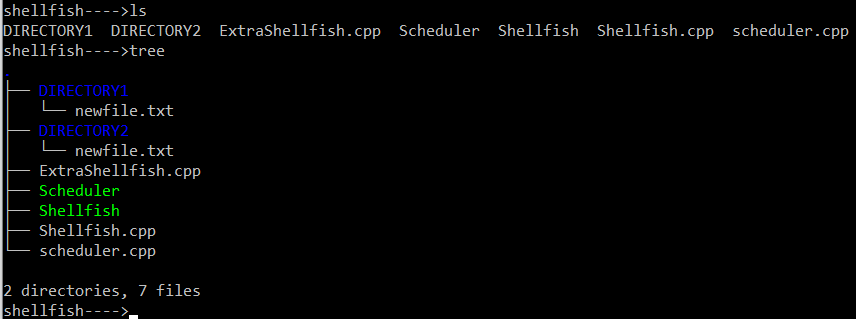
****

1. **Display a directory tree given a starting node**

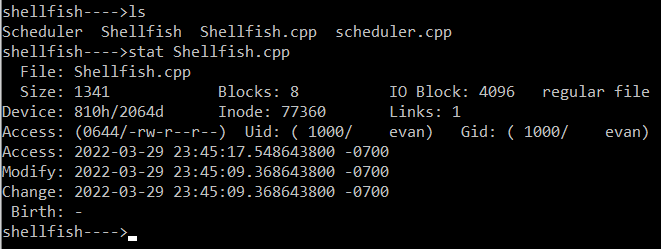
(Prerequisite: make sure tree is installed. If it is not, run the following command:

****

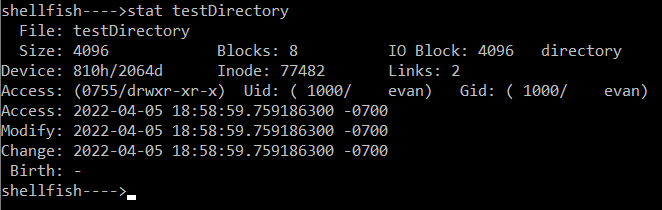
* 1. tree: recursively lists the directory given in the parameter, producing a depth-indented listing of files. If no parameter is given, it will default to listing the current directory.

****

1. **Get basic/detailed information about a file**
   1. stat: displays detailed information about a file, taking in the file’s name as a parameter.

****

1. **Get basic/detailed information about a directory**
   1. stat: displays detailed information about a directory, taking in the directory’s name as a parameter.

****

**Troubleshooting:**

The two main issues we ran into while going through the different additions of the file system commands were installing the tree command and using the cd command how it is set up in our shell. We were not able to perform the tree command right away in our shell. First we needed to type in the command sudo apt-get install tree to download the command before we could use it. Secondly, for our version of cd, it will only switch directories in the system call instance it is given. For example, if we had a directory we wanted to access and look inside, we would have to cd to the directory and then in the same line use a semicolon and the ls command to see inside.

**Correct and Incorrect Way For Our Shell:**

Shellfish→cd newdirectory; ls Shellfish→cd newdirectory

Shellfish→ls

(Correct) (Incorrect)

This is because the system() command will not permanently change directories, only temporarily. This was something that we decided not to change yet because it still has the full use of a Linux command line interpreter but it just has to be used slightly differently.

**Conclusion:**

By implementing the Linux command line interpreter using a general system call that will process each command, it allows the shell to have the full functionality of a Linux shell with the slight change in the change directory command. It is essential to have proper file system management commands because otherwise it will become too difficult for the user to find the correct files, reducing the usability of the shell. Due to the simple nature of the program that we have, it would allow us to easily add more functionality in the future and incorporate the shell into other projects.