Default Project

An extension of Fabricated Files & Folders

Tanner Hermann

Computer Science Department

Missouri University of Science and Technology

Rolla, MO 65409

thkt6@mst.edu

Abstract— This work focuses on emulating a shell using C++. The project involved extending the functionality of the original homework assignment from the Comp Sci 3800 Operating Systems course held in the Fall 2019 semester.

I. Background & Introduction

The starting point for this project was my submission for the "Fabricated Files 'n Folders" assignment. The original project was designed to emulate a Linux-style shell system. Upon starting the program the user is able to input commands to navigate through file systems and folders. The original functionality included creating "fabricated" files and folders that the user can traverse through and manage in a shell style system. I used my feedback from the original homework assignment to fix errors and expand the functionality to support the features laid out in the default project requirements. The added functionality to the shell emulator allows for functional permissions, users and groups, as well as process execution and scheduling.

II. Functional Permissions

Adding functional permissions to the system required a lot of work. The functional permissions meant that the shell would check whether or not a user had the right permissions to perform a task. Changes were made that checked each time a user entered a command if they had the permission level to perform it. Each file and folder in the system now has a permission string attached to it indicating read, write, and execute permissions. For directories; read allows the contents to be listed, write allows contents to be modified, and execute allows the directory to be part of the current path. For files; read allows the file to be read, write allows the file modification time to be updated, and execute allows the file to be ran. Write on both a directory and file allows for attribute updates with multiple commands. Attributes can be updated with the chmod, chown, and chgrp commands.

III. Users and Groups

In order to each part of the permission string usable, users and groups needed to be added. Adding users and groups to the shell adds a lot of functionality to the program. The user of the shell is able to add new users and groups as well as remove them. Each file and folder is also attached to a user and group that "owns" it. A user can be a member of multiple groups but each user only has one primary group. When the shell is first loaded up, the default user is set as root in the group "users".

IV. Process Execution and Scheduling

The final portion of the project included implementing a fake process execution system to the shell. The commands that worked with the process execution included ./<file>, ps, kill <file>, and schedHist. In order to keep things simple the run command "executed" any file that was sent to it. I chose to implement the "Round Robin" scheduling algorithm. I used this algorithm because it is simple and it was one of the algorithms used in a previous homework assignment. For my implementation I was able to use some code from the "Scheduler Showdown" assignment, which saved me a lot of time. The ps command allows the user to view any processes that are currently running along with relevant details on the process including process name, scheduled time, how long the process has been running, etc. The kill command immediately stops the indicated process and removes it from the scheduler. The schedHist command gives a detailed list of previously scheduled processes and times.

V. Conclusion

Overall I am very happy with my decision to choose the default project. I originally chose the project since I could not come up with any creative ideas for my own project. However I feel that I learned a lot working on this project and it was a good conclusion to the CS3800 course. I believe that the default project encompassed most of the topics covered throughout the course. My submission followed the default project guidelines with only a few errors. I look forward to expanding the shell emulator and adding more features to it in my free time. After finishing the project I feel that it is something that I worked hard on and I am glad that I chose it.