CIS 415 Operating Systems

Project 1 Report Collection

Submitted to:

Prof. Allen Malony

Author:

*Ellison Joanne Schilling/Largent*

*ellisons*

*951874139*

**Report**

**Introduction**

*The following project is a C-based implementation of a pseudo-shell that is capable of operating a series of commands. The project allows the user to operate the shell in two modes, interactive, and file. Interactive mode launches a pseudo-shell where the user can either type a command and its arguments and then press enter to submit them or submit a list of commands delineated by the ‘;’ character. The user is able to execute a series of different commands from Linux, that are now implemented via the “command.c” file algorithms and functions.*

**Background**

For this project, I utilized various system calls to implement essential commands. One of the key methods I used was the open() system call, which I employed with different flags to achieve the desired file operations. For instance, when implementing the copyFile function, I carefully evaluated the available flags and opted for the O-CREAT flag as one of my flags in order to ensure the creation of a file if the file did not already exist. I made this choice as it aligned better with the requirement of handling non-existent files. I also encountered challenges related to access permissions and ensuring proper file creation. To address these issues, I implemented more error handling code, including an additional error case for situations where the file failed to be created. This was particularly important for maintaining robustness and providing informative feedback to the user.

Another critical aspect of my implementation was buffer management. Initially, I struggled with ensuring that buffers were getting flushed when needed, especially in the main file where I used print statements instead of write statements. To resolve this, I incorporated flush statements, which I had only used once before. However, during my research, I came across an article that provided a comprehensive explanation of buffer flushing in various scenarios, which ultimately made me decide to stick with flushing as opposed to utilizing only write statements. Throughout the project there was much looking into many of the syscalls, many of which I had never used before such as rename(), and unlink().

**Implementation**

*Talk about your implementation of the project here. If there is anything nifty that you tried talking about it here too. If you had problems then talk about that here as well.*

*You can use Notepad++ to copy in code snippets into your document if you want. Just highlight some code, right click and select* ***Plugin Commands -> Copy text with Syntax Highlights.***

int someFunction**(**int param**)**

**{**

/\*some comment \*/

int someVar **=** 5**;**

someVar **=** someVar**+**param**;**

**return** someVar**;**

**}**

*Figure SEQ Figure \\* ARABIC 1: Some Algorithm implementation*

**Performance Results and Discussion**

*Write about the performance of your project. Give any performance results using standard performance metrics here (i.e. if in the description we say the project needs to have certain output then measure the output of your code vs. that metric). Show output from the console or from your application here if necessary (as a picture or a table). If your code does not run to specification, then explain why here. We will be more understanding if your issues are well documented. If your code does not run, and there is no explanation in either your comments or report, then you’re not leaving us with many choices concerning your grades.*

**Conclusion**

*This was a very thorough project, and while I feel like I gained a lot from it, I think the way in which I approached it could have been smoother. When I went about designing my project, I tried to finish most of it so that I could compare against the output files given, however, in retrospect I think divvying out the files a bit more would have made the entire assignment a bit less confusing.*