415 Operating Systems

Project <1> Report Collection

Submitted to:
Prof. Allen Malony

Author:
Edison Nayland Mielke
edisonm
951706836

Report

Introduction:

Project 1 was designed as a way to introduce us to operating systems in the way of having us build a basic command line. This project is set such that we had to develop C equivalents of the following unix commands: ls, pwd, mkdir, cd, cp, mv, rm and cat. Additionally, we were not allowed to use library commands in command.c, only being allowed to use system calls and string.h for simplicity sake. There was no such restriction on main.c or any other file.

Background:

I was unfamiliar with many of the system calls and because of that I had to read up as I coded significantly more than I would have liked. However the Labs were exceedingly helpful. Lab 1's skeleton.c did help accomplish a massive portion of the project as well as the string_parser elements of Lab1. A large portion of the code boiled down to:

```
-if (strstr(input,"command"))
-        //check parameters
-        command()
```

And while there are exceptionally more neatly designed ways of accomplishing this, this was more than usable during my time working on the project.

Implementation:

My *intention* was to make a fairly modular program that was very heavy in main.c and supplemental files that way I could have as much freedom as possible before working in the more restrictive command.c. and while I did take advantage of this, the code was significantly more brute force-ish and ugly than I had anticipated. I am rusty with C and I suspect my next project will be much cleaner looking. The main.c file generally handled all the organization while the commands.c file handled all of the actual logic involved and I probably could have broken up the monoliths that were file_mode() and command_line_mode() further. Which is valid.

Performance Results and Discussion:

In particular the "cat" command has an error in valgrind while not having any memory leaks, so I left that alone. There exists a "still reachable" memory leak if you ctrl+c the command line mode as it doesn't have the opportunity to free it. You can alternatively type "quit" and this keeps it from leaking at all. I tried to meet the output style requirements based on how I read the notes and the images given. I "BELIEVE" it's built to your specifications sans the cat issue. The code doesn't seem to hang at any point, and it doesn't have any memory leaks as long as you don't ctrl+c, so I am happy with the results.

Conclusion:

Overall I am fairly happy with the overall results of the program, there are some minor issues that I had of course (in particular, the cat command), but ultimately those are something that if I had started a day earlier I probably could have ironed them out. This project was very enjoyable once I got the hang of everything going on. You probably won't like me saying this, but I discovered a lot of little cheaty ways of accomplishing these problems. That isn't really a good or a bad thing per se, but I thought it was worth mentioning.