Feng Qiu (Tiger) & Jun Zhou

Joann J Ordille

CS406: Operating system Project 1: TigerShell

TigerShell

The goal of this project is to implement a shell that supports basic linux shell functions in C.

Authors

Feng Qiu (Tiger): qiuf@lafayette.edu

Jun Zhou: zhouj@lafayette.edu

Obstacles

One of the most difficult challenges that we eventually overcame is the design of pipes. Initially, we struggled to implement the functionality because we did not realize that there are build-in functions such as pipe() to help us implement pipes. We tried to find a way to communicate two processes so that the output of the first program could be sent to the second program. However, there were no available signals in the system that we could use to set up a stable communication between two processes. Then, we thought about the idea of creating a global variable or global space in the memory in which we can temporarily store the output of the first program so that the second program can access it. But we struggled to allocate the space

and type for this memory space because the space used function output can be different across different processes. Also, it is hard to keep this memory space available in all processes because other processes may interfere with it. In the class on Wednesday, Prof. Ordille mentioned the pipe() system call that was widely used in processes connection. It involves creating a virtual file so that one process can write the output on that file and the other can read. This system call perfectly solved our problem and we utilized it to make the pipe working.

Besides that, we also encountered troubles when implementing signal handlers. They were tricky since we used them in dealing with many different cases. For instance, we needed to take care of not only successful runs but also error cases. We managed to examine the exit status of all the children the shell created in order to determine if there was an error happening. Maintaining the job list was also a challenge. The shell we designed eventually achieved all the functionalities required under many iterations of tests and bug fixes.

Learning outcome

This project enhanced our understanding and gave us actual coding experience of the job control system. Job and process control was crucial to the operating system since there were multiple processes running in the background. This project corrected our previous misunderstanding of the background and foreground job. It also provided us with valuable experience with signal handling problems. Along with the lecture and homework we had in the class, we became more familiar with concepts on system calls.

Also, after this project, we came to realize the hardship that those pioneer computer scientists went through to create the exhaustive and efficient Linux shell system. Those powerful and complete system calls and functions were crucial for the development of our system.

Enhancements

For enhancements on this project, we intended to implement more build-in commands and support more system functionalities. For example, we found the interactive command-line editing functionality extremely useful in the Linux shell. This enables the system to automatically correct spelling, complete partially-typed filenames, and more. We intended to implement this system because it provides better user interaction with the shell.

Also, this project runs on a Linux command line (shell) currently. In the future, it is possible for us to implement our own GUI for the shell so that it runs independently from the current Linux shell so that it can be used by companies and individuals.