Arif Ayarci

CIS-3207-01

3/23/2022

Design Documentation

**Project 2: Developing a Linux Shell**

**Introduction:**

In this Project 2, myshell is a program that implements and reenacts the functions and utilities of a Linux/Unix shell. The shell supports the use of built-in commands, I/O redirection, piping, external commands, and background execution like any Linux shell would. Myshell takes advantage of an interactive mode, given by default, and a batch mode in order to execute these functions.

**Design Description:**

**Modes:**

* Myshell by default will run in interactive mode which takes user input straight from the keyboard one line at a time.
* Myshell that takes one and only one batch file argument will run in batch mode which takes input straight from the batchfile one line at a time.
* Both modes will function in while loops, interactive mode will end when the user exits the shell manually and batch mode will end when the entire file has been parsed and the shell self-exits.

**Parsing and Tokenizing Input:**

* Both modes are parsed by the parse\_line function that make use of the getline() function to get input line by line and strtok() function to tokenize for white spaces.
* These tokens are then stored in the argv char array to be used in the functions implemented in this myshell program.
* Argc likewise will be incremented along with argv for it also to be used in these functions.
* While strtok() is not NULL, the tokens will continue to be stored in the argv array, whether it is user input from a keyboard using stdin or a text file using a file stream opened at the batch file.
* When iterating through argv arguments in functions, strcmp() function will be used to see what the input contains and their respected flag values will be incremented along with it.

**Built-in commands:**

* Myshell will make use of cd, clr, dir, path, environ, echo, help, pause, and quit as built-in commands.
* cd command will make use of getcwd() function to print the current working directory and chdir() function to change the current working directory.
* clr command will simply clear the contents of the shell by printing "\e[1;1H\e[2J".
* dir command will make use of the recursive\_dir() function that uses the readdir() function and struct dirent to print the contents of the current directory or given directory.
* path command will store argv arguments into a char array and make use of setenv() to take the “PATH” call and set the char array as its environment.
* environ command will make use of char \*\*envp call given in main. By incrementing through envp with a for loop, myshell can print the environmental variables.
* echo command will simply use printf and a for loop to print the argv argument after the first “echo” argument.
* help command will print the contents of the readme\_doc using the read\_file() function that makes use of fgetc() function in a while loop.
* pause command will make use of getchar() in a while loop to take a user input for an enter key. Until the user presses enter, the shell will be paused.
* quit command will simply exit out of the shell using with exit() command taking a parameter of 0.

**I/O Redirection:**

* I/O redirection will be implemented in myshell with the redirection() function. The function will make use of “<” for input redirection, “>” for output redirection truncated, and “>>” for output redirection appended.
* File descriptors will be implemented for both input and output redirections. Dup() and dup2() function will be used to duplicated processes and to redirect stdin and stdout to their respected file descriptors.
* I/O redirection in myshell will make use of forking to create a copy of the process and exec to run in the child of the fork to execute external commands.
* After I/O redirection is completed, fflush() function will be used to flush out the stdin and/or stdout. Dup2() function will be used to restore stdin and/or stdout.

**Piping:**

* Piping will be implemented in myshell with the pipe\_func() function. The function will make use of the “|” symbol for piping.
* A pipe consists of a reading end (fd[0]) and a writing end (fd[1]). Both ends will make the use of forking and exec in order to run external commands.
* The external command(s) on the left side of the “|” symbol will be read by myshell and the external command(s) on the right side of the “|” symbol will be written by myshell.
* The pipe function will make use of the dup2() function on both ends of the pipe the reading side being used for duplicating stdin and the writing end being used for duplicating stdout.

**Background Execution:**

* Background execution will be implemented in myshell with the background() function. The function will make use of the “&” symbol for background execution.
* The function will iterate through the argv arguments within a for loop and extract the number of “&” symbols in the arguments.
* Every function and utility will make use of this background function and will account for “&” arguments found in their respected jobs.
* If an “&” is not present, waitpid() function will be implemented to wait for a child process to change after a fork(). If an “&” is present, kill() function will be used to terminate the child process and waitpid() will not be used.

**Test Plan:**

* For testing myshell, I used Ubuntu software as the project would be required to be done in a Linux environment as there are certain libraries and functions exclusive to Linux that Windows does not have like #include sys/wait.h. A make file was used to make compiling easier as I am implementing a .h file for executing external commands.
* For testing myshell in interactive mode I simply ran the program with no arguments so it would take input straight from the keyboard (stdin).
* For testing myshell in batch file I simply filled a batch file with commands and ran myshell with that batch file as a command as it would take input straight from the file.

**Implementation:**

* myshell> cd //to test a built-in command
* myshell> cd <path> //to test if argc will be consistent in the while loops.
* myshell> ls -la > output.txt //to test output truncate redirection with a flag.
* myshell> ls >> output.txt //to test output append redirection.
* myshell> cat < output.txt //to test input redirection to print contents of output.txt
* myshell> ls -la | grep myshell //to test piping to grep myshell contents with a flag.
* myshell> sleep 10 & //to test background execution and to see if myshell sleeps.
* ./myshell batch.txt //to test batch mode and if commands in batch file will run.