Hajime Miyazaki — ID: 922402751
Github: hMiyazaki95 — CSC415 Operating Systems

Assignment 3 - Simple Shell

Description:
The purpose of this assignment is to show how I implement my own shell that runs on top of the regular command line interpreter for Linux OS.

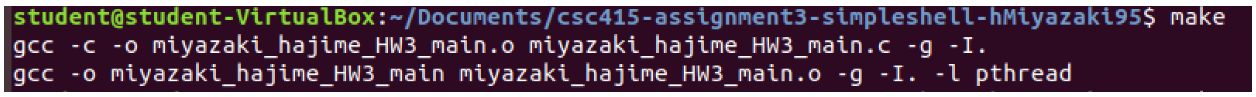
Approach / what I Did:

First I started with including header files stdio.h, stdlib.h, sys/wait.h, unistd.h, errno.h and string.h. Second, I created the function called readFromCmd, which reads user input from the command line and stores each line in an array of strings. readFromCmd() takes two arguments character array for user input and an array of the pointers to strings for storing each command. Next I used fgets(), which is referred from stdio.h to read user input from the command line and store it in the command line array. Then, I used the strlen() to get the size of the input and check if the last character is a newline. Otherwise replace it with null. After that, I used the strok() to create the token for the input string into each command with the space for delimiter.

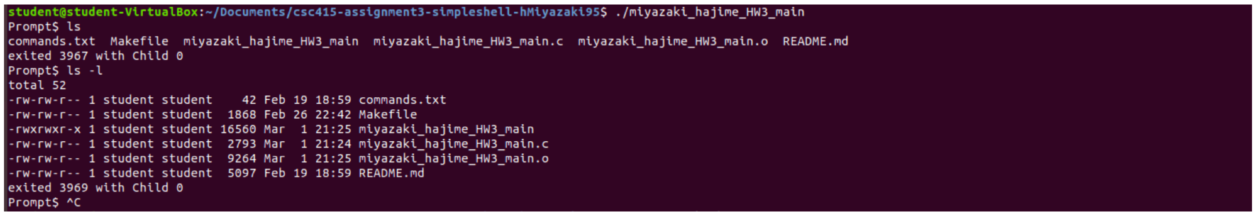
In my next step, in main() I used a variable of 1024 characters to buffer user input. Then, initialized args (a pointer array) to present a command prompt. I initialized the char array that points to the path and initialized the size of the array with a maximum path to 20. After that, I created an infinite while loop to keep prompting for input until the user hit the command 'end of file'. In the while loop, I initialized the character temporary array set to 50 to store user input strings from the command line in it. Then called readFromCmd() to read and tokenize input. Next I used fork() to create a child process and execute the user's stored commands in the child process by using execvp(). Then the parent process waits for the child process to exit, so I used waitpid(). Then, print the process id of the child process and then exit the child process status. Next I used the if statement to print an error message if an error code is returned.

Issues and Resolutions:
First issue encountered, Since I wasn't handling the space correctly, I encountered an error "No such file or directory". I needed to tokenize the command into smaller strings based on a delimiter which is a space to fix this error.

Screenshot of Compilation:



```
student@student-VirtualBox:~/Documents/csc415-assignment3-simpleshell-hMiyazaki95$ make
gcc -c -o miyazaki_hajime_HW3_main.o miyazaki_hajime_HW3_main.c -g -I.
gcc -o miyazaki_hajime_HW3_main miyazaki_hajime_HW3_main.o -g -I. -l pthread
```



```
student@student-VirtualBox:~/Documents/csc415-assignment3-simpleshell-hMiyazaki95$ ./miyazaki_hajime_HW3_main
Prompt$ ls
commands.txt  Makefile  miyazaki_hajime_HW3_main  miyazaki_hajime_HW3_main.c  miyazaki_hajime_HW3_main.o  README.md
exited 3967 with Child 0
Prompt$ ls -l
total 52
-rw-rw-r-- 1 student student    42 Feb 19 18:59 commands.txt
-rw-rw-r-- 1 student student  1868 Feb 26 22:42 Makefile
-rwxrwxr-x 1 student student 16560 Mar  1 21:25 miyazaki_hajime_HW3_main
-rw-rw-r-- 1 student student  2793 Mar  1 21:24 miyazaki_hajime_HW3_main.c
-rw-rw-r-- 1 student student  9264 Mar  1 21:25 miyazaki_hajime_HW3_main.o
-rw-rw-r-- 1 student student  5097 Feb 19 18:59 README.md
exited 3969 with Child 0
Prompt$ ^C
```