

## Assignment 3 – Simple Shell

### Description:

We will practice doing a simple command shell to practice the knowledge of process management, the child process, the parent process and the wait API. The program will read and execute the commands by forking/creating new processes after reading each word entered by the user.

### Approach / What I Did:

1. Open the VirtualBox, use terminal git clone assignment 3 from the Github
2. Using terminal create a c file by using “touch Gu\_Jiayi\_HW3\_main.c”
3. Reading the requirement from Github README.md
4. Think about that while fetching a user input line, the size of the buffer should not exceed 123 bytes and that the user input command argument lines should be allowed.
5. Try do figure out how to let the user input and how we can read the command line which are separated by space.
6. Using a while loop to display the required prompt, also need to make sure user can input each time of the loop runs. For this condition, I decided put the fgets() function inside of the loop to let the user input the command line.
7. Check some special cases, such as if user is trying to read a text file, we need to set up a EOF mention statements to let the users know that they’ve already at the end of the file, and the program will be closed after printout this mention (using exit to exit the program). Another special case is that if the user only input a ‘\n’ which is a Enter key on the keyboard, we should print out an error said that it is an invalid command argument to let the user know. The last special case should be the ‘exit’ case, the users might input exit when they want to end this program.

8. Consider if we store the user input by using `fgets()`, the `fgets()` function will also keep a NULL pointer at the last of the array. In this case, when I was trying to store each command arguments from the user input into an array by a for loop, I should also consider holding  $N + 1$  pointers at last.
9. Then, I start to consider to creating a child process by using `fork()` function and execute the output as we expected. When the child process create, we need to check if it is successfully created by using if statement to check the number of the `fork()` function's output. Since the order of execution between the parent and the child process is indeterminate, we also need to consider that if the parent process is faster than the child process. That's the reason why the parent process must wait upon the terminate of the child process.
10. All of the code seems work when I finished all the steps. I start to check each one and make sure my output is same as the example output. I found that we also need to print out the child process ID and the exit status of each time the while loop runs, so I set up a variables to printout the process ID and using the `WEXITSTATUS()` function to get the exit status of the child.

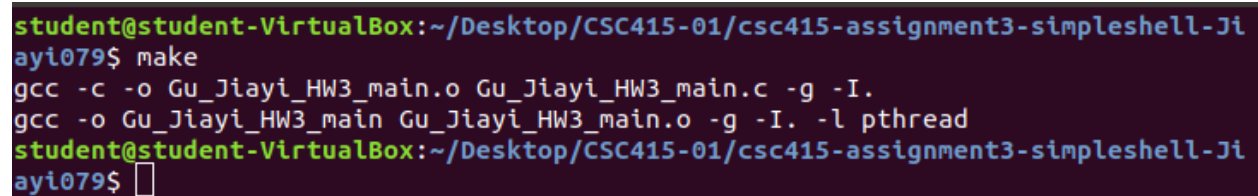
### Issues and Resolutions:

When I was trying to separate each user input command line into my `argv[]` array, I found that the output doesn't show what I expected, and it shows an error that seems like a '-l\$ \n' error in my output. Since I remembered that the readme file mentions that the `fgets()` function will also keep a NULL at the end of the command lines which input by the user, I thought maybe there were some problems happening in my separating the command arguments into arrays part, or maybe the last word of the user input doesn't store correctly due to I used the `fgets()` function. I tried to use a for loop to print out each argument the user input (line 73 – line 78 in my program), and I found that my guess is correct. Then, I tried to set the last point of the array to become NULL to make sure everything was stored correctly (line 71). Finally, this error is fixed!

Another issue I found in my program is that I was trying to initialize a char to store the correct path to use in the child process to execute and print out the expected output. In this way, I

found that even though my program can work, I still have some warnings said that about using the variables becoming a path to execute the child process. I started to think about if the user does input the 'ls' at the beginning of the command argument lines, why I can not just use it when I want to execute my child process? I tried to store the path by using the call "argv[0]" (line 102), and it works! All of the warnings are fixed!

#### Screen shot of compilation:

A terminal window with a dark background and light-colored text. The prompt is 'student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079\$'. The user enters 'make'. The output shows two gcc commands: 'gcc -c -o Gu\_Jiayi\_HW3\_main.o Gu\_Jiayi\_HW3\_main.c -g -I.' and 'gcc -o Gu\_Jiayi\_HW3\_main Gu\_Jiayi\_HW3\_main.o -g -I. -l pthread'. The prompt returns to 'student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079\$' with a cursor.

```
student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079$ make
gcc -c -o Gu_Jiayi_HW3_main.o Gu_Jiayi_HW3_main.c -g -I.
gcc -o Gu_Jiayi_HW3_main Gu_Jiayi_HW3_main.o -g -I. -l pthread
student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079$
```

#### Screen shot(s) of the execution of the program:

```
student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079$ make run <commands.txt
./Gu_Jiayi_HW3_main "Prompt> "
commands.txt      Gu_Jiayi_HW3_main.c  Makefile
Gu_Jiayi_HW3_main  Gu_Jiayi_HW3_main.o  README.md
prompt$ Child 2795, exited with 0
"Hello World"
prompt$ Child 2796, exited with 0
total 64
drwxrwxr-x 4 student student 4096 Sep 22 22:13 .
drwxrwxr-x 4 student student 4096 Sep 19 23:17 ..
-rw-rw-r-- 1 student student  42 Sep 19 23:18 commands.txt
drwxrwxr-x 8 student student 4096 Sep 19 23:18 .git
-rwxrwxr-x 1 student student 16032 Sep 22 22:13 Gu_Jiayi_HW3_main
-rw-rw-r-- 1 student student 3260 Sep 22 21:27 Gu_Jiayi_HW3_main.c
-rw-rw-r-- 1 student student 8800 Sep 22 22:13 Gu_Jiayi_HW3_main.o
-rw-rw-r-- 1 student student 1861 Sep 19 23:18 Makefile
-rw-rw-r-- 1 student student 5058 Sep 19 23:18 README.md
drwxrwxr-x 2 student student 4096 Sep 21 18:35 .vscode
prompt$ Child 2797, exited with 0
  PID TTY          TIME CMD
 2158 pts/0    00:00:00 bash
 2792 pts/0    00:00:00 make
 2793 pts/0    00:00:00 sh
 2794 pts/0    00:00:00 Gu_Jiayi_HW3_ma
 2798 pts/0    00:00:00 ps
prompt$ Child 2798, exited with 0
ls: cannot access 'foo': No such file or directory
prompt$ Child 2799, exited with 2
prompt$ [ERROR] Invalid command argument!
prompt$ [EOF MENTION] Thank you for using!
student@student-VirtualBox:~/Desktop/CSC415-01/csc415-assignment3-simpleshell-Jiayi079$
```