

Assignment 3 – Simple Shell

Description:

We will implement our own shell to run on top of the command line-interpreter for Linux, using `fork()` and `execvp()`.

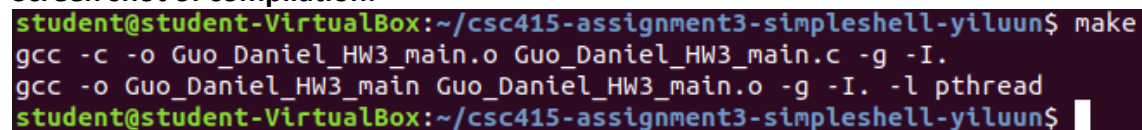
Approach / What I Did:

After reading the assignment, I defined `BUFFER_SIZE` to be 130 so that I remember to use it when creating the buffer that takes in the command line arguments. I broke down the inputs and stored them in an array that was half the size of the buffer + 1. This was to efficiently make a buffer that can hold the maximum number of possible chars (of char size 1) plus the NULL at the end (the + 1). I used `fgets` to get the inputs and `strtok` to process the input. I made small if statements to check for specific command line values like spaces/no command or “exit”. I utilized the `strstr` function to check if “exit” existed in the first parameter of `strstr`, and if it did I had it return 0 to exit. Once everything was tokenized, I forked the process and built if else statements to deal with the different values of the process. I knew that a return value of -1 meant a failure and a return of 0 meant success.

Issues and Resolutions:

My first issue was figuring out the structure of the code I was going to create. I also had to look up a lot of functions because for every function, there’s a lot of variations that you need to understand to know when to use them. I had issues with my delimiter for parsing stdin. Even though I used an empty space as a delimiter, I was getting an error where it wouldn’t give me the right output. I had to ask a tutor for help and they didn’t really know either, but eventually I fixed it by adding more delimiters like the tab and newline. I’m still confused on how this fixed it because my commands being input did not use tabs or newline, and only spaces. Another issue was that it wasn’t printing out my process id and result, and I solved it by changing the last statement from else if to just an else statement. Segmentation fault was terrible and because I wanted to avoid making a mistake, I ran my program after every small change to make sure I didn’t break anything. It’s also difficult to search up the error codes, because some of them are vague and don’t get any search hits, so I can’t really find any solutions.

Screen shot of compilation:



```
student@student-VirtualBox:~/csc415-assignment3-simpleshell-yiluun$ make
gcc -c -o Guo_Daniel_HW3_main.o Guo_Daniel_HW3_main.c -g -I.
gcc -o Guo_Daniel_HW3_main Guo_Daniel_HW3_main.o -g -I. -l pthread
student@student-VirtualBox:~/csc415-assignment3-simpleshell-yiluun$
```

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

```
student@student-VirtualBox:~/csc415-assignment3-simpleshell-yiluun$ make run
./Guo_Daniel_HW3_main "Prompt> "
prompt$ ls -l
total 44
-rw-rw-r-- 1 student student  42 Jun 21 08:18 commands.txt
-rwxrwxr-x 1 student student 11904 Jun 25 21:39 Guo_Daniel_HW3_main
-rw-rw-r-- 1 student student  2232 Jun 25 21:39 Guo_Daniel_HW3_main.c
-rw-rw-r-- 1 student student  8296 Jun 25 21:39 Guo_Daniel_HW3_main.o
-rw-rw-r-- 1 student student  1774 Jun 24 14:50 Makefile
-rw-rw-r-- 1 student student  5046 Jun 21 08:18 README.md
Child 3021, exited with 0
prompt$ ls foo
ls: cannot access 'foo': No such file or directory
Child 3021, exited with 2
prompt$ exit
student@student-VirtualBox:~/csc415-assignment3-simpleshell-yiluun$
```