

## Assignment 3 – Simple Shell

### Description:

This assignment is to write a simple shell in C much like bash. The user should be able to input text to a command line interface and have those commands run. This input from the user is initially input as a string and needs to be parsed into a subset of strings. Similar to the command line arguments for bash, the simple shell should handle the input from the user the same way. The first argument is the command itself and the following arguments are the command line arguments.

From github: “For this assignment you will implement your own shell that runs on top of the regular command-line interpreter for Linux.

Your shell should read lines of user input, then parse and execute the commands by forking/creating new processes. For each command, your shell should call `fork()` followed by `execvp()`. Following each command, your shell should wait for its child process to complete, and then print the child PID and the return result from the child process. The user should be able to specify the command to execute by giving a path to the executable file (e.g. `/bin/ls`) or by using path expansion to locate the executable file (i.e. searching each directory in the `PATH` environment variable). “

### Approach / What I Did:

The first thing I did to start this project was to organize the steps of how the program should run. Namely, I broke down how the function should work for each step of the Fork-Execute-Wait process. The input is accepted into one large string of a specified buffer size. From this string, it is broken up into individual strings using the `strtok()` function to use a space as a delimiter. Each of these strings is then put into an array of pointers to hold each of them. This array works as the `argv[]` for the shell that will run. Once we have that we can actually use them as commands/arguments. First we must use `fork()` to create another process so that when the command gets called the shell isn't overwritten. The child process from `fork()` then runs the `execvp()` command which overwrites the child process with the command from the first element from the array of pointers. This is when the parent process then utilizes the `wait()` function. After the child terminates, it's PID is printed as well as the result from the return of it terminating. This three function process is contained within a while loop such that when a command is entered and the shell executes it, the parent process doesn't end and the user can enter another command unless they choose to exit.

### Issues and Resolutions:

My first issue was breaking up the string from the user input. The problem I faced was not understanding how delimiters worked. I ended up trying to program the process myself and it

took me a lot of time with very little progress. After searching online for a better way I came across a youtube video of a similar example for shell input. It introduced delimiters and the strtok() function which I had been trying to implement myself. It helped immensely.

The next problem I faced was the output of the pid from the fork() function. From the examples in class and online I was under the impression that setting a value equal to the fork() function was the pid itself. I suppose it is, because the if loop I have runs the child process when it's 0 for example but I mean the pid of the process itself. What I was looking for was actually the getpid() function. This is what I actually needed to print the pid and not that the child process is 0 because it always is.

The last problems I faced are ones that I couldn't resolve.

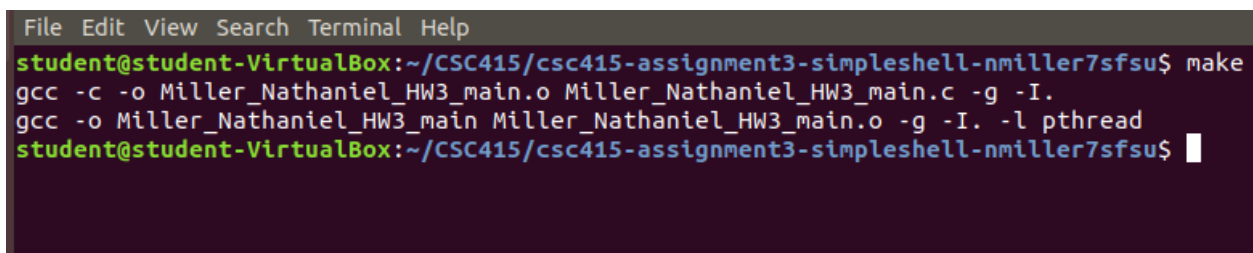
One is that I could not figure out how to get the shell to recognize Ctrl+D (EOF). I had used strcmp(), the string compare function, to recognize when the command exit was printed to end the shell. The problem was that if I included it to try and recognize EOF it is a different data type.

The other unresolved problem is that of the exit/return status of the processes. I attempted to use the waitpid() function while looking at the manpages as well as online resources and couldn't really figure out how the numbers were supposed to return.

### Analysis:

N/A?

### Screenshot of compilation:

A screenshot of a terminal window with a dark background. The terminal title bar shows 'File Edit View Search Terminal Help'. The prompt is 'student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu\$'. The user enters 'make', and the output is 'gcc -c -o Miller\_Nathaniel\_HW3\_main.o Miller\_Nathaniel\_HW3\_main.c -g -I.'. The user then enters 'gcc -o Miller\_Nathaniel\_HW3\_main Miller\_Nathaniel\_HW3\_main.o -g -I. -l pthread', and the output is 'student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu\$' followed by a cursor.

```
File Edit View Search Terminal Help
student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu$ make
gcc -c -o Miller_Nathaniel_HW3_main.o Miller_Nathaniel_HW3_main.c -g -I.
gcc -o Miller_Nathaniel_HW3_main Miller_Nathaniel_HW3_main.o -g -I. -l pthread
student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu$
```

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

```
File Edit View Search Terminal Help
student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu$ make run < commands.txt
./Miller_Nathaniel_HW3_main "Prompt> "
commands.txt Makefile Miller_Nathaniel_HW3_main Miller_Nathaniel_HW3_main.c Miller_Nathaniel_HW3_main.o README.md
Prompt> Child 5774 exited with 0
"Hello World"
Prompt> Child 5775 exited with 0
total 60
drwxrwxr-x 3 student student 4096 Sep 22 21:20 .
drwxrwxr-x 5 student student 4096 Sep 21 12:41 ..
-rw-rw-r-- 1 student student 42 Sep 22 21:11 commands.txt
drwxrwxr-x 8 student student 4096 Sep 21 12:42 .git
-rw-rw-r-- 1 student student 1869 Sep 21 12:46 Makefile
-rwxrwxr-x 1 student student 16024 Sep 22 21:20 Miller_Nathaniel_HW3_main
-rw-rw-r-- 1 student student 2399 Sep 22 21:19 Miller_Nathaniel_HW3_main.c
-rw-rw-r-- 1 student student 8360 Sep 22 21:20 Miller_Nathaniel_HW3_main.o
-rw-rw-r-- 1 student student 5058 Sep 21 12:42 README.md
Prompt> Child 5776 exited with 0
  PID TTY          TIME CMD
 1882 pts/0    00:00:01 bash
 3686 pts/0    00:00:00 vim
 5771 pts/0    00:00:00 make
 5772 pts/0    00:00:00 sh
 5773 pts/0    00:00:00 Miller_Nathanie
 5777 pts/0    00:00:00 ps
Prompt> Child 5777 exited with 0
ls: cannot access 'foo': No such file or directory
Prompt> Child 5778 exited with 0
Segmentation fault (core dumped)
Makefile:59: recipe for target 'run' failed
make: *** [run] Error 139
student@student-VirtualBox:~/CSC415/csc415-assignment3-simpleshell-nmiller7sfsu$
```

: