Andrei Georgescu

CSC415 – 01

SFSU ID: 920776919

Github: https://github.com/Doxify

Fall 2020

Simple Shell Program

What does it do?

The simple shell program is a basic recreation of the Linux shell. The purpose of the program is to get a better understanding of how processes work and how to use them in my own programs. The simple shell uses three system calls which are all related to processes: fork, exec (execvp), and wait. Fork is responsible for creating another instance of the program in a child process, execvp is responsible for executing commands, and wait is responsible for determining when the child process completes its execution.

What did I do?

- **georgescu_andrei_HW3_main.c**

  Main acts as a wrapper for the Shell and all it does is create a shell, initialize it, start the shell loop and waits for the shell to complete its execution. From the very beginning, I approached this project from an OOP point of view, and I think that keeping main simple and not defining any shell functions in main accomplishes this. In theory, the way the program is designed makes it extremely easy to launch multiple shell instances.

- **Shell.h / Shell.c**

  The shell program works by creating a Shell struct and uses this struct throughout the entirety of the program. Each shell struct consists of the following:

  - **bool running** – returns if the shell is running or not.
  - **char *prefix** – stores the shell's prefix
  - **char* buffer** – stores the shell's buffer, unparsed.
  - **char** args** – stores the shell's buffer, but in an array of strings.
  - **int argsCount** – stores the amount of arguments in char** args.

  The shell header file also consists of functions that are implemented by the Shell in order to carry out its functions and they consist of the following:

  - **int init(Shell *)** – The init function is responsible for initializing the shell. It allocates memory to **buffer** and **args** as well as set default variable values. If a shell is already running and this function is called, nothing happens.

  - **void run(Shell *)** – The run function acts as the shell loop and as a result this function is very important. Each shell loop uses helper functions described below to grab user input and tokenize it. After input is tokenized, the shell loop forks via system call and two things happen: In the child process the shell uses the exec system call to execute the commands in **args** and in the main process the shell uses the wait system call in order to determine when

the child process has finished terminating so that it can display it's return result.

- **void shutdown(Shell *)** – The shutdown function is called after a shell has stopped running and is responsible for freeing up any allocated memory by the shell.

- **void freeArgs(Shell *)** – The freeArgs function is a simple helper function that loops through the args array and frees up the memory taken by each pointer in the array as well as the array itself.

- **int getUserInput(Shell *)** – The getUserInput function is a helper function that prompts the user for input and uses **fgets** to store the input in the shell's buffer. If successful, the function returns 1, but if not, it returns 0.

- **int parseBuffer(Shell *)** – The parseBuffer function is a helper function that parses the shell's buffer. It dynamically allocates memory for **args** and uses the **strtok** function to tokenize the Shell's buffer. If the user's input is successfully parsed, it returns 1, but if an error occurs and the shell should exit, it returns 0.

Output:

- Shell Output

```
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify  make run
./georgescu_andrei_HW3_main "Prompt> "

Prompt> ls foo
ls: cannot access 'foo': No such file or directory
Child 24222, exited with 2
Prompt> ls -la
total 88
drwxr-xr-x 4 doxify doxify  4096 Sep 25 22:44 .
drwxr-xr-x 5 doxify doxify  4096 Sep 19 17:23 ..
drwxr-xr-x 8 doxify doxify  4096 Sep 25 22:03 .git
drwxr-xr-x 2 doxify doxify  4096 Sep 24 19:18 .vscode
-rw-r--r-- 1 doxify doxify  1744 Sep 23 16:16 Makefile
-rw-r--r-- 1 doxify doxify  4857 Sep 19 17:23 README.md
-rw-r--r-- 1 doxify doxify  6547 Sep 25 22:44 Shell.c
-rw-r--r-- 1 doxify doxify   979 Sep 25 21:10 Shell.h
-rw-r--r-- 1 doxify doxify 12240 Sep 25 22:44 Shell.o
-rwxr-xr-x 1 doxify doxify 23568 Sep 25 22:44 georgescu_andrei_HW3_main
-rw-r--r-- 1 doxify doxify  1222 Sep 25 22:43 georgescu_andrei_HW3_main.c
-rw-r--r-- 1 doxify doxify  7656 Sep 25 22:44 georgescu_andrei_HW3_main.o
Child 24248, exited with 0
Prompt> git pull
Already up to date.
Child 24262, exited with 0
Prompt> ls -l -a -d -f
.
Child 24306, exited with 0
Prompt> quit
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify
```

- Linux (Ubuntu) Output

```
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify  ls foo
ls: cannot access 'foo': No such file or directory
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify  2  ls -la
total 88
drwxr-xr-x 4 doxify doxify  4096 Sep 25 22:44 .
drwxr-xr-x 5 doxify doxify  4096 Sep 19 17:23 ..
drwxr-xr-x 8 doxify doxify  4096 Sep 25 22:03 .git
drwxr-xr-x 2 doxify doxify  4096 Sep 24 19:18 .vscode
-rw-r--r-- 1 doxify doxify  1744 Sep 23 16:16 Makefile
-rw-r--r-- 1 doxify doxify  4857 Sep 19 17:23 README.md
-rw-r--r-- 1 doxify doxify  6547 Sep 25 22:44 Shell.c
-rw-r--r-- 1 doxify doxify   979 Sep 25 21:10 Shell.h
-rw-r--r-- 1 doxify doxify 12240 Sep 25 22:44 Shell.o
-rwxr-xr-x 1 doxify doxify 23568 Sep 25 22:44 georgescu_andrei_HW3_main
-rw-r--r-- 1 doxify doxify  1222 Sep 25 22:43 georgescu_andrei_HW3_main.c
-rw-r--r-- 1 doxify doxify  7656 Sep 25 22:44 georgescu_andrei_HW3_main.o
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify  git pull
Already up to date.
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify  ls -l -a -d -f
.
doxify  master  ~ > CSC415 > assignment-3-simple-shell-Doxify
```