CSC 385 Final Project

Jay Bang

Saint Martin's University

Dr. Guyangyan Li

Abstract

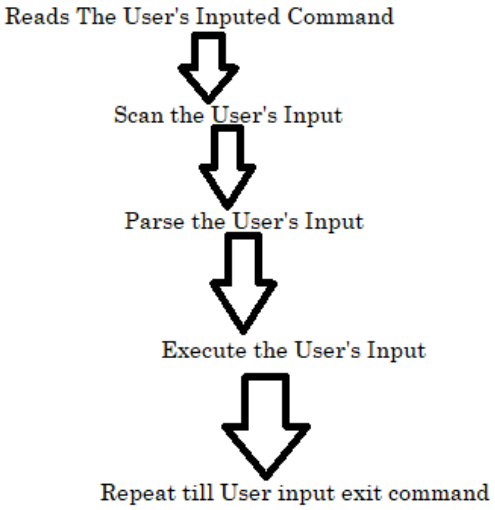
Linux shell is maybe one of the most important features of the operating system. Their main purpose to allow users access to the Linux operating system. The shell is the main focal point between the machine and the user, in which the main priority for a shell job is to allow the user to run command. The shell will then translate it for the kernel to run the user's command. This research paper will go over what is needed for a Linux shell with the primary language of C. It will also go over the difficulties I faced while performing my research and the knowledge I gained from it. Finally, this report will gain the futurity of the project if I do intend to finish the project on my own time in the future. Also, the source code that I derived from a tutorial on how to create a Linux shell is included in the folder during submission with the rights towards the author in the readme file. The code that I derived will also be there in which for a better understanding of the code I changed the function name. Which made me make sure that I was not autopiloting rather, attempting to learn what I feel as maybe one of the most intriguing projects I have done.

Linux shell, probably one of the major important features in the operating system. The shell offers the users to be able to access the operating system, in which the user can put in commands and the shell would translate it for the kernel to execute the user's command. There are multiple popular versions of a Linux shell some popular one is BASH shell and the other one being C programming shell. Since our class was focused on C programming, I chose to do a C programming shell. Throughout this past week, I researched about the necessary needs of a shell with a concentration on C language. Doing so I laid out some goals that I wanted to achieve, the first one being obvious which is learning the shell and how it works. The second goal was to have my personally own running Linux shell. To achieve my second goal, I used a tutorial that I found on the web to help me guide with my project.

One of the components for a Linux shell is the ability to have a working user interface. To create a user interface, a Read Eval Print Loop <REPL> is essential, this loop would have to read the user command, parse the user command, and finally executes the user command. This will be runed into a loop until the user inputs a command that will close the shell. To accomplish this the source code main.c with the main(); will be the REPL.

The main component of the Linux shell is the ability to parse and execute simple commands. To accomplish this part of the shell the following code: source.c, scanner.c, parser.c, node.c, executor.c, and along with their respective header file. To parse and execute the simple command the shell first must go through the lexical scanning. A lexical scanning is the process of reading the inputted command one character at a time to find the next token. Which will then go through the phase called the tokenizing input, which is extracting all the token from the user's input.

With the tokens from the user's input, the code would then create an Abstract Syntax Tree. The Abstract Syntax Tree will then parse the token of the user input and after the token is



parsed it will go to the executor. The executor as it sounds will be the one that will execute the Abstract Syntax Tree. The following picture will show the process on how the Linux shell needs to accomplish. To be clear there are many other steps that is needed but due to time constraints I was only able to implement the scan, parse, and execute the user's command. Also, due to little knowledge of C programming trying to implement these four parts of the command was very challenging for me.

The final component that I was able to implement was the addition of the symbol table. The symbol table is essential towards the shell as it allows us to define shell variables with the ability to modify their values. Each of the shell variables in the symbol table will have its own unique key, this means that there can not be multiple same keys for different variables.

The shell that was created by following the tutorial I was able to accomplish one of the major components. Which was the ability to execute a simple command, to do this, I opened my terminal and ran the shell. Inside the shell is ran $ls -ls. The picture below is the result that was

```
osboxes@osboxes:~$ cd 385project
osboxes@osboxes:~/385project$ gcc -o shell executor.c initsh.c main.c node.c parser.c prompt.c scanner.c source.c builtins/builtins.c builtins/dump.c symtab/symtab.c
osboxes@osboxes:~/385project$ ./shell
$ls
builtins    initsh.c  node.h    prompt.c   shell     source.h
executor.c  main.c    parser.c  scanner.c  shell.h   symtab
executor.h  node.c    parser.h  scanner.h  source.c
$ls -ls
total 92
 4 drwxrwxr-x 2 osboxes osboxes  4096 Dec  8 17:18 builtins
 4 -rw-rw-r-- 1 osboxes osboxes  2518 Dec  8 17:17 executor.c
 4 -rw-rw-r-- 1 osboxes osboxes   179 Dec  8 17:17 executor.h
 4 -rw-rw-r-- 1 osboxes osboxes   660 Dec  8 17:18 initsh.c
 4 -rw-rw-r-- 1 osboxes osboxes  2253 Dec  8 17:17 main.c
 4 -rw-rw-r-- 1 osboxes osboxes  1359 Dec  8 17:18 node.c
 4 -rw-rw-r-- 1 osboxes osboxes   755 Dec  8 17:18 node.h
 4 -rw-rw-r-- 1 osboxes osboxes   687 Dec  8 17:18 parser.c
 4 -rw-rw-r-- 1 osboxes osboxes   140 Dec  8 17:18 parser.h
 4 -rw-rw-r-- 1 osboxes osboxes   518 Dec  8 17:18 prompt.c
 4 -rw-rw-r-- 1 osboxes osboxes  2086 Dec  8 17:18 scanner.c
 4 -rw-rw-r-- 1 osboxes osboxes   236 Dec  8 17:18 scanner.h
28 -rwxrwxr-x 1 osboxes osboxes 28136 Dec  8 17:22 shell
 4 -rw-rw-r-- 1 osboxes osboxes   366 Dec  8 17:19 shell.h
 4 -rw-rw-r-- 1 osboxes osboxes   953 Dec  8 17:19 source.c
 4 -rw-rw-r-- 1 osboxes osboxes   324 Dec  8 17:19 source.h
 4 drwxrwxr-x 2 osboxes osboxes  4096 Dec  8 17:17 symtab
```

retrieved when the command was executed. As you can see the first thing, I tested was the listing file service by itself. With the command just being $ls it prompt up the listed file in the program directory. Then I tested out with $ls -ls and it gave me the result that showed every file with a description towards each one of the files that was listed.

```
$dump
 Symbol table [Level 0]:
 ==========================
   No          Symbol              Val
 -----------------------------------------------
 [0000] SHELL                       '/bin/bash'
 [0001] SESSION_MANAGER             'local/osboxes:@/tmp/.ICE-unix/1548,unix/osboxes:/tmp/.ICE-unix/1548'
 [0002] QT_ACCESSIBILITY            '1'
 [0003] COLORTERM                   'truecolor'
 [0004] XDG_CONFIG_DIRS             '/etc/xdg/xdg-ubuntu:/etc/xdg'
 [0005] XDG_MENU_PREFIX             'gnome-'
 [0006] GNOME_DESKTOP_SESSION_ID    'this-is-deprecated'
 [0007] GTK_IM_MODULE               'ibus'
 [0008] QT4_IM_MODULE               'ibus'
 [0009] GNOME_SHELL_SESSION_MODE    'ubuntu'
 [0010] SSH_AUTH_SOCK               '/run/user/1000/keyring/ssh'
 [0011] XMODIFIERS                  '@im=ibus'
 [0012] DESKTOP_SESSION             'ubuntu'
 [0013] SSH_AGENT_PID               '1450'
 [0014] GTK_MODULES                 'gail:atk-bridge'
 [0015] PWD                         '/home/osboxes/385project'
 [0016] LOGNAME                     'osboxes'
 [0017] XDG_SESSION_DESKTOP         'ubuntu'
 [0018] XDG_SESSION_TYPE            'x11'
 [0019] GPG_AGENT_INFO              '/run/user/1000/gnupg/S.gpg-agent:0:1'
 [0020] XAUTHORITY                  '/run/user/1000/gdm/Xauthority'
 [0021] GJS_DEBUG_TOPICS            'JS ERROR;JS LOG'
 [0022] WINDOWPATH                  '2'
 [0023] HOME                        '/home/osboxes'
 [0024] USERNAME                    'osboxes'
 [0025] IM_CONFIG_PHASE             '1'
 [0026] LANG                        'en_US.UTF-8'
```

I also tested it out with the $dump command, when executed it would print out the full symbol table. The symbol table consisted with the number of the variable and the symbol variable name and finally with its value.

```
 [0027] LS_COLORS                   'rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=01;32:
*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.
lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.r
ar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35
:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=0
1;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli
=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka
=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:'
 [0028] XDG_CURRENT_DESKTOP         'ubuntu:GNOME'
 [0029] VTE_VERSION                 '6001'
 [0030] GNOME_TERMINAL_SCREEN       '/org/gnome/Terminal/screen/38bd7b58_2ac4_4d29_96e5_d5c8f23e0ec0'
 [0031] INVOCATION_ID               '211f5540086243 5baf311c6b8d3373b1'
 [0032] MANAGERPID                  '1238'
 [0033] CLUTTER_IM_MODULE           'ibus'
 [0034] GJS_DEBUG_OUTPUT            'stderr'
 [0035] LESSCLOSE                   '/usr/bin/lesspipe %s %s'
 [0036] XDG_SESSION_CLASS           'user'
 [0037] TERM                        'xterm-256color'
 [0038] LESSOPEN                    '| /usr/bin/lesspipe %s'
 [0039] USER                        'osboxes'
 [0040] GNOME_TERMINAL_SERVICE      ':1.123'
 [0041] DISPLAY                     ':0'
 [0042] SHLVL                       '1'
 [0043] QT_IM_MODULE                'ibus'
 [0044] XDG_RUNTIME_DIR             '/run/user/1000'
 [0045] JOURNAL_STREAM              '9:34233'
 [0046] XDG_DATA_DIRS               '/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop'
 [0047] PATH                        '/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin'
 [0048] GDMSESSION                  'ubuntu'
 [0049] DBUS_SESSION_BUS_ADDRESS    'unix:path=/run/user/1000/bus'
 [0050] OLDPWD                      '/home/osboxes'
 [0051] _                           './shell'
 [0052] PS1                         '$'
 [0053] PS2                         '>'
 -----------------------------------------------
```

```
$pwd
/home/osboxes/385project
$wc main.c
 116  223 2253 main.c
$
$du
12      ./builtins
16      ./symtab
116     .
$
osboxes@osboxes:~/385project$ ./shell
$exit
osboxes@osboxes:~/385project$
```

Due to curiosity, I ran a series of simple command to see if my shell that I was able to compile. Can indeed run other simple command. The first command I tested was the print work directory ($pwd), the result as shown in the very fist picture. The second picture showed the result of the second command I tested which was word count ($wc) for the file main.c. The third picture showed the result of the command disk usage ($du) and finally the last picture. To be a working shell, when a user inputs an exit command the shell should be exited. I tested it out with a brand-new terminal and ran the shell then executed the command. Which resulted it exiting the shell and prompting to the Linux terminal.

In all there are many different components toward a Linux shell however, I believe that I implemented the four most important components. The first being the Read Eval Print Loop which reads the user command, it will then parse the user command and finally executes the user command. The second and third components are correlated to one another, which is the ability to

parse the users' command and execute the users' command. To enable us to parse the users command it first needs to be able to extract the input token and create an Abstract Syntax Tree. Then it must be able to execute the users' command by executing the Abstract Syntax Tree.

While researching about this final project, I ran into variety range of problems. The first and foremost was with very little knowledge of the actual C language. It was very hard for me to figure out the best way to go about this project. I found multiple website that "showed" me how to build a Linux shell. However, when I read the instruction the finished product was not an actual Linux shell rather it was just a terminal that was made to look like a shell. However, one tutorial which I used to my best actually gave out the result I wanted.

The second problem I ran into was figuring out how to translate the C code that was given by the tutorial. Rather, due to my basic knowledge of C++ and C#, I felt like it did not take me a long time then I expected to understand the basic of the code that was given. I commented only in my C programming file with what each function would be used for. This gave me the chance to enhance my understanding of both the language C and what the shell is trying to do in the background that us human can not see.

The most common problem I ran into was the compiling error that I encountered. The compiling error that I received was due to my process of changing the function name to enhance my learning process. I realized how painful it was to make sure the compiling error was fixed in the Linux machine as the terminal would print out the error making it hard to keep on tract the hundred of the compiling errors that I was getting. To make my code work I spent hours on my virtual machine fixing all the compiling errors so I can show my result during the presentation. I had to research some error since I had a hard time figuring out what the error was on. But in the long end I am proud to say that my code was able to run and I was able to show my result to the class during my presentation.

For me personally the futurity of this project is rather broad and very huge. During the summer or maybe during the break, I would want to make my own Linux shell rather it being in BASH form or C programming from scratch. Since I was able to derive the code and make my shell running form a tutorial. I believe that I can use that knowledge and make my own Linux shell from scratch. Also, I kind of want to make a full-blown shell so if time permitted, I would want to look at the actual shell source code to see what else is needed for the Linux shell.

In all during my times coding and with compilation error, I believe gave me massive migraines and wanted to stop multiple times during the night. But the knowledge that I was able to gather with Linux shell was helping me able to finish this project. I did not expect the Linux shell to be this complex but knowing now I felt like this project was very good for me. Hopefully with the knowledge that I was able to attain will help me out in the near future. That is all I have; I hope you liked the research as much as I did during the process of studying for this project.

References

Isam, M. (2020, July 04). Let's Build a Linux Shell [Part I]. Retrieved December 12, 2020, from https://hackernoon.com/lets-build-a-linux-shell-part-i-bz3n3vg1

Isam, M. (2020, July 04). Building A Linux Shell - Part II [A Step-by-Step Guide]. Retrieved December 12, 2020, from https://hackernoon.com/building-a-linux-shell-part-ii-a-step-by-step-guide-pk203ywg

Isam, M. (2020, July 04). Building a Linux Shell [Part III]. Retrieved December 12, 2020, from https://hackernoon.com/building-a-linux-shell-part-iii-wzo3uoi

Isam, M. (2020, June 08). Building a Linux Shell [Part IV]. Retrieved December 12, 2020, from https://hackernoon.com/building-a-linux-shell-part-iv-h21o3uwl