Batuhan Yalçın

64274

COMP 304

Project 1

Custom Commands: starwars and copy.

# Run:

make clean

make

gcc -o shellfyre shellfyre.c

Then any command can be finded in below or Shell commands

# Part 1 )

Run: To run see the Run part shown in the upper.

From the given description:

“execv() System Function:

In execl() function, the parameters of the executable file is passed to the function as different arguments. With execv(), you can pass all the parameters in a NULL terminated array argv. The first element of the array should be the path of the executable file.” [1].

To write code I get to current environment then after using strok I parse the environment “:” then I add “/” to environment address to get correct binary path implementation then I add adrres to command name to use already define binary commands. Then with execv I get able to call already defined function with the given additional arguments.

Simple Output:

Text

Description automatically generated

# Part 2)

**File search:**

Run: To run see the Run part shown in the upper.

For that command I use <dirent.h>, <sys/stat.h> libraries which allow to use some directory functions and for the filesearch command I implement a filesearch() function.

The function mainly checks the argument number if argument number 1 after file search it directly take given keyword and search it. If argument number 1 but first argument -r or -o it return a message to user and says “Please specify the keyword you want to search”. Similar to that it checks the argument type until it get correct input format which is filesearch ("your search option eaither -r -o or both or nothing") keyword. If it not obey the correct format it warns says that: “"Expected format of filesearch \"your search option eaither -r -o or both or nothing\" keyword \n"”

For go through the directories and search recursively, I am calling function repeatedly and I am searching for each item in each runned directory if the item is directory, it calls function with the founded directory. It goes until all the directories visited. While going through the directories it searches all items and if fond any item includes given string it prints the item.

For the open files I used xdg-open function to open files with the default open program. Like searching when item found it opens the item.

Sample output:Text

Description automatically generatedA screenshot of a computer

Description automatically generated

**Cdh Command:**

Run: To run see the Run part shown in the upper.

For this question I used lots of the helper function which are given in below:

int closePipes();

int cdhChild();

int user\_Selection(int initialLength);

int cdhChildofChild(char\* name\_of\_command );

int cdhistory(char\* cd\_history\_file);

int countDirectory(char\* cd\_history\_file);

int readDirectory(char\* cd\_history\_file);

int openPath();

int correctDirectories(char\* cd\_history\_file, int size);

For this function I use <ctype.h> liblarary to check digit or letter charecters.

Since the function show the directory history it already running without any user command. For the cdh task first I use the cdhistory () function. The function puts the given directory to the cd\_history.txt file the file created in current shellfire directory with the openpath(). If the path reaches the 10 after each visit countDirectories(), readDirectories() and correctDirectories() functions runs and the fix the cd\_history.txt file and adds new visit history.

After each visit with cd take or cdh recorded the file. When user enter the command cdh

cdhChild() function runs. The function prints the history and asks for the user selection then userSelection() function calls after user select his/her choices the input passed the child process with the pipe then cdhChildofChild() function run and selected path open also the path recording to the history.

closePipe() functions basically closes the pipes.

Sample Output:

Text

Description automatically generated

**Take cd command:**

Run: To run see the Run part shown in the upper.

When user use take command it calls take() function. It basically takes directories as an input and parse the directories according to “/” with strok() function. For the each directory using mkdir() and chdir() functions goes around the path if path was not using it creates and changes current directory to the desired directory.

Note that cdh history only takes final path since the other directories not used as a current directory.

Sample output:

Text

Description automatically generated

**Joker command:**

Run: To run see the Run part shown in the upper.

When the user use joker command it calls the joker() function. The function basically using crontab function programs the system to call \*/15 \* \* \* \* XDG\_RUNTIME\_DIR=/run/user/1000 notify-send joke: \"$(curl -s https://icanhazdadjoke.com)\" function in each 15 minute. To do that I used the execlp() command because other wise it was not returning the different jokes execlp() calls the program with the given arguments in txt file. To get more information look the references [1]. Since all the students in discussion bord couln’t sen any text to terminal I shows the joke as an notify to the system with using notify send references [3].

Sample Output:

Graphical user interface, text, application, email

Description automatically generated

**Custom Commands**

**starwars command:**

Run: To run see the Run part shown in the upper.

To run just write command line to starwars.

When user enter the starwars command it basically runs a starwars movie in the terminal. But since the this command was very easy I wrote another custom command.

Sample output:

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface

Description automatically generated

**copy command:**

Run: To run see the Run part shown in the upper.

When user enter the copy command it runs copy function. The function basically copies given file another directory given. If the given directory already includes file, it warns the user and not changes the file. Otherwise, it copies the files to the given directory with char by char to copy byte by byte. Then turns back the initial directory.

Sample Output:

Graphical user interface

Description automatically generated

Graphical user interface

Description automatically generated

# Part 3)

Run: To run see the Run part shown in the upper.

For this part I create a module with the name of it Pstraverse.c. The modules consist of initialize() remove() DFS\_search() and BFS\_serach() functions and module parameters rootinfo to get root pid and search to declare search type. In make file it prepares the necessary kernel files.

With the shellfyre.c I send the user arguments to the module Pstareverse. After pstraverse command given argument pid sended to the module as a root parameter rootinfo. The second arguments describe search type -d or -b. This argument goes to module as char parameter.

There is a problem with when I wrote -b but I couldn’t figure out. Actually, there were no time the figure out, but program works fine with the -d with DFS algorithm.

Then the result of the search, prints to the kernel print screen with the dmesg function results can be seen.

Sample output:

Graphical user interface

Description automatically generated

Text

Description automatically generated

Github repo: <https://github.com/batuhanyalcin/COMP-304-Project-1.git>

# References:

[1] <https://linuxhint.com/exec_linux_system_call_c/>

[2] <https://stackoverflow.com/questions/8436841/how-to-recursively-list-directories-in-c-on-linux>

[3] [linux - Cron with notify-send - Stack Overflow](https://stackoverflow.com/questions/16519673/cron-with-notify-send)

[4] <https://icanhazdadjoke.com/api>

[5] <https://www.digitalocean.com/community/tutorials/how-to-use-cron-to-automate-tasks-ubuntu-1804>

[6] <https://phoenixnap.com/kb/set-up-cron-job-linux>