TALHA ENES GULER 64507
MURAT HAN AYDOGAN 64756

COMP304 - Project 1 Report

Part 1 :

In the first part of the project, we used execv() system call to execute a given command. The execv() requires the path of the flie of the entered command and its arguments. We write a function called execHelper(char*) which takes the name of the command and searches for the path of that command. To do that there exist specific file locations that the file of the command can be found which are "usr/local/sbin/", "/usr/local/bin/", "/usr/sbin/", "/usr/bin/", "/sbin/", "/bin/", "/usr/games/", "/usr/local/games/", "/snap/bin/". We Iterate over them one by one and if the mastches file is found then it returns the path. And we passed this path and the argument from terminal to the execv() system call.

Part 2:

In this part we need to design a command to save the current location and to execute other operations with them. In order to do that, we implemented hashmap similar structure by using struct and array of structs which keeps key and value attributes. The necessary functions are also implemented for adding new name-directory association, deleting one of them and all of them, listing the all list, and jumping the wanted directory. deleteAll() and mapPrinter() have similar structure that is iterating over the array and putting NULL or print the association. putMap(char*, char*) function takes the name and director pointers and put them into the map. To do that it first check the name whether there exists or not by calling hasVal(char*) function which also returns the position of the given key. If it exists then it changes the value of that key, otherwise it puts the key value pair to the first NULL element on the array. For the jump, find the directory associated with the given key(name), then call the chdir().

Other requirements that need to be satisfied is to keep the previous association created before. To implement that we have created two new seashell operations called shortdir save and shortdir load. “shortdir save” saves the current associations to a local text file called

“lines.txt”. “shortdir load” loads the associations at the local “lines.txt” files to our hashmap data structure. After setting associations or deleting them the user can choose to save newly modified map locally by using “shortdir save”. User can also load local associations using “shortdir load” and execute operations on those associations.

The execution of this command is done by the parent process unlike others. The reason behind this is that the child process can not change the map that is defined in the parent process. It can be implemented via pipelining however it makes the process more complicated. So we decided to call this command function on the parent process.

In addition to these, shortdir() function takes directly to the struct commad_t and processes it inside itself by checking the args[1] and passing the args[2] arguments to the appropriate functions.

Part3:

The command in this part executed with the highlight(char* word, char* clr, char* path) function and these parameters are passed to function in the child process. We implementing this part by opening the file from given path and reading the .txt file line by line by using fget() function. Then we put every line to the checking process by calling checker(). This function returns whether the given word is in the line or not. If it exists, then the tokenizer part begins. Each word in the line taken one by one with the strtok() function with space separator until it returns NULL. The word again compared with the given word and if they match the print color changes according to the given clr code with printf("\033[0;31m"). After every word print we returned to the original color. If the checker() returns 0 then, the line is skipped without printing.

Part4:

In this part, the crontab executes the line that is in the crontab.txt file to set the time for alarm and to play the wanted music on the alarm. The form of the txt file is :
* * * * * XDG_RUNTIME_DIR=/run/user/$(id -u) /usr/bin/rhythmbox-client --play-uri PATH

So the middle part “XDG_RUNTIME_DIR=/run/user/$(id -u) /usr/bin/rhythmbox-client --play-uri” of the line is the same for all function calls. The initial distinction is first five * which is for wanted time. The first * is set the minute and the second one is for an hour. And the PATH is the address of the music file that is wanted to play. Both of these will be provided from the user.

The main difficulty is fitting the given arguments to this form. In order to do that we have used string tokenizer with “.” separator to take a minute and hour separately. Then we created an array with the length of the total text. Then we append every string in order (min hour * * * ----- PATH). For string operation we always copied the given string to another string whose pointer is created with malloc and freed them after the process..

Part 5:

In this part, we have created two functions to compare files. kdiff function to compare .txt files and kdiffByte function to compare any type of file. In kdiff we take both files’ names from the user and open them using the fopen function. After that we calculated each file's line number by iterating over files line by line and closed both files and opened them again to start the comparing process. To compare files we look at each of the file's lines with getting them using fgets function in while loop. Before the comparing part program first checks if to be compared line is in the scope of available line numbers. This checking process is necessary because fgets file continues to give the last line of the file after lines are finished. If this happens the program equalizes “ “ string to that line and continues with the comparison process. To compare them to each other using strcmp function. If they are different we’ve printed the lines to screen with their line numbers. If they are the same nothing is printed to the screen and the process continues. After every line is checked our function prints how many different lines were found by incrementing the defined integer(diffCount) when the lines are different. For comparing them using bytes we’ve created two unsigned char arrays named BYTE array and fill those arrays with each character’s byte size in the files. While filling those arrays, we’ve also calculated the total size of each file by first summing the total bytes in the line by adding every element in the BYTE array and then we add every byte total of the line to the general byte total of the text. After this process is done we have compared two integers that show the files size. If they are equal to each other the program prints that

those two files are identical and if they are different the program prints that the two files are different and also prints how they differ in terms of bytes.

Part6:

In this part, we wanted to implement a command that includes computer science and art together. From this perspective, we utilized ASCII art which is a graphical design technique that consists of pictures pieced together from the 95 printable (from a total of 128) characters defined by the ASCII Standard. We found some of those art pieces and saved them to the txt file separately. Our command takes two arguments which are the name of the art and first letter of the wanted print color (r,b,g). After taking the argument, it finds the txt file that contains wanted art and then reads it line by line then changes the color of the printer and prints the image to the terminal screen.

How to run:

1. To run the project, the user must compile seashell.c file first. To compile it, the user first needs to open the terminal in the seashell.c files directory. After that the user needs to type the following two command to the terminal to run the program:
2. $ gcc -o seashell seashell.c
3. $ ./seashell
4. After entering these two commands, seashell will start executing and waits for user input. The user can either enter Linux commands or seashell commands. Seashell commands are as follows:
    - 4.1. shortdir set <name>
    - 4.2. shortdir jump <name>
    - 4.3. shortdir del <name>
    - 4.4. shortdir clear
    - 4.5. shortdir list
    - 4.6. shortdir load
    - 4.7. shortdir save
    - 4.8. highlight <word> <r | g | b > <filename>
    - 4.9. goodMorning <time> <.mp3 file path>
    - 4.10. kdiff -a <filename.txt> <filename.txt>
    - 4.11. kdiff -b <filename> <filename>
    - 4.12. kdif <filename.txt> <filename.txt>
    - 4.13. painter <ACSII art name (shrek, jordan, paris, earth)> <color (r g b)>
    - 4.14. exit (to terminate seashell)

Important Warning!

To execute the highlight and kdiff operations files that will be used in the operation need to be in the same file as seashell.c. Otherwise operation will fail. User be aware.