**PROJECT-1 REPORT**

**Part I - execv**

In this part we needed the path of the command to run it. So, we implemented a function called “getFilePath”. It takes the command name as argument and returns its path. We used the system call “which” to get the path and write the path into a temporary file. For example: which ls > temp.txt. Then we read the path from the temporary file, return it and deleted the temporary file. We gave the path as an argument to execv command.

**Part II - shortdir**

In this part we have implemented two structures. First is called “element” and stores name – path pairs. Second one is called list and it stores a list of elements(pairs) and pair counts. When program is executed, we first read the shortdir.txt file into the list structure in order to get previously set short directions.

***shortdir set:*** we implemented a function called “insert\_shortdir”. It creates an element and inserts it into the pair list. If the name already exists, it overwrites the path.

***shortdir jump:*** we found the path of the given name by using find\_shortdir\_path function. By using chdir command we have jumped to the path. The functions loops over the name of the pairs and returns the wanted path.

***shordir clear:*** we set the pair count to 0.

***shortdir del:*** we searched for the given name, and if the pair count is 1, we deleted it by setting the pair count to 0. If the count is larger, we have shifted the last element to the place of the deleted pair.

***shortdir list:* we implemented a function called**

Graphical user interface, text

Description automatically generated

**Part III - highlight**

In this part, we first divided the command parameters into three such as word, color and filename. Then we wrote a helper function and read the file entered by the user. The readfile method opens and reads the specified file, writes a to buffer string and returns it. After taking the whole file as a string, we compared all the words with the word that we will change the color of which is the command parameter. In this section, we used string comparison by converting it to lowercase letters to make case insensitive. While printing the words one by one, if the iterated word and the command parameter match, we highlighted the color of the word with the specified color and printed it.

**Part IV - goodMorning**

In this part, after reading the parameters entered in the terminal, we pulled the day of the month, month of the year and day of the week data from the computer's timer. Then we tried to put them in a new crontab file with the crontab -e command. However, we had difficulty here and we couldn't succeeded.

**Part V - kdiff**

**Part VI – whatTime?**

**Work-Load Distribution**

**Resources**

**[1] https://www.sanfoundry.com/c-program-number-lines-text-file/**