**COP304 Project 1 Report**

*Mehmet Mert Bezirgan - 68756*

*Umut Öztunç – 69264*

We used dual-booted Ubuntu 20.04 distribution throughout the Project and worked on VS Code and default Ubuntu terminal. For better coworking we opened a git repository too.

**PART I**

Because we do not use *execvp()*, which do not require a path of the command as a parameter; we need to find the path to execute *execv()* command which would execute the given command. We got path environment variable using getenv and we try all possibilities.

**PART II**

We start by opening a file with *a+ mode*, which created at the beginning of the main part by appending the text file name just after the path of the directory (because our *shortdir.txt* file is in the same directory with *seashell*, we need the full path which is the current path plus the corresponding text file name; if not, reaching *shortdir.txt* file after *cd* command would not be possible.). We save our first directory to global variable because when we change current directory with cd, we need to save our whole path.

We handled necessary commands in five *if clauses*.

For *set* command, we write the given commands into *shortdir.txt*.

For *del* command, we copy all but not the given second argument after *del*, in the *shortdir.txt* to a new text file (by comparing line by line via *strcmp()*). Then, we remove the old file and rename the new file as *shortdir.txt*.

For *clear* command, we open *shortdir.txt* via *write mode* and rewind the file.

For *list* command, we control the end of *shortdir.txt* via *fgets()*, and print out the names and directories consecutively.

For jump command, we get the path of given key and then change directory using chdir method.

**PART III**

We hold and capitalize the word that we look for, in a *char\* word*.We read the text file with *fgets()* function and control the end of file situation with *fgets() != NULL* utility. We separate lines by tokenizing via “ “ and we compare each word 1 by 1. While comparing the wordswe print them as well.

**PART IV**

After storing hour and minute info coming from the command line in the *char\**, we open a new file called *music.txt* (append mode) and put our hour and minute into there along with necessary commands for rhythm to play our music.

We used *execvp()* command to run *crontab* command inside our code, which tend to invoke the text file *music.txt*, and so, the alarm to wake us up at given time.

**PART V**

We read the text files with *fgets()* functions and control the end of file situation with *feof()*  function. While we do not reach at the end of file, we compare two files simultaneously line by line with *strcmp()* function.

For *-b* command, we control the loop with same technique but instead, we compare the bytes with *getc()* function.

We handle the *args* count and *-a*, *-b*, and *default* command inputs with straightforward *if* clauses and print appropriate outputs accordingly. Also, we handle the null file inputs and text file extension mismatches via tokenizing and checking with point in the filename, in relevant *if* clauses.We used strrchr to find last occurrence of “.” In a filename and check if that file is a txt file.

**PART VI**

We would like to print results of an operation (addition, subtraction, division, multiplication, power) on two numbers given in a *txt* file. The command name is *solve* and it takes the file that has the expressions in.

We read the file line by line with *fgets()* function and control the end of file with that function so. After tokenizing the numbers and the operation type and storing them in *char\**, we apply corresponding operation via *if clauses*, regarding the basic mathematics principles. Then, we print the inputs, operations, and results, consecutively.