COMP 304 - seashell - Your Custom Shell: Project 1

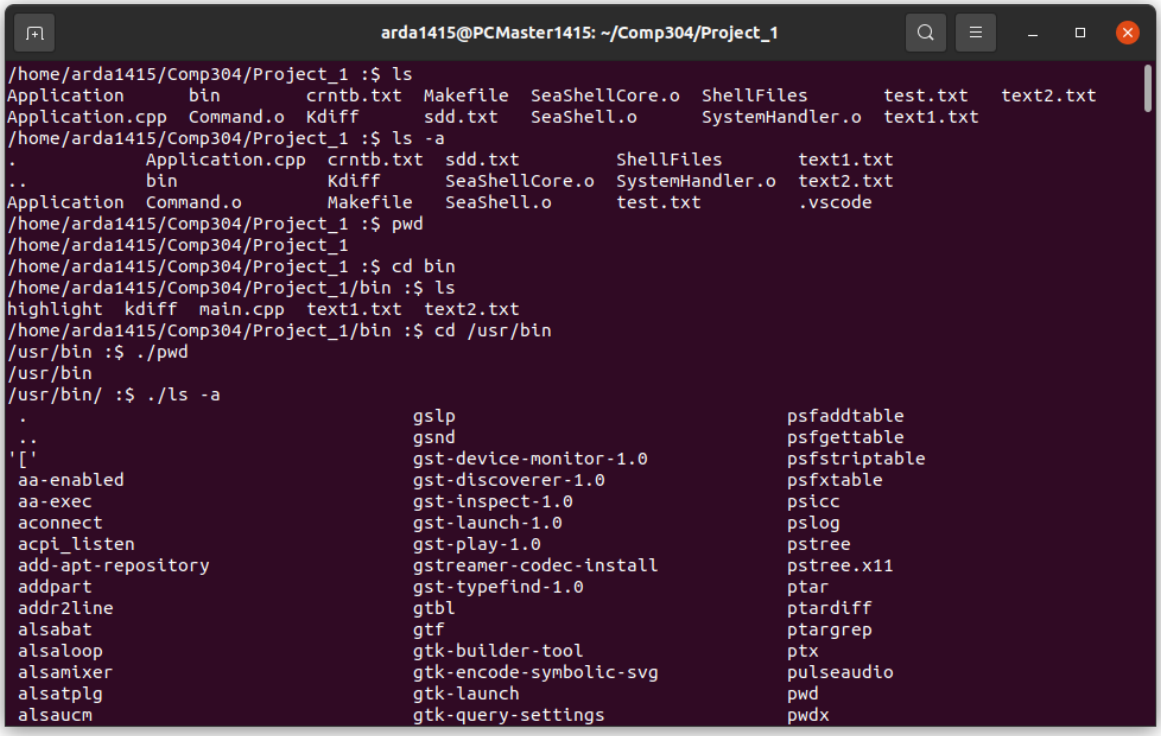
ATİLLA ÖZBEK – CAN ARDA OKÇUOĞLU

KOÇ UNIVERSITY

PART-1)

In this section, we aimed to implement the skeleton program which reads the next command line, parses, and separates it into distinct arguments using blanks as delimiters. We implemented the action that needs to be taken based on the command and its arguments entered in seashell. Also, we used execv() system call (instead of execvp()) to execute common Linux programs (e.g. ls, mkdir, cp, mv, date, gcc) and user programs by the child process. Our program searches the path for the command invoked and execute accordingly. You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUTS FOR PART-1



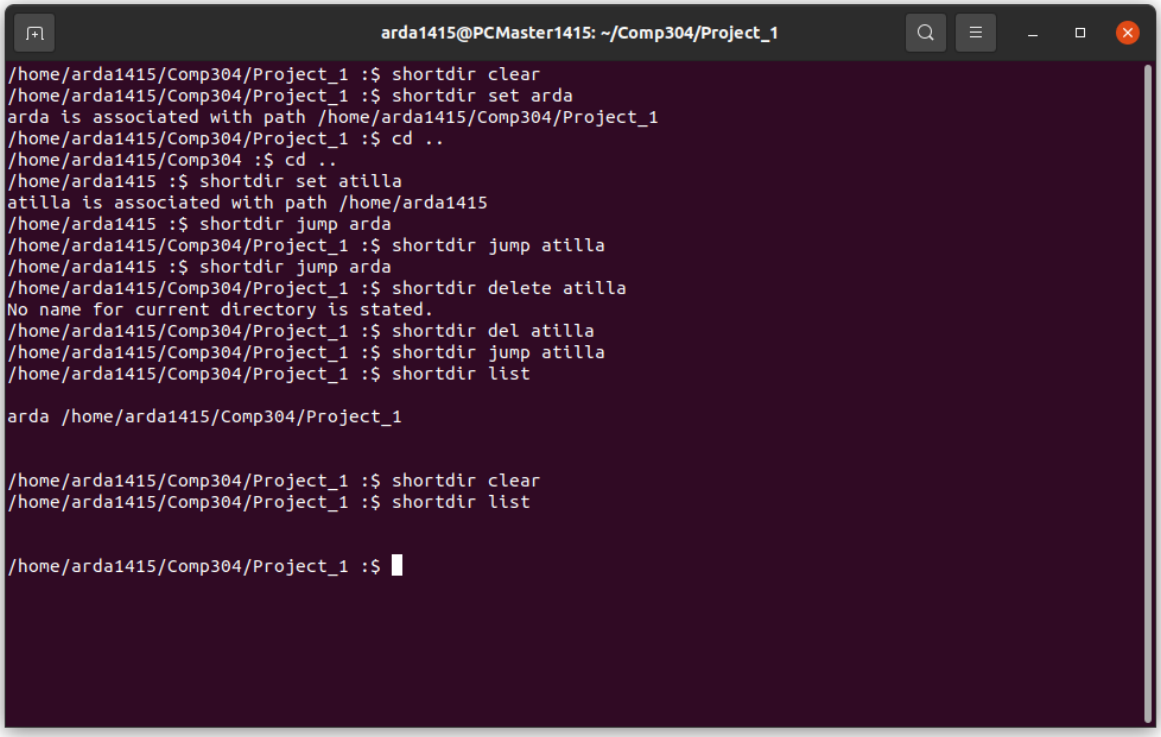
PART-2)

In this part, we implemented a new command **shortdir(),** to associate short names with the current directory. The purpose was to reach the directory with a short name instead of typing the whole path. Also, we implemented supportive options for the **shortdir()** as instructed in the document.

The SeaShell that we designed check every user input word by word. If the user input starts with "**shortdir**" SeaShell handles this command via its internal subroutine instead of executing it as an external executable.

You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUT FOR PART-2



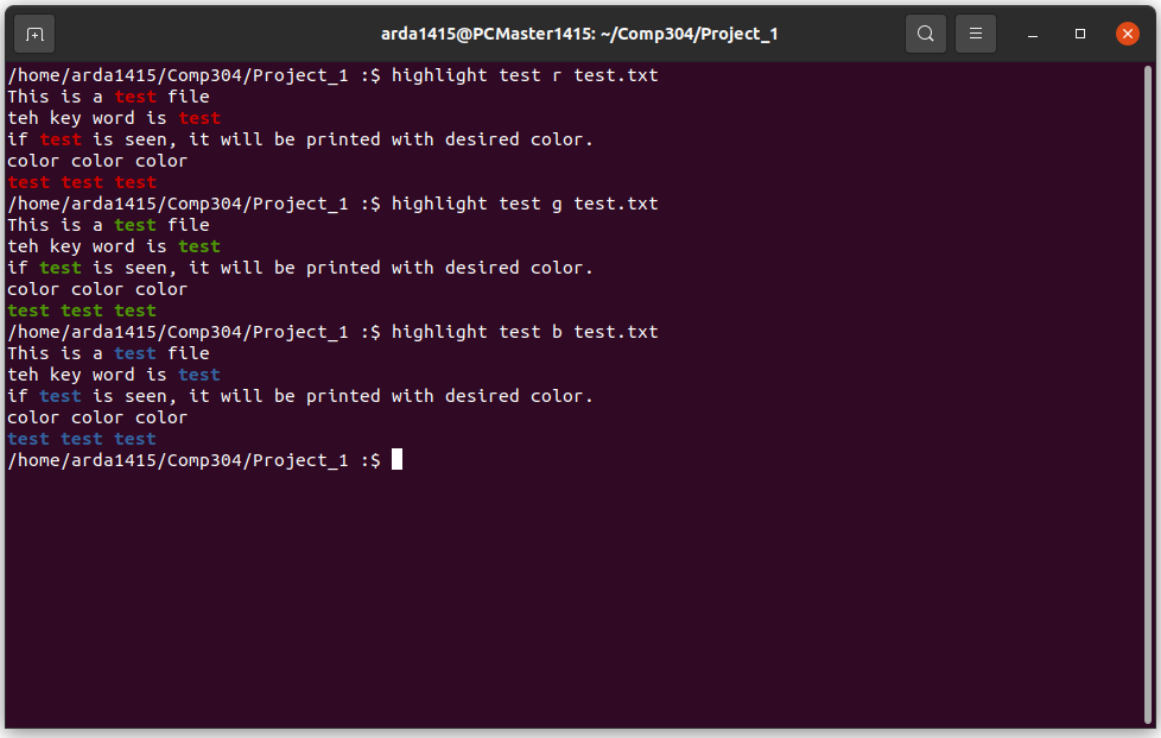
PART-3)

In this section, we implemented the highlight command that takes a word-color pair and a text file as an input. For each instance of the word appearing in the text file, the command printed the line where the word appears and highlights the word with that color.

If the user input starts with "**highlight**" SeaShell handles this command by executing it as an external executable. Unlike "**shortdir**" we thought that embedding the "**highlight**" executable into our SeaShell's bin folder would be more beneficial. In this way, our SeaShell become more modular.

You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUT FOR PART-3



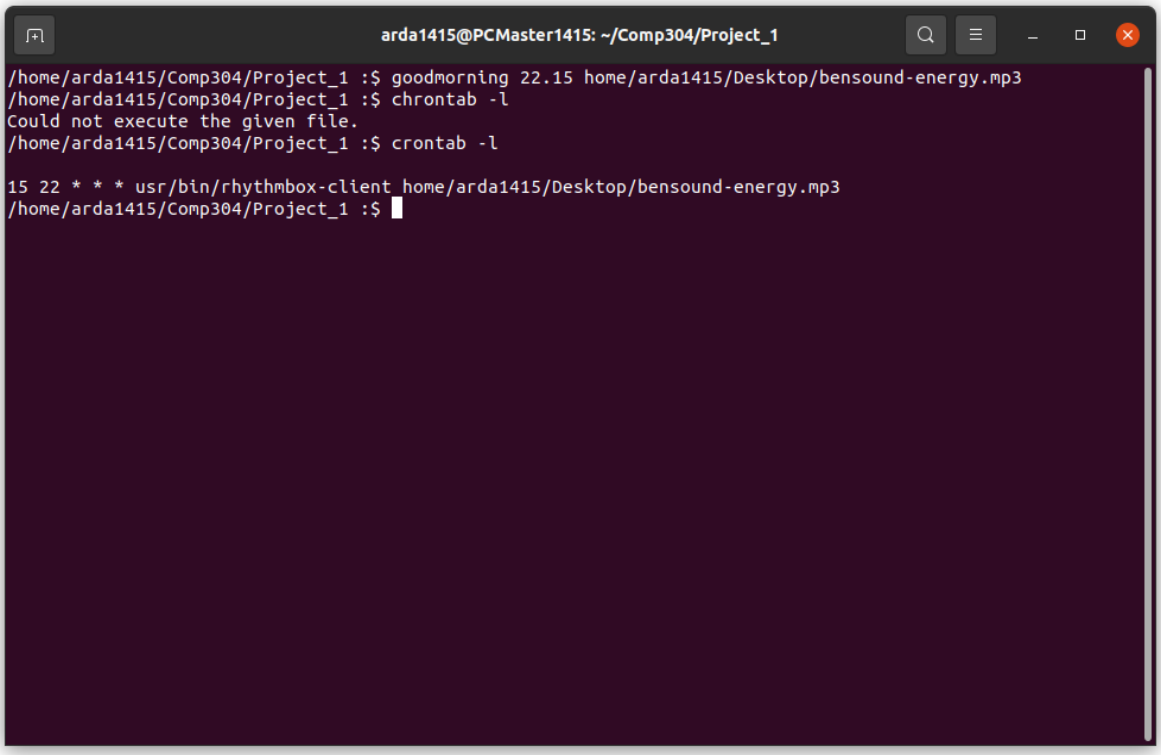
PART-4)

In this part of the project, we implemented a new seashell command, **goodmorning** which take a time and a music file as arguments and set an alarm to wake us up by playing the music using rhythmbox. To implement **goodmorning**, we used the crontab command.

Just like "**shortdir**" If the user inputs with "**goodmorning**" shell handles this command via its internal subroutine instead of executing it as an external executable. During this process, SeaShell opens a text file, converts user inputs into Crontab format and then, add them into opened text file. After that, we have registered this text file to Crontab.

You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUT FOR PART-4



PART-5)

In this part, we implemented the **kdiff** utility to compare two files in given paths. The utility operated in two modes (-a, -b):

- **-a**: The utility read the input files as text files and compared them line-by-line. If the two files were different, the utility printed the differing lines from each file and then the printed the count of differing lines. Also, the utility checked for file extensions and if they are not .txt, flagged an error. If the two files were identical, it displayed a message: “The two files are identical.”

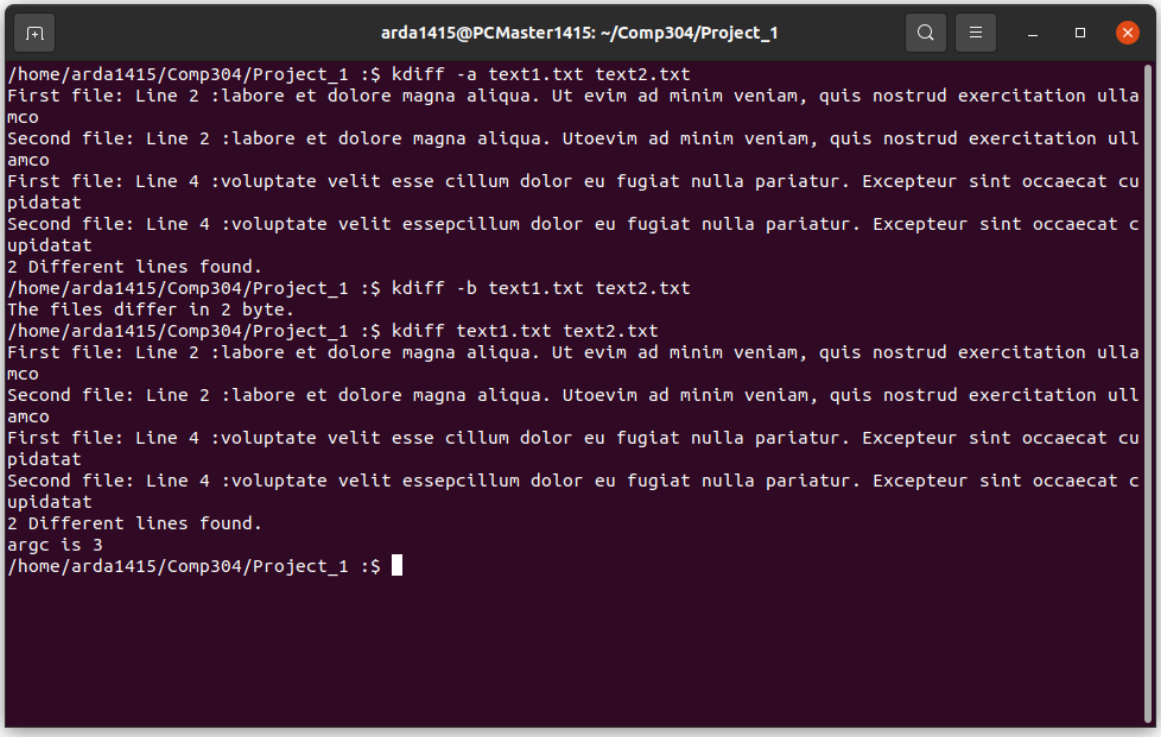
- **-b**: The utility read the input files as binary files and compared them byte-by-byte. If the two files were different, the utility printed the message "The two files are different in xyz bytes", where xyz is the number of bytes different between the files. The utility accepted files with any extension in this case. If the files were identical, the message said "The two files are identical" is displayed.

If the user input starts with “**kdiff**” SeaShell handles this command by executing it as an external executable. Just like “**highlight**” we thought that embedding the “**kdiff**” executable into our SeaShell’s bin folder would be more beneficial.

In addition, If the user does not provide either -a or -b, we assumed -a by default as instructed in the document.

You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUT FOR PART-5



PART-6)

In this part of the project, we decided to implement "**history**" command that is the basic part of the linux's bash shell. We basically store every command entered by the user into a text file.

If the user starts with "**history**" command the SeaShell's internal subroutine prints this text file index form.

You can see the basic operation of the shell in the screenshot below.

SCREENSHOT OF THE SAMPLE OUTPUT FOR PART-6