

COMP 304 - seashell - Your Custom Shell: Project 1

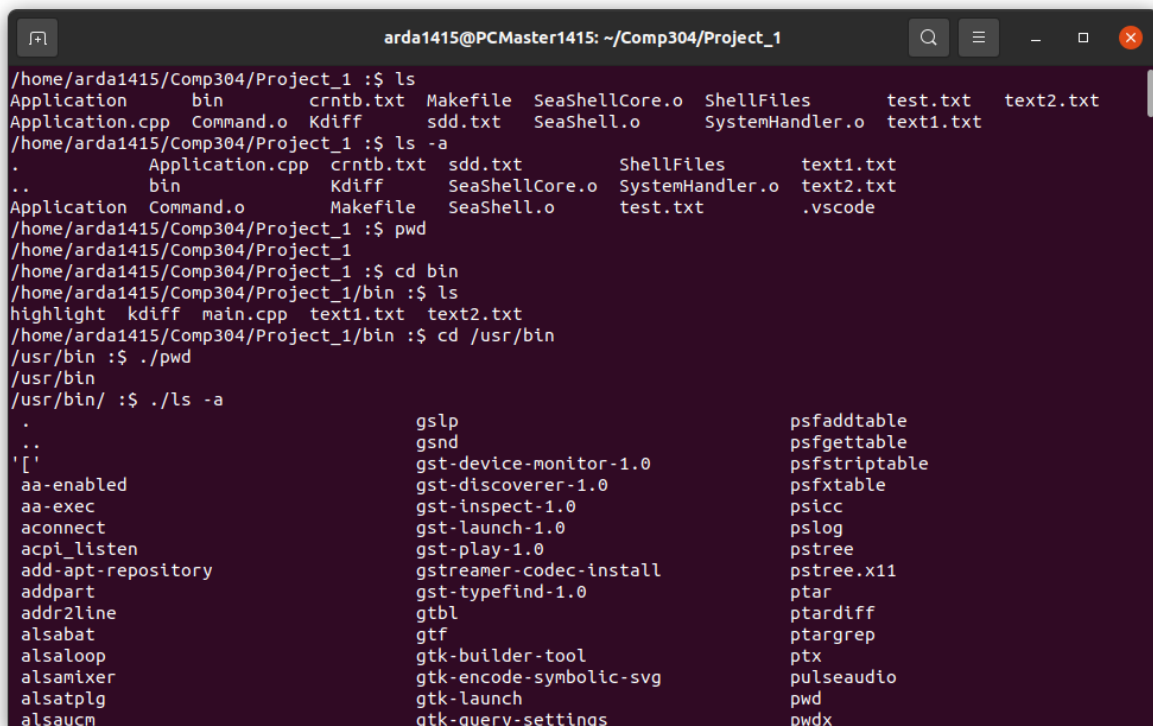
ATILLA ÖZBEK – CAN ARDA OKÇUOĞLU

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



```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ ls
Application      bin      crntb.txt  Makefile  SeaShellCore.o  ShellFiles  test.txt  text2.txt
Application.cpp  Command.o Kdiff      sdd.txt   SeaShell.o      SystemHandler.o  text1.txt
/home/arda1415/Comp304/Project_1 :$ ls -a
.      Application.cpp  crntb.txt  sdd.txt      ShellFiles  text1.txt
..     bin             Kdiff      SeaShellCore.o  SystemHandler.o  text2.txt
Application  Command.o      Makefile   SeaShell.o    test.txt     .vscode
/home/arda1415/Comp304/Project_1 :$ pwd
/home/arda1415/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ cd bin
/home/arda1415/Comp304/Project_1/bin :$ ls
highlight kdiff main.cpp text1.txt text2.txt
/home/arda1415/Comp304/Project_1/bin :$ cd /usr/bin
/usr/bin :$ ./pwd
/usr/bin
/usr/bin/ :$ ./ls -a
.      gslp      psfaddtable
..     gsnd      psfgettable
[''    gst-device-monitor-1.0  psfstriptable
aa-enabled  gst-discoverer-1.0  psfxtable
aconnect   gst-inspect-1.0    psicc
acpi_listen  gst-launch-1.0     pslog
add-apt-repository  gst-play-1.0      pstree
addpart     gstreamer-codec-install  pstree.x11
addr2line   gst-typefind-1.0   ptar
alsabat     gtbl              ptardiff
alsaloop    gtf               ptargrep
alsamixer   gtk-builder-tool   ptx
alsatplg    gtk-encode-symbolic-svg  pulseaudio
alsaucm     gtk-launch         pwd
            gtk-query-settings  pwdx
```

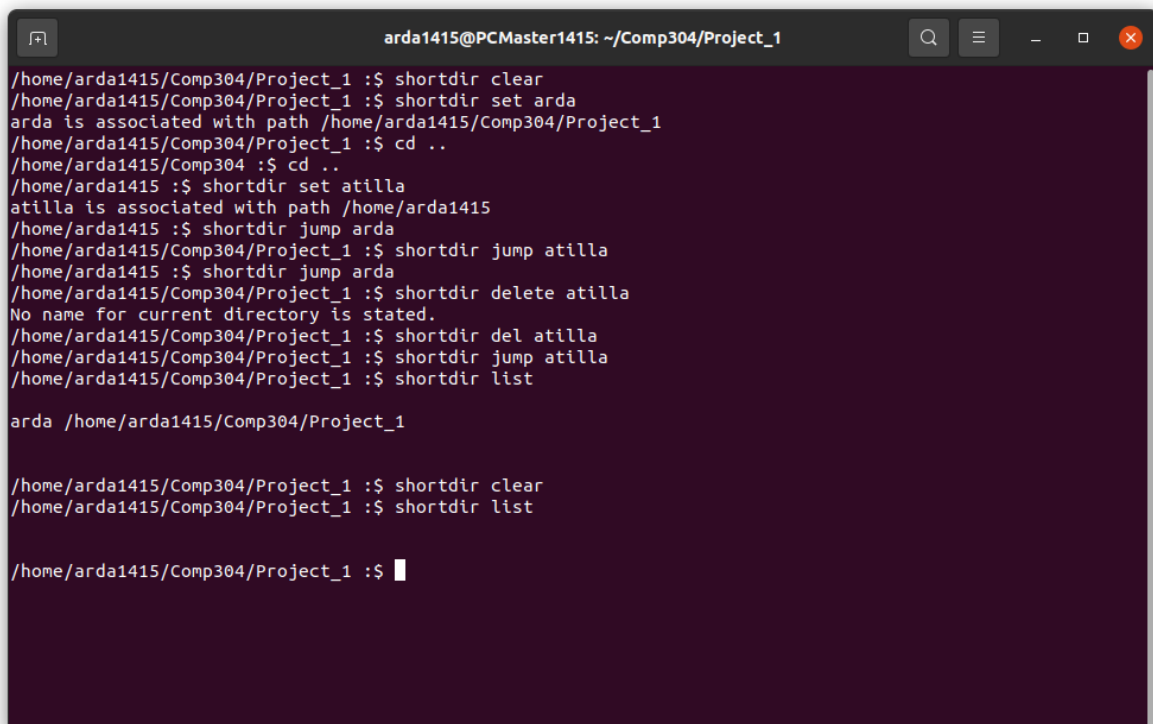
PART-2)

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



```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ shortdir clear
/home/arda1415/Comp304/Project_1 :$ shortdir set arda
arda is associated with path /home/arda1415/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ cd ..
/home/arda1415/Comp304 :$ cd ..
/home/arda1415 :$ shortdir set atilla
atilla is associated with path /home/arda1415
/home/arda1415 :$ shortdir jump arda
/home/arda1415/Comp304/Project_1 :$ shortdir jump atilla
/home/arda1415 :$ shortdir jump arda
/home/arda1415/Comp304/Project_1 :$ shortdir delete atilla
No name for current directory is stated.
/home/arda1415/Comp304/Project_1 :$ shortdir del atilla
/home/arda1415/Comp304/Project_1 :$ shortdir jump atilla
/home/arda1415/Comp304/Project_1 :$ shortdir list

arda /home/arda1415/Comp304/Project_1

/home/arda1415/Comp304/Project_1 :$ shortdir clear
/home/arda1415/Comp304/Project_1 :$ shortdir list

/home/arda1415/Comp304/Project_1 :$
```

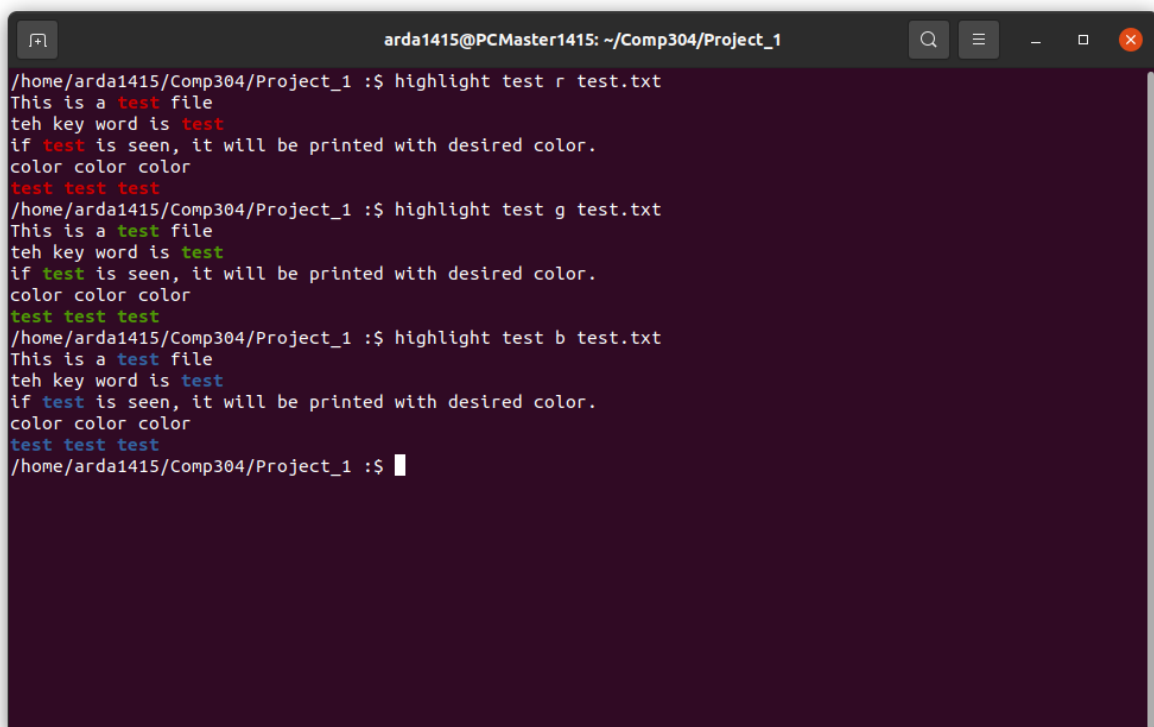
PART-3)

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



```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ highlight test r test.txt
This is a test file
teh key word is test
if test is seen, it will be printed with desired color.
color color color
test test test
/home/arda1415/Comp304/Project_1 :$ highlight test g test.txt
This is a test file
teh key word is test
if test is seen, it will be printed with desired color.
color color color
test test test
/home/arda1415/Comp304/Project_1 :$ highlight test b test.txt
This is a test file
teh key word is test
if test is seen, it will be printed with desired color.
color color color
test test test
/home/arda1415/Comp304/Project_1 :$
```

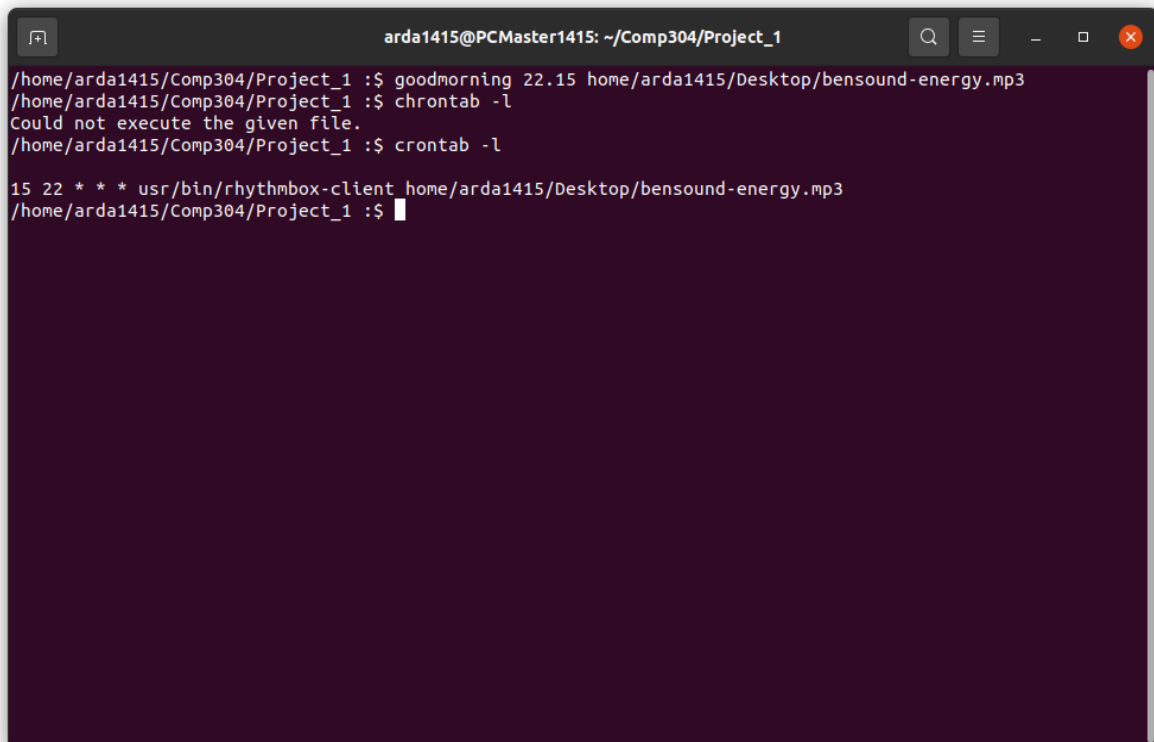
PART-4)

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



```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ goodmorning 22.15 home/arda1415/Desktop/bensound-energy.mp3
/home/arda1415/Comp304/Project_1 :$ crontab -l
Could not execute the given file.
/home/arda1415/Comp304/Project_1 :$ crontab -l
15 22 * * * usr/bin/rhythmbox-client home/arda1415/Desktop/bensound-energy.mp3
/home/arda1415/Comp304/Project_1 :$
```

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):

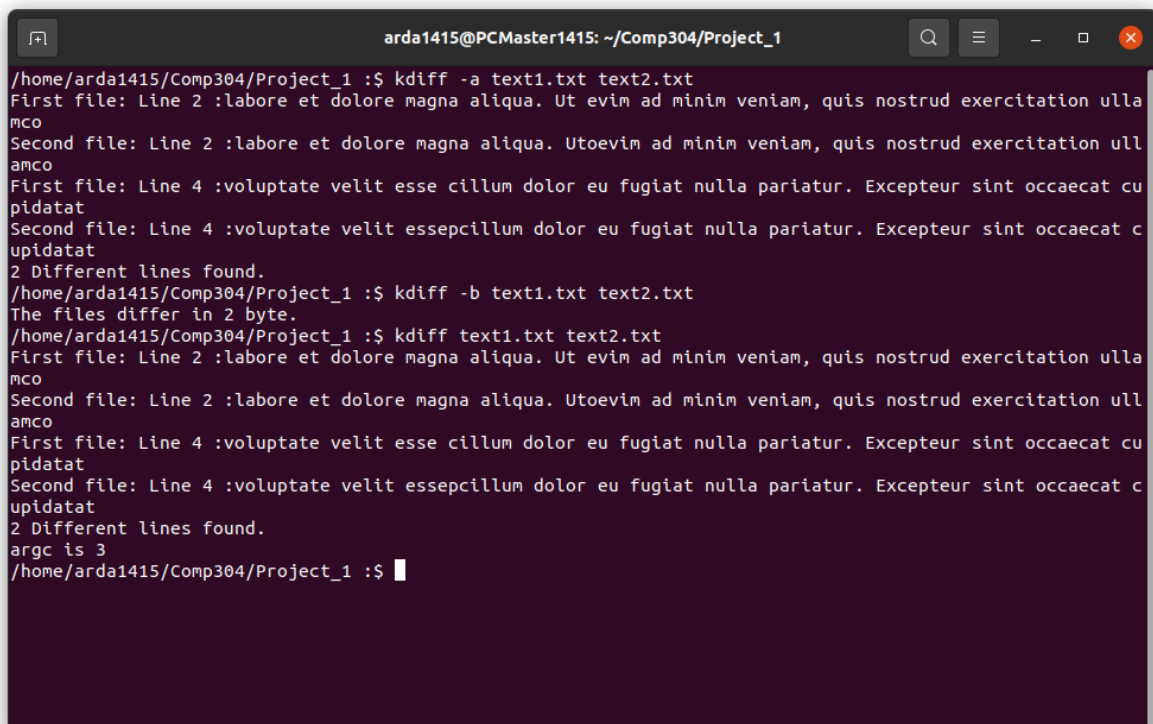
- **-a**: The utility read the input files as text files and compared them line-by-line. If the two files were different, the utility printed the differing lines from each file and then the printed the count of differing lines. Also, the utility checked for file extensions and if they are not .txt, flagged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



```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ kdiff -a text1.txt text2.txt
First file: Line 2 :labore et dolore magna aliqua. Ut evim ad minim veniam, quis nostrud exercitation ulla
mco
Second file: Line 2 :labore et dolore magna aliqua. Utoevim ad minim veniam, quis nostrud exercitation ull
amco
First file: Line 4 :voluptate velit esse cillum dolor eu fugiat nulla pariatur. Excepteur sint occaecat cu
pidatat
Second file: Line 4 :voluptate velit essepcillum dolor eu fugiat nulla pariatur. Excepteur sint occaecat c
upidatat
2 Different lines found.
/home/arda1415/Comp304/Project_1 :$ kdiff -b text1.txt text2.txt
The files differ in 2 byte.
/home/arda1415/Comp304/Project_1 :$ kdiff text1.txt text2.txt
First file: Line 2 :labore et dolore magna aliqua. Ut evim ad minim veniam, quis nostrud exercitation ulla
mco
Second file: Line 2 :labore et dolore magna aliqua. Utoevim ad minim veniam, quis nostrud exercitation ull
amco
First file: Line 4 :voluptate velit esse cillum dolor eu fugiat nulla pariatur. Excepteur sint occaecat cu
pidatat
Second file: Line 4 :voluptate velit essepcillum dolor eu fugiat nulla pariatur. Excepteur sint occaecat c
upidatat
2 Different lines found.
argc is 3
/home/arda1415/Comp304/Project_1 :$
```

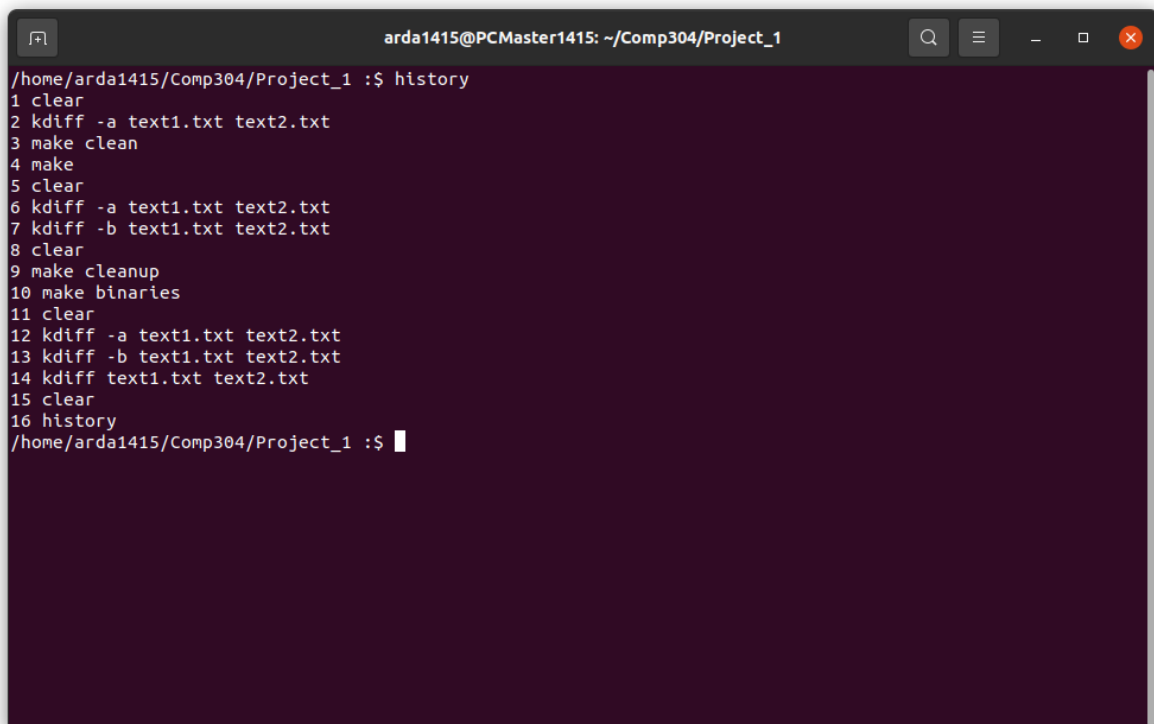
PART-6)

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

A screenshot of a terminal window with a dark purple background. The window title bar shows the user 'arda1415@PCMaster1415' and the directory '~/Comp304/Project_1'. The terminal displays the output of the 'history' command, which lists 16 commands in a numbered format. The commands are: 1 clear, 2 kdiff -a text1.txt text2.txt, 3 make clean, 4 make, 5 clear, 6 kdiff -a text1.txt text2.txt, 7 kdiff -b text1.txt text2.txt, 8 clear, 9 make cleanup, 10 make binaries, 11 clear, 12 kdiff -a text1.txt text2.txt, 13 kdiff -b text1.txt text2.txt, 14 kdiff text1.txt text2.txt, 15 clear, and 16 history. The prompt '/home/arda1415/Comp304/Project_1 :\$' is visible at the bottom of the terminal.

```
arda1415@PCMaster1415: ~/Comp304/Project_1
/home/arda1415/Comp304/Project_1 :$ history
1 clear
2 kdiff -a text1.txt text2.txt
3 make clean
4 make
5 clear
6 kdiff -a text1.txt text2.txt
7 kdiff -b text1.txt text2.txt
8 clear
9 make cleanup
10 make binaries
11 clear
12 kdiff -a text1.txt text2.txt
13 kdiff -b text1.txt text2.txt
14 kdiff text1.txt text2.txt
15 clear
16 history
/home/arda1415/Comp304/Project_1 :$
```