KOÇ UNIVERSITY

SPRING 2021

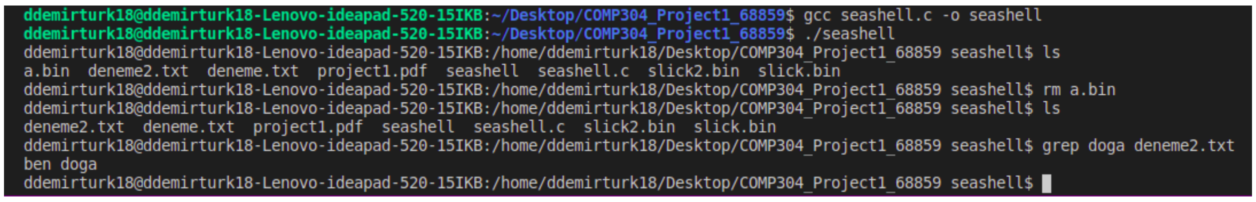
COMP 304

PROJECT 1

Group Members: Doğa Demirtürk
İrem Şahin

PART 1

In the first part of the project, we used the “execv” command instead of the “execvp” command to run the basic commands of Linux. The difference is that execvp finds the path itself but with execv we need to specify the path. To accomplish this, we take the environment variable PATH and tokenize it to find possible paths. Then by adding the name of the command at the end of the tokens we find the right path to execute the command.

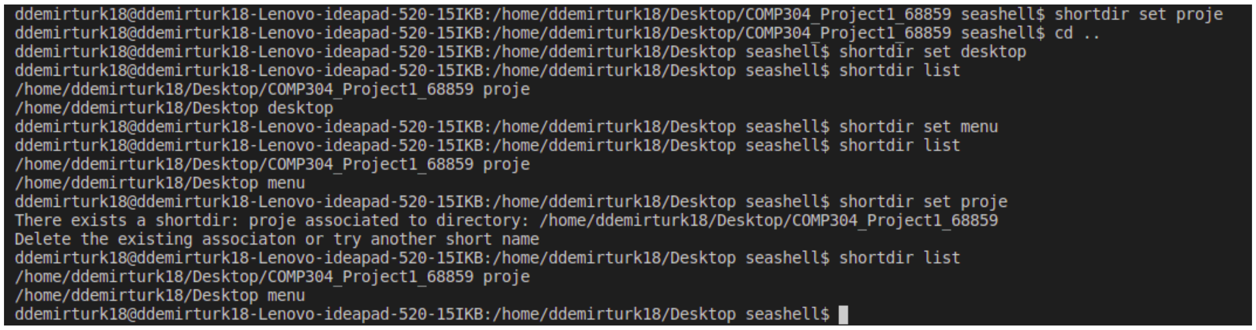


```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ gcc seashell.c -o seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ ./seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ ls
a.bin  deneme2.txt  deneme.txt  project1.pdf  seashell  seashell.c  slick2.bin  slick.bin
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ rm a.bin
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ ls
deneme2.txt  deneme.txt  project1.pdf  seashell  seashell.c  slick2.bin  slick.bin
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ grep doga deneme2.txt
ben doga
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ 
```

PART 2

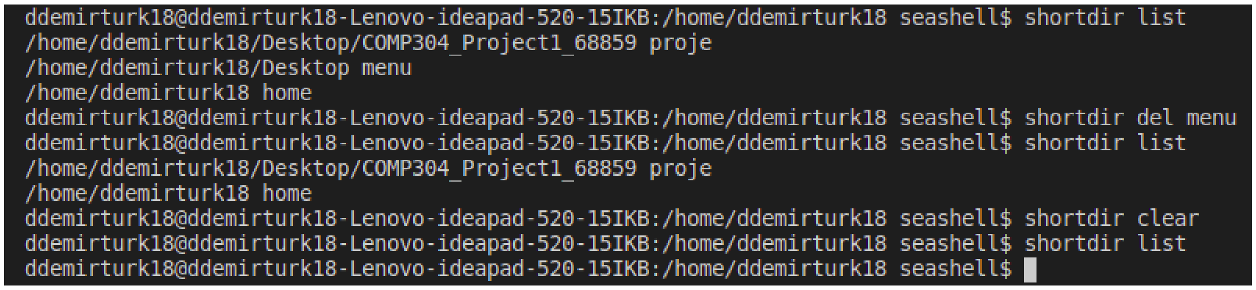
In this part, we implemented a new command called “shortdir” for our shell. The aim of the command is to create short names for directories and jump between them using their short names instead of usings “cd” command where we have to write the whole path. There are several functionalities of the command: set, jump, del, list, clear.

We can set a new short name for a directory as shown in the picture below. When we set a short name for a directory, we keep them in a file which we choose to put in the home of the system. By keeping them in a file, we can use them in every run of our shell. We can see the list of the current associations with the “list” command. If we enter the same short name for different directories, we get a warning message, and the new association is not created. Also, if we set another alias for the same directory, the old alias is deleted.



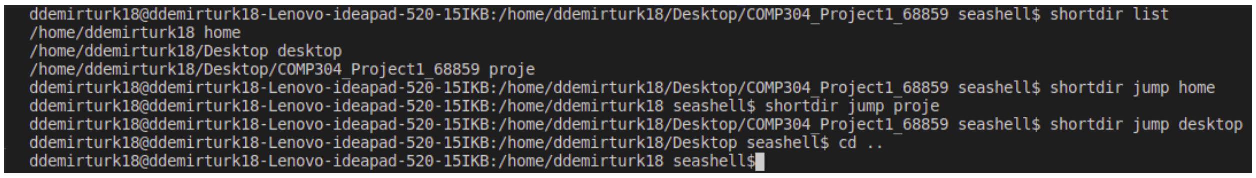
```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ shortdir set proje
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ cd ..
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir set desktop
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir list
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
/home/ddemirturk18/Desktop desktop
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir set menu
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir list
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
/home/ddemirturk18/Desktop menu
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir set proje
There exists a shortdir: proje associated to directory: /home/ddemirturk18/Desktop/COMP304_Project1_68859
Delete the existing associaton or try another short name
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ shortdir list
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
/home/ddemirturk18/Desktop menu
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$
```

We can delete an association directory-alias by using the “del” argument as shown in the picture. In addition, we can clear all the associations by using the “clear” argument.



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir list
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
/home/ddemirturk18/Desktop menu
/home/ddemirturk18 home
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir del menu
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir list
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
/home/ddemirturk18 home
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir clear
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir list
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$
```

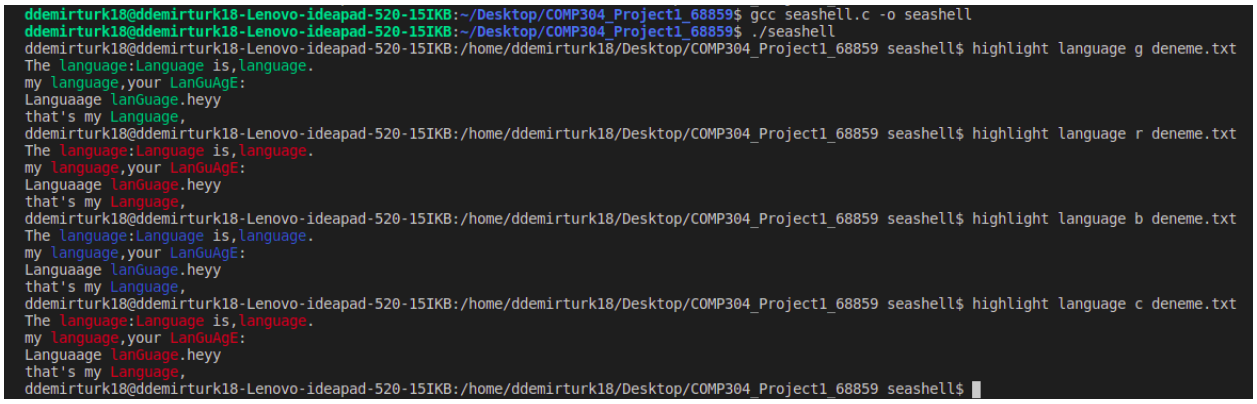
We can jump between directories as shown in the picture below. In the implementation we used a pipe to make the parent process to change the directory instead of the child process which our code runs in.



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ shortdir list
/home/ddemirturk18 home
/home/ddemirturk18/Desktop desktop
/home/ddemirturk18/Desktop/COMP304_Project1_68859 proje
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ shortdir jump home
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$ shortdir jump proje
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ shortdir jump desktop
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop seashell$ cd ..
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18 seashell$
```

PART 3

In this part, we implemented the **highlight** command. If we realize the argument name is “highlight”, we execute this command. First, we check whether there are any missing arguments by checking if the arguments count is equal to 5, if not, it exits. It needs to be 5 since we get a word, a color and a file for this command and there are extra arguments: command name at the beginning and NULL at the end. If there are enough arguments but the given text file does not exist or returns NULL, it exits again. After these checks, our code opens the file, reads lines one by one and divides the words using space, comma, full stop, semicolon, colon and new line. After this, it compares the lowercase versions of the words (since it needs to be case insensitive), and if a match is found it highlights the word using ANSI color codes. For simplicity, we used red if something other than r-g-b is used for the second argument. After printing color codes, our word, and punctuation that was deleted by strtok, it moves on to the next word.



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ gcc seashell.c -o seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ ./seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ highlight language g deneme.txt
The language:Language is,language.
my language,your LanGuAgE:
Languaage lanGuage.heyy
that's my Language,
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ highlight language r deneme.txt
The language:Language is,language.
my language,your LanGuAgE:
Languaage lanGuage.heyy
that's my Language,
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ highlight language b deneme.txt
The language:Language is,language.
my language,your LanGuAgE:
Languaage lanGuage.heyy
that's my Language,
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ highlight language c deneme.txt
The language:Language is,language.
my language,your LanGuAgE:
Languaage lanGuage.heyy
that's my Language,
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ 
```

Acknowledgements: Pointer arithmetic for reprinting the deleted delimiters are borrowed from: https://stackoverflow.com/questions/12460264/c-determining-which-delimiter-used-strtok

PART 4

In this part, we implemented the “goodMorning” command. We first started by checking whether enough arguments were given. After this, we also checked whether the given time is in the correct format. Next, we got our home directory from getenv to save our text file that was going to save the command to be executed. After getting the hour, minute and music file directories, we wrote them to the file we opened in the following form:

*minute hour * * * XDG_RUNTIME_DIR=/run/user/$(id -u) DISPLAY=:0.0 /usr/bin/rhythmbox-client --play music_dir*

minute hour * * * is for the crontab syntax, which requires the minute information before hour.

XDG_RUNTIME_DIR=/run/user/$(id -u) is to set environment variables and make rhythmbox give the desired sound.

DISPLAY=:0.0 is to make our sound start from 0.0, and this was borrowed from our TA's answer from the discussion board.

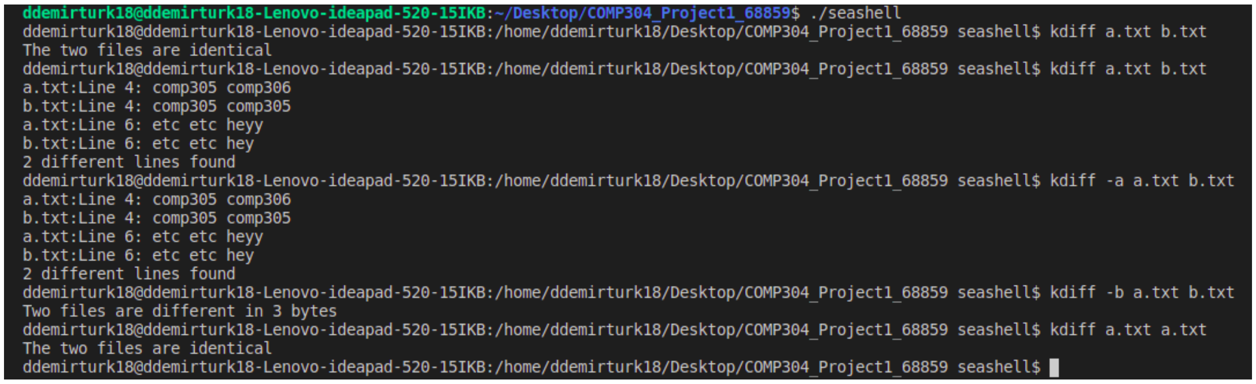
/usr/bin/rhythmbox-client --play music_dir is to make rhythmbox play the sound in the given directory.

After these, we give the system “crontab filename”, where filename involves the command we have written, and crontab schedules the given job for the given hour and minute.

PART 5

In this part, we implemented the “kdiff” command. For simplicity, we wrote a helper function named kdiff for this part. Program starts by checking the arguments, and calls the kdiff if the arguments are given in one of the three accepted formats. If not, it exits from the program.

In the kdiff function, we take the kdiff mode, first file’s and second file’s name. According to the mode, we open the files either in normal reading mode or binary reading mode. After a file null check, we start our comparison of the files. For the first mode, we get lines from each file and strcmp them to see whether we got an exact match. If they are not equal, we print each line from the both files. If we get an exact match for each line, we print a line indicating they are equal files, if not, we show how many lines were different. Similar thing is done for second mode, but this time instead of reading lines, we read bytes as integers and compare them to see whether they are equal. In the end, we print how many bytes were different for the second mode.



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ ./seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ kdiff a.txt b.txt
The two files are identical
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ kdiff a.txt b.txt
a.txt:Line 4: comp305 comp306
b.txt:Line 4: comp305 comp305
a.txt:Line 6: etc etc heyy
b.txt:Line 6: etc etc hey
2 different lines found
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ kdiff -a a.txt b.txt
a.txt:Line 4: comp305 comp306
b.txt:Line 4: comp305 comp305
a.txt:Line 6: etc etc heyy
b.txt:Line 6: etc etc hey
2 different lines found
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ kdiff -b a.txt b.txt
Two files are different in 3 bytes
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ kdiff a.txt a.txt
The two files are identical
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$
```

PART 6

In this part, we implemented a command of our own, “unique”. This command takes 2 arguments, a mode and a text file containing a list of words, and deletes the duplicate words in that file.

unique <-l | -f> filename.txt

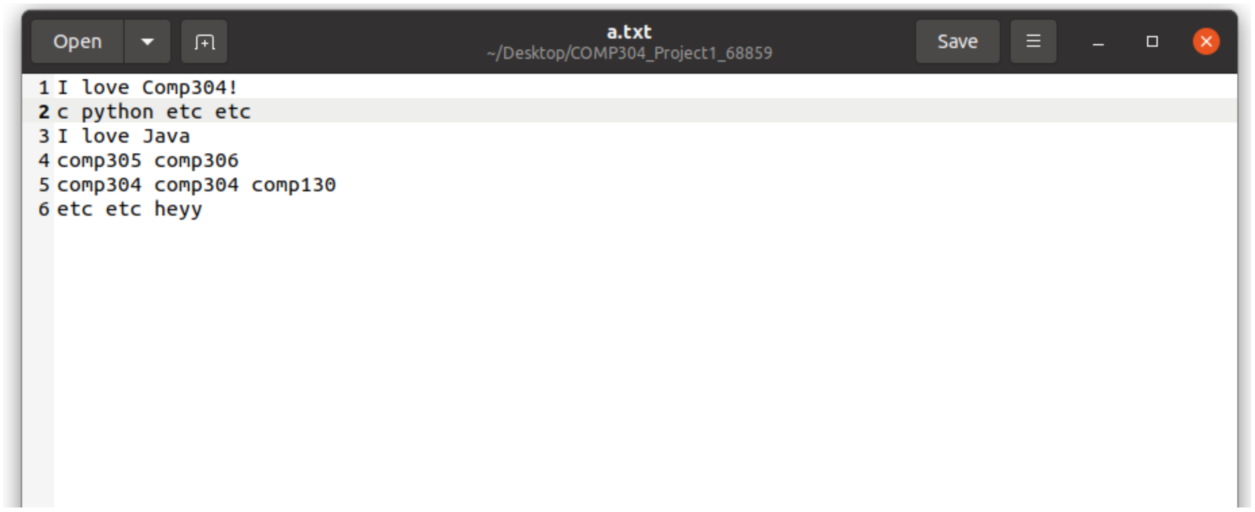
There are two modes for that command: if -l is used, duplicate words are controlled within a line, meaning a word can be used multiple times as long as they are in different lines. For example, for a text file

Istanbul Moscow Budapest Istanbul
Moscow Budapest Moscow Ankara

,our command with mode -l returns

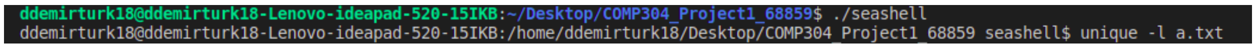
Istanbul Moscow Budapest
Moscow Budapest Ankara

It deletes the duplicate words within line only.

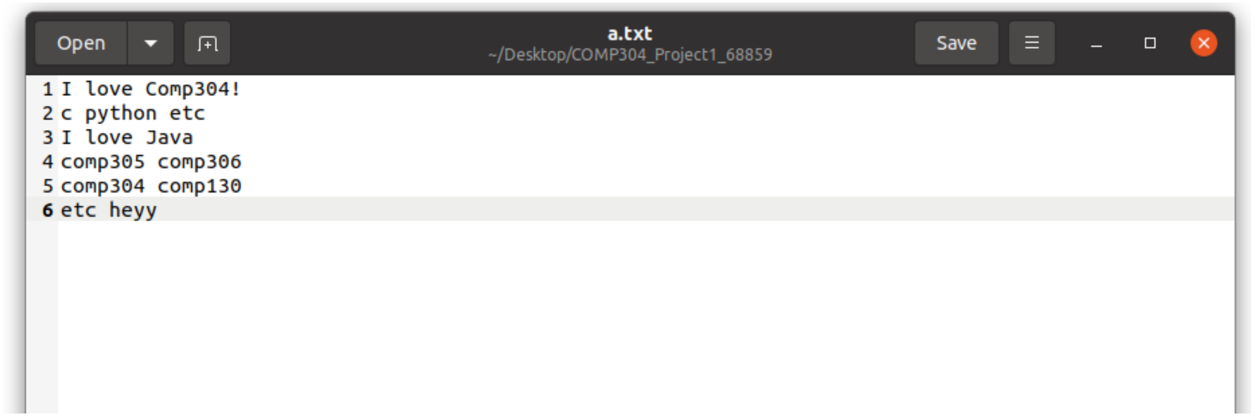


```
I love Comp304!
c python etc etc
I love Java
comp305 comp306
comp304 comp304 comp130
etc etc heyy
```

Initial file



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ ./seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ unique -l a.txt
```

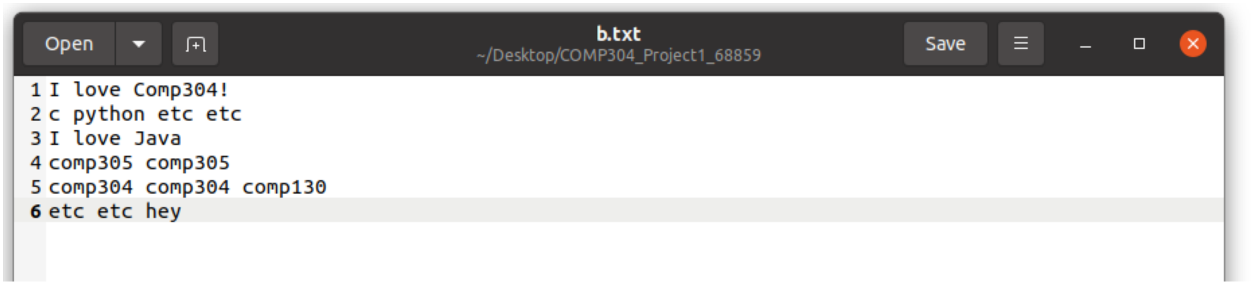


```
I love Comp304!
c python etc
I love Java
comp305 comp306
comp304 comp130
etc heyy
```

File after running command unique with mode -l

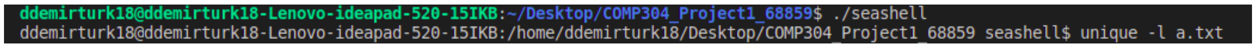
In mode -f, words are compared to every other word in the file, not only within line. Our command with mode -f for the same file returns

Istanbul Moscow Budapest
Ankara

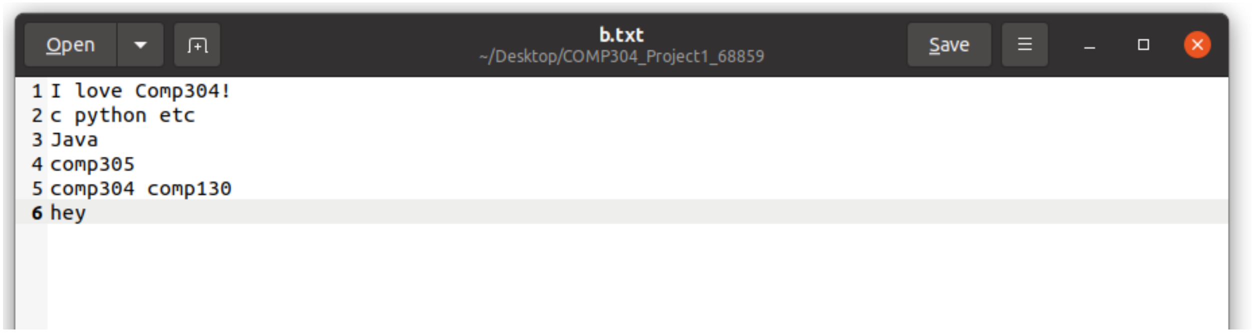


```
I love Comp304!
c python etc etc
I love Java
comp305 comp305
comp304 comp304 comp130
etc etc hey
```

Initial file



```
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:~/Desktop/COMP304_Project1_68859$ ./seashell
ddemirturk18@ddemirturk18-Lenovo-ideapad-520-15IKB:/home/ddemirturk18/Desktop/COMP304_Project1_68859 seashell$ unique -l a.txt
```



```
I love Comp304!
c python etc
Java
comp305
comp304 comp130
hey
```

File after running command unique with mode -f

For this command’s implementation, we used 2 temporary files. One to hold the unique words(unique.txt) for each line or file(depending on the mode), and the other to hold the current situation of the original file(temp.txt). By writing the unique words to a file and comparing our words from the original file to the word in the unique words file, we have found the duplicates and didn’t write them to temp.txt. We cleared the unique.txt for every line if -l mode, delete duplicates within line, is used.