Hanish Patel

HW 5

CS 497-01

Report for HW 5

**Abstract**

To script and program in bash. To understand the syntax and errors in the bash programming. To understand that bash has flaws and does not do a well job to prevent errors. To know the requirements of programming in bash such as adding shebang bin bash. To be able to use ceasar cipher and rot13 and tr to translate and encrypt/decrypt messages.

**Introduction**

We will use this command to run our program:

Bash Rot13Ceasar.sh

We will use the following options:

1 for choice 1

2 for choice 2

3 for choice 3

F for file

To store our function output

Result=$(name parameters)

To print

Echo $result

To define functions and local

Function name local name

To open a file and write

Cat

To create a bash file

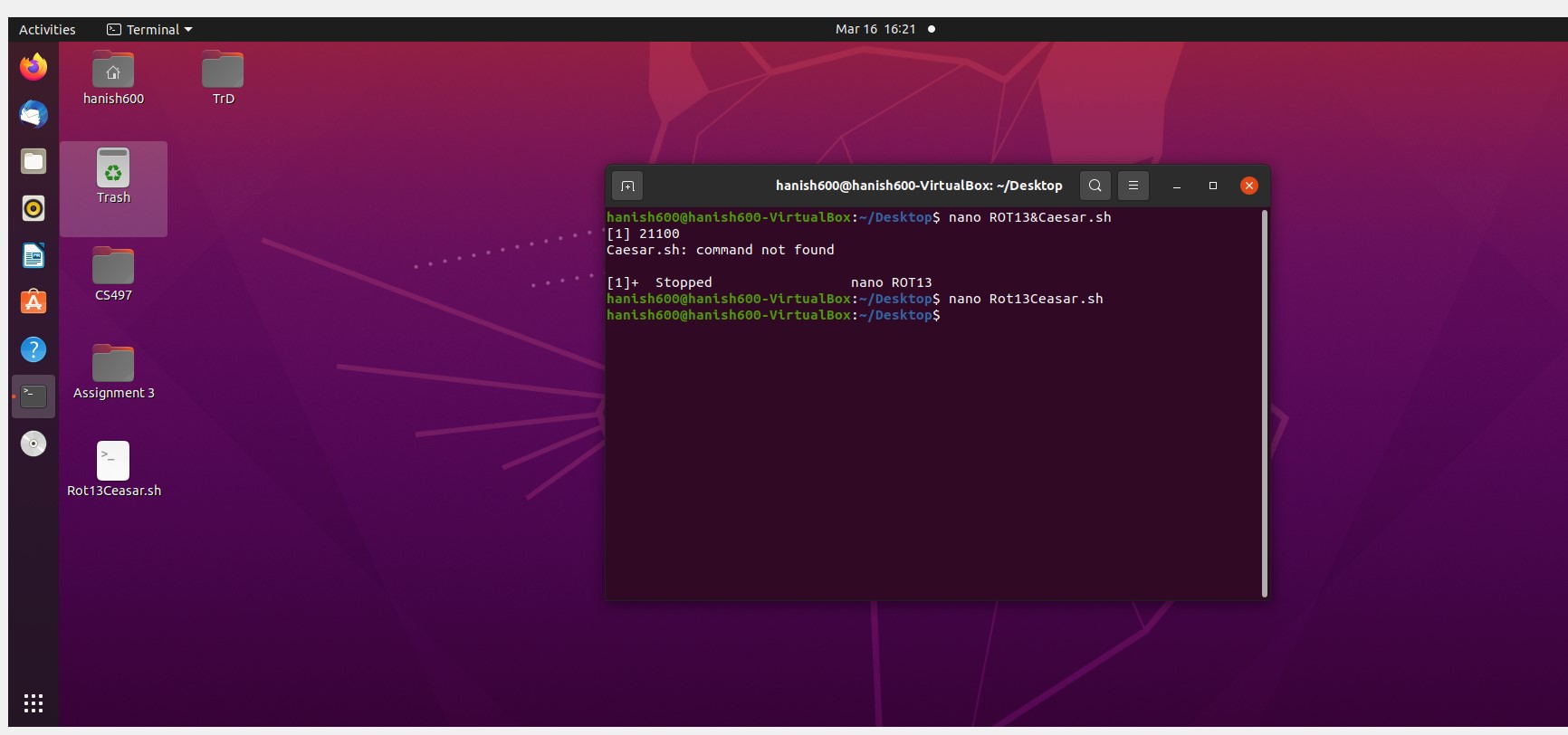
Nano filename

Use tr to translate

**Summary of results**

Start by creating a bash program. Use nano bashname.sh

Also add in commands to run bash at top and print a basic hello world message.

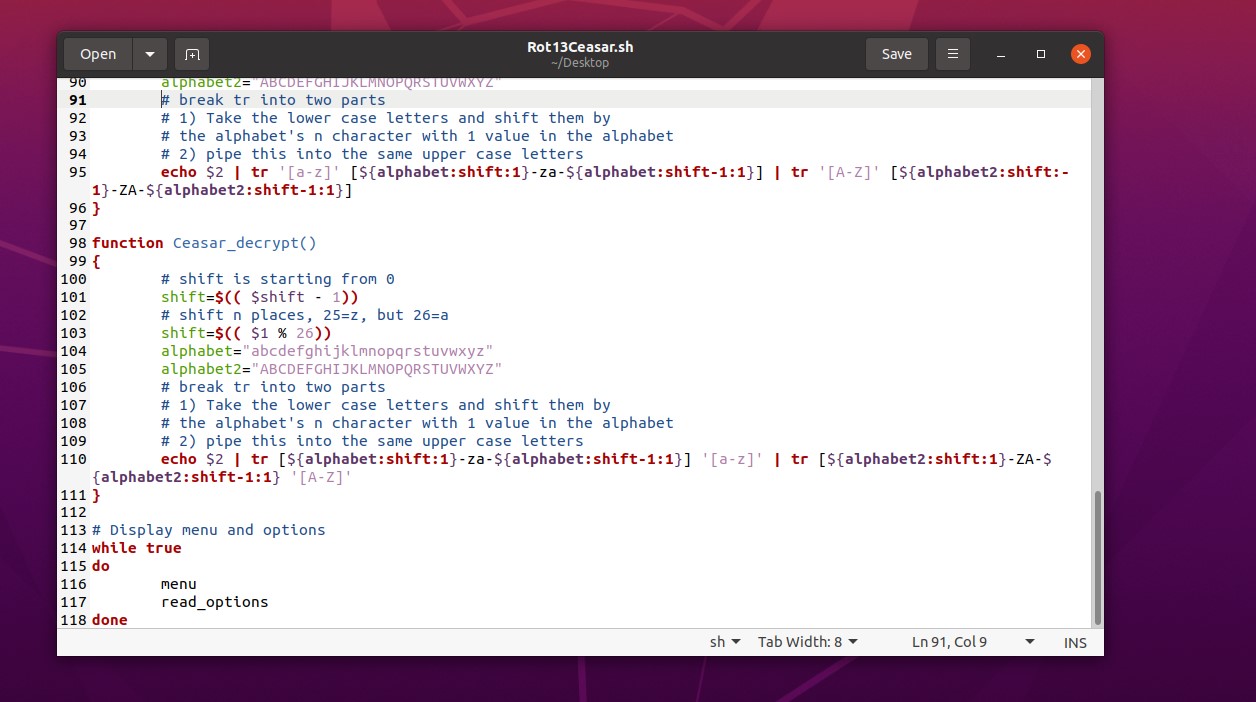
Text

Description automatically generated

We get presented with our message



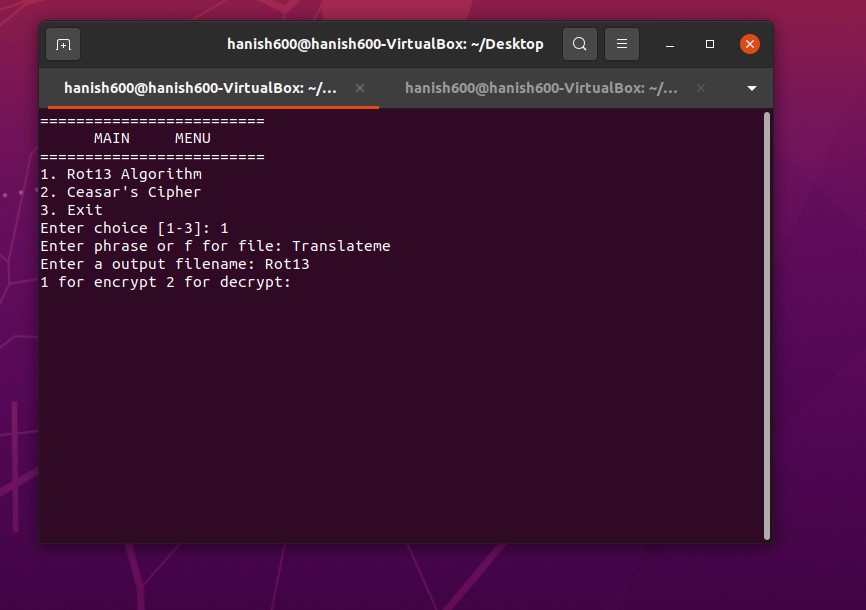
Next use nano and create basic functions and calls and add necessary loops, variables, etc



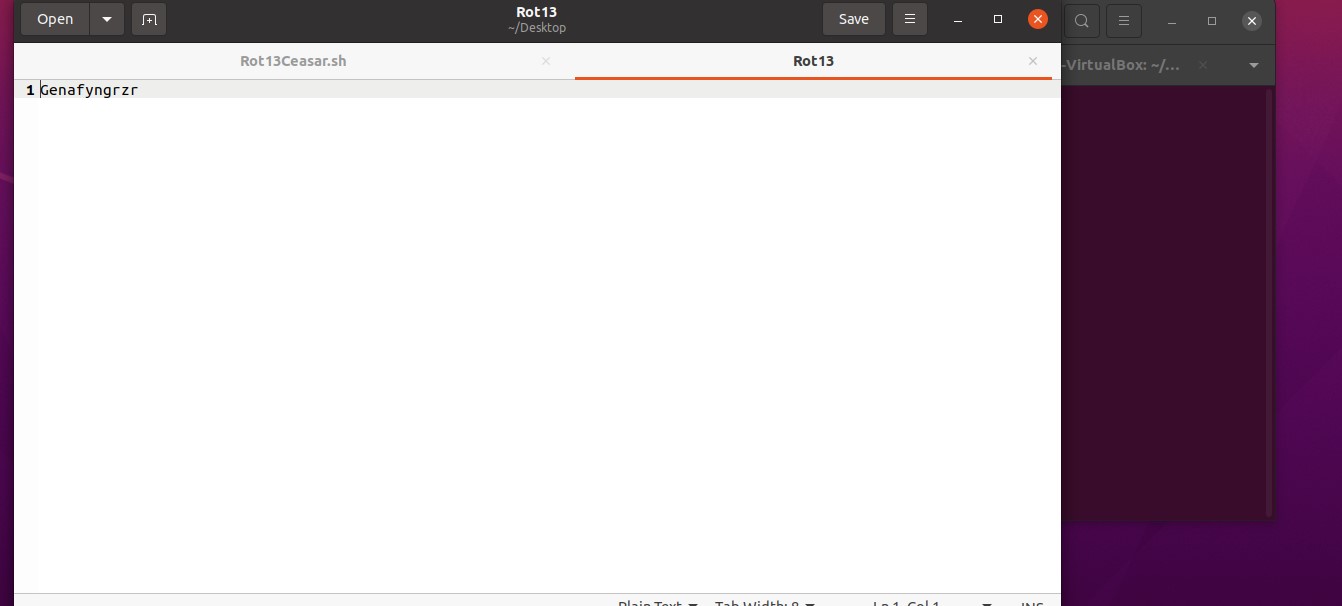
After playing around, we can use pipe | to join two translate statements since a single tr was not working properly. We use shift -1 to start from index 0 and use $(()) to modify the values. We also use it with mod 26 to restart the index. With Rot13, we instead just do one line with tr and then the 13th alphabet. We also create a menu and do some options.

We try to test the program and notice several errors that are syntactial by Bash.

First, we translate our word.

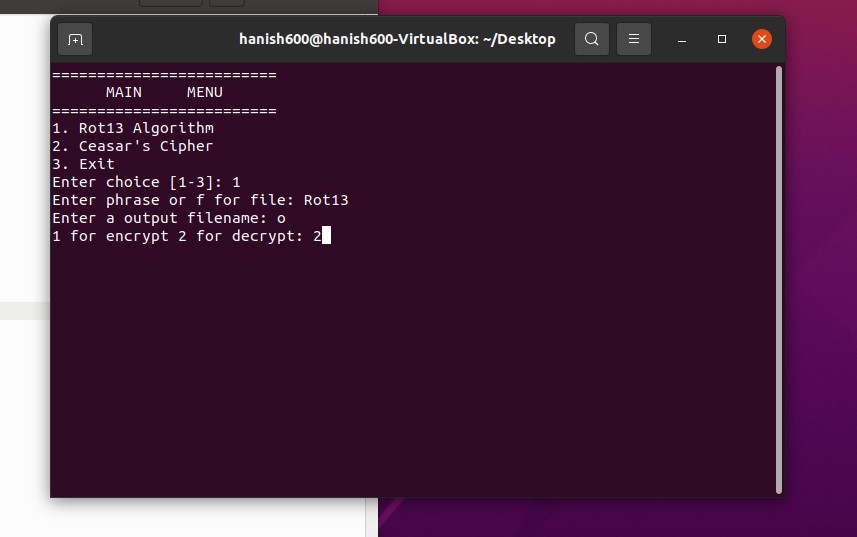


We open the file.



There was no errors here.

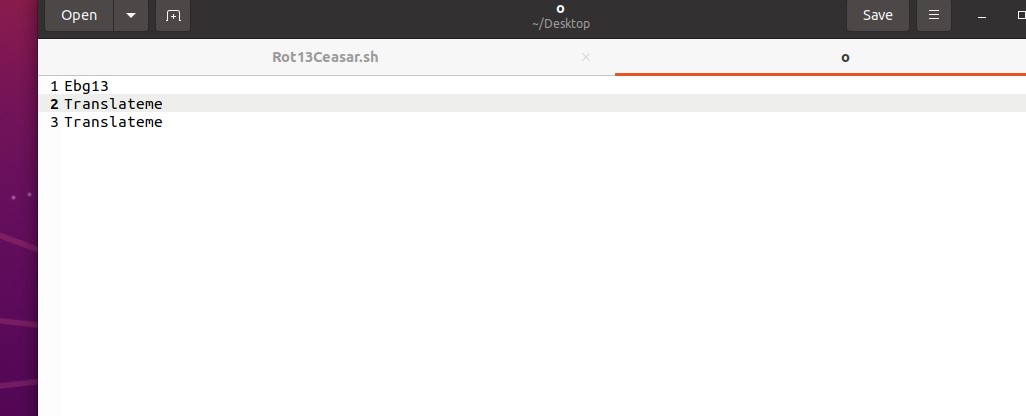
Next, we decrypt this.



We spot errors. We see that if we do f for file and we do a phrase=cat<phrase, bash does not understand what to do. It somehow thinks this is some weird phrase?

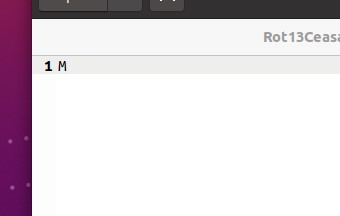


We try again, but this time we use the encoded message.

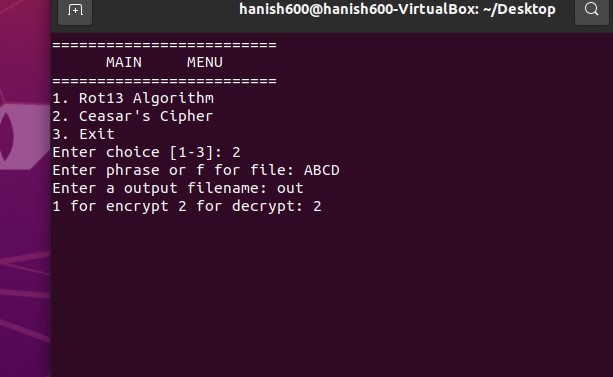


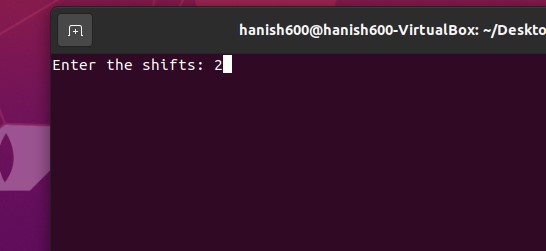
We see that this might be a error with bash since it logically works fine. Somehow it does not enter the else statement, nor understand catting the file.

We do H shifted by 5 now and we get this:

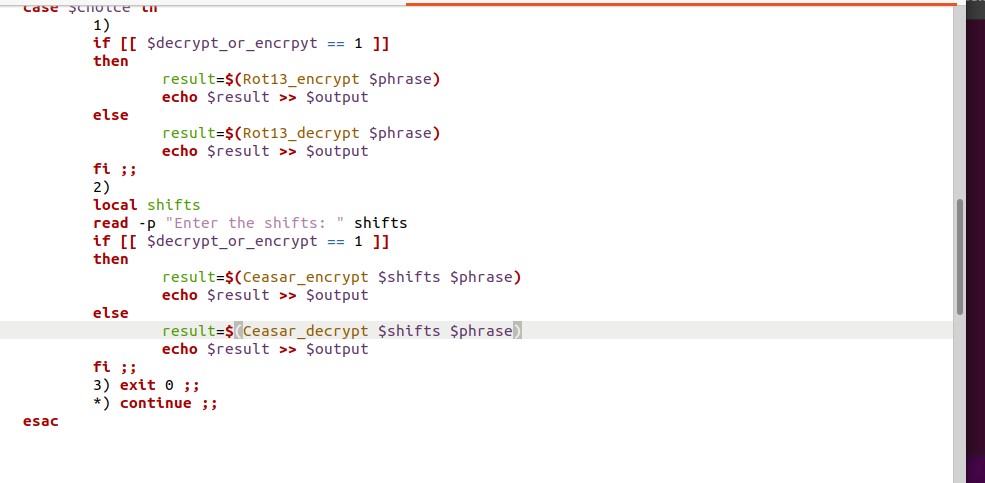


Again, we try to decrypt it and it should work. We get this again



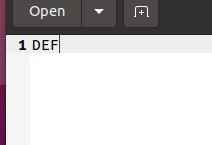


We retried it after this.



Perhaps we made an error in our logic, since we translated the shifts to the original a-z.

So we tried ABC with 3 shifts.



We now get the right output.

This one turned out to be a faulty logic error.

So we simply removed one of the functions.

We figured out there’s an error with bash with else statements and bash is unreliable with the current statements used in ROT13. We also found out a logical error and fixed it.

The statement in ROT13 was not reading or taking filename properly. We could also not figure out how to get bash to read newline.

**Conclusion**

We found out that with the current statements, ROT 13 was not functional for decrypting, but this was not a logical error since it was working with function call. Somehow the else statement or the filename was not read properly. We had put phrase= cat<$phrase, but it was not working as it should. It was throwing out garbage. We tried this without the Menu and it worked as an individual function without filename. Secondly, we found a logical error with ceasar cipher and fixed it and got it to work. This was not an issue and we used shifting and a set of words with tr to do so.

I find the bash can be unreliable due to the ROT 13 decryption error causing improper file read or not executing statements properly. This was a weird encounter since in theory it worked as an individual function without any file read. Bash also does not like new line characters. We tried using -e -r and so on as optional commands, but it refused to accept a white line character. This was another weird bug. I had tons of fun with creating both encryption and decryptions. The command line was too fast to capture every image, but I tried to get some images.