Everlyn Leon

Abstract
Writing a shell script to demonstrate encryption features, file reading and file writing, string manipulation, and a looping menu.

Introduction
Using an Ubuntu Virtual Machine to demonstrate the encryption utility using ROT13 and the Caesar cipher using a written Bash shell script. ROT13 is a letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet, and the Caesar cipher is another substitution technique where a plain text is encrypted into a ciphertext by shifting each letter a number of times, which will both be demonstrated with the written shell script.

Summary of Results

To start the shell script first, write a menu that prompts the user to choose if they would like to use the ROT13 cipher or the Caesar Cipher. This menu loops with a while loop. The choices will reference the menus for the ROT13 and Caesar Cipher.

```
#!/usr/bin/env bash

menu() {
  clear
  echo "✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿"
  echo "      M E N U"
  echo "✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿✿"
  echo "1. ROT13"
  echo "2. Caesar's Cipher"
  echo "3. Exit"
  read -p "Enter choice [ 1 - 3 ] " choice

  case $choice in
    1) rot13Menu ;;
    2) cipherMenu ;;
    3) exit 0;;
  esac
}
```

```
while true
do
  menu
done
```

ROT13

A ROT13 cipher replaces a letter with the 13th letter after it in the alphabet. In the ROT13 menu, we ask the user to choose if they would like to upload a file or type one out, which references the ‘file’ and ‘read’ ROT13 functions.

In the fileRot13 function, the filename will be read, then the ROT13 cipher will be applied using the ‘tr’ command, which is the translate function. To use ROT13, we use the following command:
tr 'A-Za-z' 'N-ZA-Mn-za-m'

Which makes it so that each character in the first set of letters gets replaced by the corresponding character of the second set.

```
rot13Menu(){
  echo "Would you like to read from a file or type a message?"
  echo "1. Read from a file"
  echo "2. Type a message"
  echo "3. Exit"
  read -p "Enter choice [ 1 - 3 ] " choice

  case $choice in
    1) fileRot13 ;;
    2) typeRot13 ;;
    3) exit 0;;
  esac
}
```

```
fileRot13(){
  echo "What is the name of the file?: "
  filename = $1
  while read line; do
  echo $line | tr '[A-Za-z]' '[N-ZA-Mn-za-m]')
  done < $filename
}

typeRot13(){
  echo -n "Type your message: "
  read;
  echo You typed ${REPLY} | tr '[A-Za-z]' '[N-ZA-Mn-za-m]')
}
```

Caesar Cipher with Shifts

For the Caesar Cipher with shifts, we would need to prompt the user to choose the number of shifts, as well as if they would like to Encrypt or Decrypt the message. First, we ask if they would like to Encrypt or Decrypt, which reference their corresponding functions. Encrypting with a Caesar Cipher simply means that we will apply the number of shifts requested to the text read, so if the number of shifts is 5, then the plaintext will be shifted 5 letters to the left, and when decrypted, it would be in the opposing direction.

```
cipherMenu(){
  echo "Would you like to Encrypt or Decrypt the message?: "
  echo "1. Encrypt"
  echo "2. Decrypt"
  echo "3. Exit"
  read -p "Enter choice [ 1 - 3 ] " choice

  case $choice in
    1) encrypt ;;
    2) decrypt ;;
    3) exit 0;;
  esac
}
```

We need to apply our own shifts to the cipher, and to do this we make variables of the uppercase and lowercase letters, then apply the shifts to them using the ‘sed’ command which is used for a substitution or finding and replacing. For the sed command, the ‘y’ is its translate command, which allows for a Caesar shift with the upper and lower case alphabet. The Caesar Cipher is then applied to the file/text by using the ‘cat’ command, which concatenates the plaintext, which then results as cipher text after the Caesar Cipher is applied.

```
encrypt(){
  echo "Would you like to read from a file or type a message?"
  echo "1. Read from a file"
  echo "2. Type a message"
  echo "3. Exit"
  read -p "Enter choice [ 1 - 3 ] " choice

  case $choice in
   1) enfileCipher ;;
   2) entypeCipher ;;
   3) exit 0;;
 esac
}

enfileCipher(){
  echo "How many shifts would you like to have?: "
  read -p shifts

 echo "What is the name of the file?: "
  filename = $1
  while read line; do
  echo $line
  done < $filename

 up = ABCDEFGHIJKLMNOPQRSTUVWXYZ
 low = abcdefghijklmnopqrstuvwxyz
 encc=$(cat $filename | sed "y/$up$low/${up:$shifts}${up::$shifts}${low:$shifts}${low::$shifts}/")
 echo ${encc}
}
```

```
entypeCipher(){
  echo "How many shifts would you like to have?: "
  read -p shifts

 echo -n "Type your message: "
  read;
  echo You typed ${REPLY}

  filename = ${REPLY}

  up = ABCDEFGHIJKLMNOPQRSTUVWXYZ
  low = abcdefghijklmnopqrstuvwxyz
  encc=$(cat $filename | sed "y/$up$low/${up:$shifts}${up::$shifts}${low:$shifts}${low::$shifts}/")
  echo ${encc}

 }
```

For decrypting, the Caesar Cipher is simply applied in the opposing direction.

```
decrypt(){
  echo "Would you like to read from a file or type a message?"
  echo "1. Read from a file"
  echo "2. Type a message"
  echo "3. Exit"
  read -p "Enter choice [ 1 - 3 ] " choice

  case $choice in
   1) defileCipher ;;
   2) detypeCipher ;;
   3) exit 0;;
  esac
}

defileCipher(){
 echo "How many shifts would you like to have?: "
 read -p shifts

 echo "What is the name of the file?: "
  filename = $1
  while read line; do
  echo $line
  done < $filename

  up = ABCDEFGHIJKLMNOPQRSTUVWXYZ
  low = abcdefghijklmnopqrstuvwxyz
  decc=$(cat $filename | sed "y/${up:$shifts}${up::$shifts}${low:$shifts}${low::$shifts}/$up$low/")
  echo ${decc}

}
```

```
detypeCipher(){
 echo "How many shifts would you like to have?: "
 read -p shifts

 echo -n "Type your message: "
  read;
  echo You typed ${REPLY}

filename = ${REPLY}

up = ABCDEFGHIJKLMNOPQRSTUVWXYZ
low = abcdefghijklmnopqrstuvwxyz
decc=$(cat $filename | sed "y/${up:$shifts}${up::$shifts}${low:$shifts}${low::$shifts}/$up$low/")
echo ${decc}

 }
```

Conclusion

A shell script was written to demonstrate the encryption and decryption abilities ROT13 and Caesar Cipher have, which for encrypting is simply shifting letters in plain text, thus creating a ciphertext, and when decrypting, the shift is simply in the opposite direction.