

Oregon Institute of Technology

CST 240 Linux Programming Lab #2: Shell, Encryption (C)

CST 240

LAB 2: Shell, Encryption (C)

1. (10 pts) If you haven't already, make a new directory named L2 that is a child of the LAB directory from the previous L1 assignment. Copy RichesIHold.txt from directory L1 into the new directory L2. Create a file named "HW2_prog.sh". Enter the following contents of the file:

```
#!/bin/bash
for filename in "$@"
do
    linecount="1"
    while IFS="\n" read line
    do
        echo "${linecount}: $line"
        linecount=$(( $linecount + 1 ))
    done < $filename
done
exit 0
```

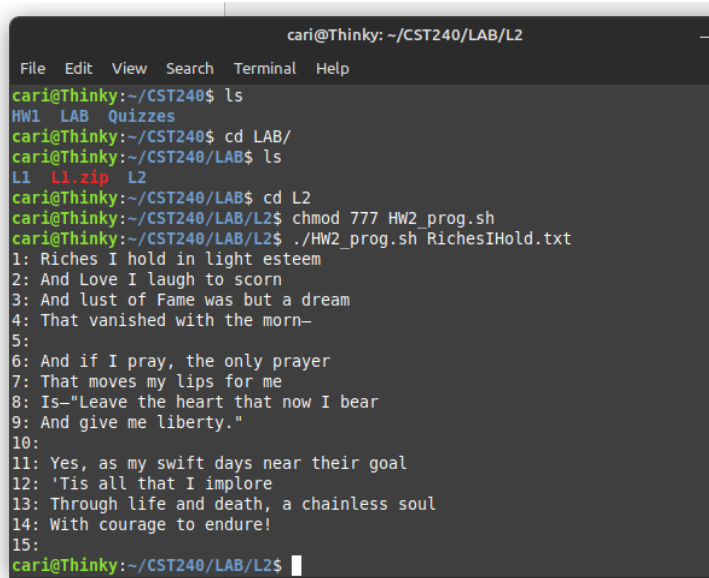
Afterward, type the command:

```
$ chmod 777 HW2_prog.sh
```

Then:

```
$ ./HW2_prog.sh RichesIHold.txt
```

Show the result you see after `$./HW2_prog.sh RichesIHold.txt`



```
cari@Thinky: ~/CST240/LAB/L2
File Edit View Search Terminal Help
cari@Thinky:~/CST240$ ls
HW1 LAB Quizzes
cari@Thinky:~/CST240$ cd LAB/
cari@Thinky:~/CST240/LAB$ ls
L1 L1.zip L2
cari@Thinky:~/CST240/LAB$ cd L2
cari@Thinky:~/CST240/LAB/L2$ chmod 777 HW2_prog.sh
cari@Thinky:~/CST240/LAB/L2$ ./HW2_prog.sh RichesIHold.txt
1: Riches I hold in light esteem
2: And Love I laugh to scorn
3: And lust of Fame was but a dream
4: That vanished with the morn-
5:
6: And if I pray, the only prayer
7: That moves my lips for me
8: Is-"Leave the heart that now I bear
9: And give me liberty."
10:
11: Yes, as my swift days near their goal
12: 'Tis all that I implore
13: Through life and death, a chainless soul
14: With courage to endure!
15:
cari@Thinky:~/CST240/LAB/L2$
```

Oregon Institute of Technology

CST 240 Linux Programming Lab #2: Shell, Encryption (C)

2. (10 pts)

Create a file named "rev_HW2_prog.sh". The content of the file is:

```
#!/bin/bash
linecount = 10
while [$linecount -gt 0 ]
do
    echo "${linecount}"
    linecount= ${linecount-1}
//done
exit 0
```

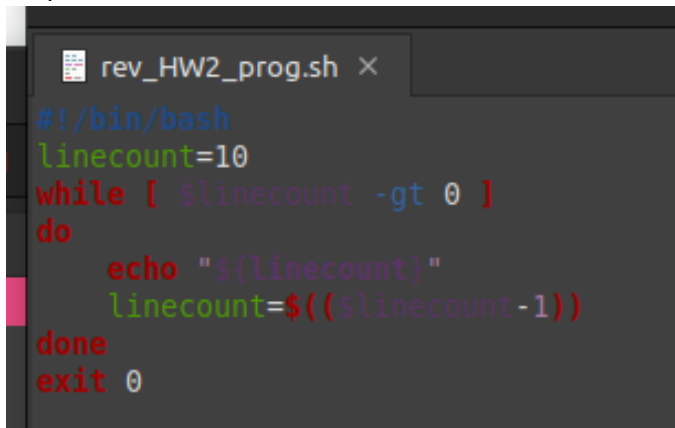
Afterward, type command:

```
$ chmod 777 rev_HW2_prog.sh
```

Then:

```
$ ./rev_HW2_prog.sh RichesIHold.txt
```

You'll see that it will not work due to an error. Correct the error(s) and provide your corrected script below:



```
rev_HW2_prog.sh x
#!/bin/bash
linecount=10
while [ $linecount -gt 0 ]
do
    echo "${linecount}"
    linecount=$((linecount-1))
done
exit 0
```

Type input:

```
$ ./rev_HW2_prog.sh RichesIHold.txt
```

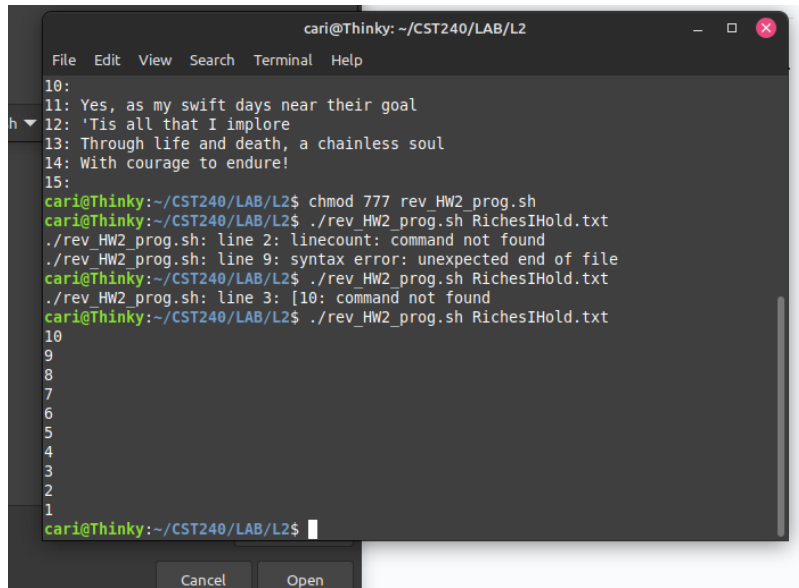
Show result below:

--	--	--

Oregon Institute of Technology

CST 240 Linux Programming

Lab #2: Shell, Encryption (C)



```
cari@Thinky: ~/CST240/LAB/L2
File Edit View Search Terminal Help
10:
11: Yes, as my swift days near their goal
12: 'Tis all that I implore
13: Through life and death, a chainless soul
14: With courage to endure!
15:
cari@Thinky:~/CST240/LAB/L2$ chmod 777 rev_HW2_prog.sh
cari@Thinky:~/CST240/LAB/L2$ ./rev_HW2_prog.sh RichesIHold.txt
./rev_HW2_prog.sh: line 2: linecount: command not found
./rev_HW2_prog.sh: line 9: syntax error: unexpected end of file
cari@Thinky:~/CST240/LAB/L2$ ./rev_HW2_prog.sh RichesIHold.txt
./rev_HW2_prog.sh: line 3: [10: command not found
cari@Thinky:~/CST240/LAB/L2$ ./rev_HW2_prog.sh RichesIHold.txt
10
9
8
7
6
5
4
3
2
1
cari@Thinky:~/CST240/LAB/L2$
```

3. (10 pts) In the **HW2** directory, create a shell script "mystere.sh". The content of mystere.sh is:

I put it in the L2 directory, since this is Lab2...

```
#!/bin/bash
# Given date, tells what day Mon-Sun date is

if [ $# -ne 3 ] ; then
    echo "Usage: $(basename $0) mon day year" >&2
    echo " with just numerical values (ex: 7 4 1993)" >&2
    exit 1
fi

date --version > /dev/null 2>&2
baddate="$?"

if [ ! $baddate ] ; then
    date -d $1/$2/$3 +"That was a %A."
else
    if [ $2 -lt 10 ] ; then
        pattern=" $2[^0-9]"
    else
        pattern="$2[^0-9]"
    fi
    dayofweek="$(ncal $1 $3 | grep "$pattern" | cut -c1-2)"
    echo $dayofweek
    fi
    exit 0
```

--	--	--

Oregon Institute of Technology

CST 240 Linux Programming
Lab #2: Shell, Encryption (C)

```
fi
    exit 0
```

Type input:

```
$ ./mystere.sh
```

Show result below:

```
cari@Thinky:~/CST240/LAB/L2$ ls
HW2_prog.sh  mystere.sh  Part1_screenshot  rev_HW2_prog.sh  RichesIHold.txt
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh
bash: ./mystere.sh: Permission denied
cari@Thinky:~/CST240/LAB/L2$
```

Type input:

```
$ ./mystere.sh Jan 3 '63
```

What is the result?

```
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh Jan 3 '63
>
>
```

Type input:

```
$ ./mystere.sh 1 3 1963
```

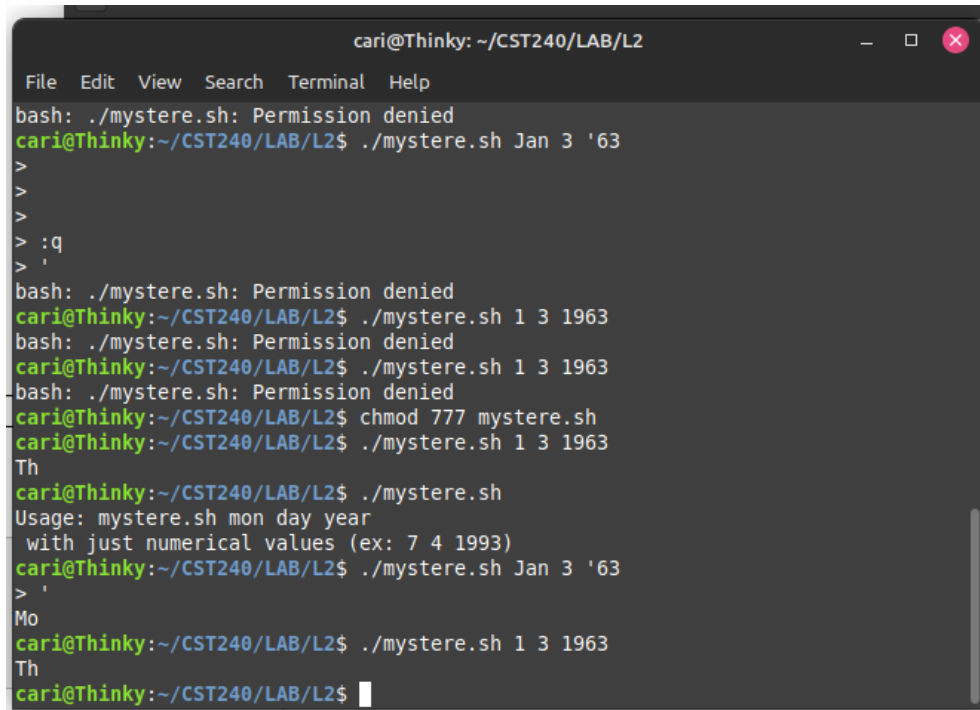
Show the results below:

Well, okay, so the above screenshots were coming from the commands before I changed the mode. I was following the instructions too exactly. So this screenshot has all of the commands after the mode change.

--	--	--

Oregon Institute of Technology

CST 240 Linux Programming Lab #2: Shell, Encryption (C)



```
cari@Thinky: ~/CST240/LAB/L2
File Edit View Search Terminal Help
bash: ./mystere.sh: Permission denied
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh Jan 3 '63
>
>
>
> :q
>
bash: ./mystere.sh: Permission denied
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh 1 3 1963
bash: ./mystere.sh: Permission denied
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh 1 3 1963
bash: ./mystere.sh: Permission denied
cari@Thinky:~/CST240/LAB/L2$ chmod 777 mystere.sh
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh 1 3 1963
Th
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh
Usage: mystere.sh mon day year
with just numerical values (ex: 7 4 1993)
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh Jan 3 '63
> '
Mo
cari@Thinky:~/CST240/LAB/L2$ ./mystere.sh 1 3 1963
Th
cari@Thinky:~/CST240/LAB/L2$
```

Describe in the space below what the mystere.sh shell script does and how it goes about doing it. Describe the following:

- What is the input including the boundaries (what isn't acceptable)
- What is the output
- How is this transformation achieved

This should be about 3 paragraphs with the 1st and 3rd paragraphs being about 3-5 sentences each, and the 2nd paragraph being about 1 sentence.

This shell script takes a date as input (month, day, year) and outputs the corresponding day of the week (Monday-Sunday) for that date. The script checks if the correct number of arguments are provided, and prints usage instructions and exits if not. It then checks if the date command is available and sets a flag variable baddate accordingly. If date is available, it uses the command to output the day of the week for the input date. If not, it uses the ncal command to get a calendar for the input month and year, searches for the line containing the input day, and extracts the day of the week from that line.

The input to the script should be three numerical values representing the month, day, and year of the date to be queried. The month should be a number between 1 and 12, the day should be a number between 1 and 31 (depending on the month and year), and the year should be a four-digit number. If the input is not in this format, or if the script is not able to determine the day of the week, it will not produce a correct output.

--	--	--

The output of the script is the day of the week corresponding to the input date, as a string. The output is achieved by using either the date or ncal command to determine the day of the week, and then printing it to the console. If the input is not in the correct format or if the script is unable to determine the day of the week, it may produce an error message or an empty output.

2. (70 Pts) C Encryption Program in C

This program reads in a text file in the same directory as the executable. The file is guaranteed to have only a-z characters on a single line. The total number of characters is guaranteed to be even. The total number of characters will be less than 100. The program will use Caesar and Railfence 2 encryption.

Use a conversion algorithm as shown in the code below.

The test file is named LAB2_sample.txt in the Week 3 module in Canvas

The program will prompt for a Caesar key, an integer 0-25. If the key is not 0-25, force the user to reenter until she/he does so.

The program may only accept lower-case characters a-z and will convert these characters based on the key and converted to upper-case characters.

The program will output the encrypted version into a text file named cipher.txt in the same directory as the executable.

Example: if the content of LAB2_sample.txt is csttwofourzero and key is 5.

Output should be:

From Caesar: HXYBTKTZWEJW

because H=c X=s Y=t etc...

From Railfence 2: H Y B K Z E W
X Y T T W J T

The final Railfence output should be: HYBKZEWXYTTWJT

Sample output from test program:

\$./encrypt.out LAB2_sample.txt

csttwofourzero

Oregon Institute of Technology

CST 240 Linux Programming Lab #2: Shell, Encryption (C)

Enter key: 5

After Caesar with key of 5:

HXYBTKTZWEJWT

After Railfence 2 and thus Final Answer in caps

HYBKZEWXYTTWJT

Again, this will be demonstrated during lab

Below is C source code that encrypts a single character. Modify this code to encrypt the characters in a file.

No comments are given so students will have to do some work in figuring things out.

```

/*****
This program takes in a lowercase letter, key (integer) and output
ciphertext
based on Caesar Cryptosystem. Input is guaranteed to be lowercase
a-z. In this
example ONLY, key is guaranteed to be +integer <1000. This key
"guarantee"
is NOT the same as specification of lab which forces key to be 0-25.
*****/
/
#include <stdio.h>

int main()
{
    char plaintext, ciphertext;
    int plain;
    char continu = 'y';
    int key; while (continu == 'y')
    {
        printf("Enter lowercase character to be encrypted using
Caesar: ");
        scanf(" %c", &plaintext);
        plain = (int)(plaintext) - 97; // plaintext - 97 works also
        printf("%d\n", plain);
        printf("Enter key, a number between 0 and 1000:");
        scanf("%d", &key);
        key = key%26; // can't do this in your program. Need to force
user to input 0-25
    }
}
```

Oregon Institute of Technology

CST 240 Linux Programming

Lab #2: Shell, Encryption (C)

```
    printf("Key is %d\n", key);
    ciphertext = ((plain+key)%26)+97;
    printf("Ciphertext is %c\n",ciphertext);
    printf("Enter y to do again: ");
    scanf(" %c", &continuu);
}
return 0;
}
```

Grading:

Caesar code working correctly:	20 pts
Railfence code working correctly:	20 pts
Code documentation:	15 pts
Code testing:	15 pts
Total:	70 pts

For documentation, be sure to include your name, the date, the class and assignment at the top.

Also be sure to describe what every major section of code is doing in a comment.

When testing, be sure to test boundary conditions and unexpected data along with expected data. You should have at least 5 different tests for each encryption method.

--	--	--