

Lab 06 – Scripting

Name: Jason Guzman

Course/Section: IS-1003-202520

Date: 4/15/2025

INTRODUCTION

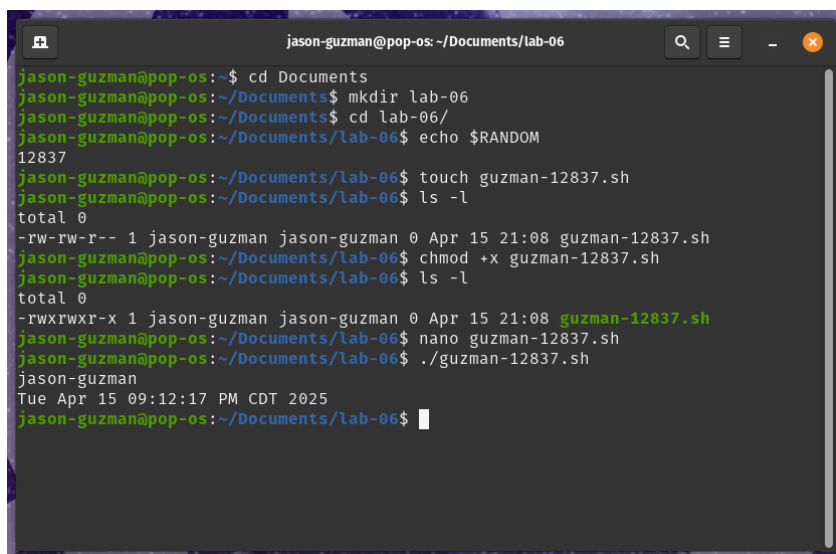
The goals of this lab are to write a Bash script in the Linux text editor Nano by designing a code structure for 3 functions each with a different task.

BREAKPOINT 1

The first step in this lab was to create a directory to house the lab documents. First, I changed directories to the Documents folder and used the “mkdir” command to make a directory titled “lab-06”. Next, I generated a random number using “\$RANDOM” and “echo” to print it to the terminal. The number that was generated was 12837 and I used that to create a shell script named “guzman-12837.sh” using the command “touch”. I then listed the files in the current directory to verify that it was created.

The next step was to make the file executable, I used the command “chmod” with the +x option to make the shell script executable. This is shown as the file changes from “-rw-rw-r--” to “-rwxrwxr-x”. I then opened the file in Nano by using the command “nano guzman-12837.sh”. This then opened the nano editor and I typed the shebang “#!/bin/bash” and below that I typed the “whoami” and “date” command to verify that the script runs.

After saving the script, I went back into the terminal and executed the file using “./guzman-12837.sh” and got the output “Jason-guzman” and “Tue Apr 15 09:12:17 PM CDT 2025” this verifies that the “whoami” and “date” commands were executed correctly.

A screenshot of a terminal window titled 'jason-guzman@pop-os: ~/Documents/lab-06'. The terminal shows the following commands and output:

```
jason-guzman@pop-os:~$ cd Documents
jason-guzman@pop-os:~/Documents$ mkdir lab-06
jason-guzman@pop-os:~/Documents$ cd lab-06/
jason-guzman@pop-os:~/Documents/lab-06$ echo $RANDOM
12837
jason-guzman@pop-os:~/Documents/lab-06$ touch guzman-12837.sh
jason-guzman@pop-os:~/Documents/lab-06$ ls -l
total 0
-rw-rw-r-- 1 jason-guzman jason-guzman 0 Apr 15 21:08 guzman-12837.sh
jason-guzman@pop-os:~/Documents/lab-06$ chmod +x guzman-12837.sh
jason-guzman@pop-os:~/Documents/lab-06$ ls -l
total 0
-rwxrwxr-x 1 jason-guzman jason-guzman 0 Apr 15 21:08 guzman-12837.sh
jason-guzman@pop-os:~/Documents/lab-06$ nano guzman-12837.sh
jason-guzman@pop-os:~/Documents/lab-06$ ./guzman-12837.sh
jason-guzman
Tue Apr 15 09:12:17 PM CDT 2025
jason-guzman@pop-os:~/Documents/lab-06$
```

Figure 1 Setting up scripting environment

BREAKPOINT 2

The next step in the lab is to create a framework for our code. Running the script should run 3 functions that will display the current date and time, display public and private IP addresses and the open ports for my machine, and provide the capability to encrypt and decrypt a string using a Caesar cipher.

The first step was to create a section of the code to house the functions, I did this by using a comment to show where the function bodies start. Below that I created a function titled “current_date” with no parameters and a body that prints “Todays date” using the “echo” function. This is shown below.

Current date function

Next, I created the function to display the IP addresses and ports on my machine. I titled it “ip_ports” with no parameters and a body that echoes 3 lines code listing “Private IP address: “, “Public IP address: “, and “Open ports: “.

```
10
11 # A function to display the IP addresses and ports on this machine
12 ip_ports() {
13     # Display private Ip address =es using hostname
14     echo "Private IP address: "
15     # Display public IP address using ifconfig.,e
16     echo "Public IP address: "
17     # Diplay open ports using nmap
18     echo "Open ports: "
19 }
20
```

Figure 2 IP and port display functions

The last function I created was the Ceaser cipher function. This function is titled “cipher” with no parameters and a body that echoes placeholder text.

```
20
21 # A function to encrypt and decrypt a string using a simple Caesar cipher
22 cipher () {
23     echo "We will ask te user for a string, and then encrypt/decrypt that string"
24 }
25
```

Figure 3 Caesar cipher function

The next step in code designing was to make a section for the function calls. I used a comment to show where the section begins and called the 3 functions.

```
26
27 #####Function calls #####
28
29 # Displays the formatted date and time
30 current_date
31
32 # Displays the public/private IP addresses and ports in use
33 ip_ports
34
35 # Encrypts and decrypts a user string
36 cipher
37
```

Figure 4 Calling the 3 functions of the script

Below is the execution of the shell script showing the success of the framework.

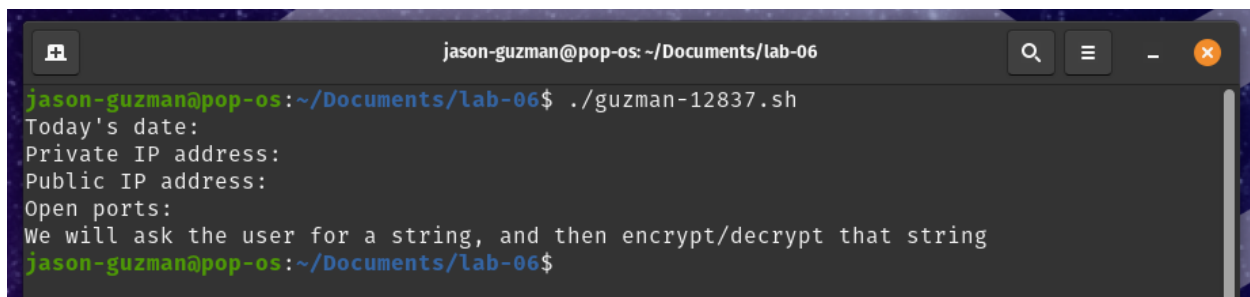
A screenshot of a terminal window with a dark background. The window title bar shows 'jason-guzman@pop-os: ~/Documents/lab-06'. The terminal content shows the user running './guzman-12837.sh'. The script outputs: 'Today's date:', 'Private IP address:', 'Public IP address:', 'Open ports:', and 'We will ask the user for a string, and then encrypt/decrypt that string'. The prompt returns to 'jason-guzman@pop-os:~/Documents/lab-06\$'.

Figure 5 Executing the script

Design is important in code because it lays down the foundation of the script and gives a roadmap of the product. Comments are important in code so you or anyone else editing your code and understand what the purpose of the code section is.

BREAKPOINT 3

To start working on the date and time function, I created a variable named “today” to store the date to be called in the function. I used the date command with the format of month-day-year. I did the same with the time variable. I created a variable named “current_time” to store the current time by using the date command with the format of hour:minute:seconds and AM or PM. This is shown below.

```

4
5 # A fucntion to diplay the date and time
6 current_date () {
7     # Display the current date
8     today=$(date +%m-%d-%y)
9     echo -e "\nToday's date: $today "
10
11     # Display the time
12     current_time=$(date +%H:%M:%S %p)
13     echo -e "The current time: $current_time\n"
14 }

```

Figure 6 Creating functionality for current date function

This is the output of executing the shell script with the current date function working properly.

```

jason-guzman@pop-os:~/Documents/lab-06$ ./guzman-12837.sh

Today's date: 04-15-25
The current time: 22:16:32 PM

```

Figure 7 Verifying function is working

2 conventions used in this code are implementation comments and lowercase function titles spaced by a “_”.

BREAKPOINT 4

To create functionality of the ip_port function I started with the private IP address part. Under the echo of the private IP address, I added the command “hostname -I” to display my private IP address, then I used the command “curl ifconfig.me” to display my public IP address, and finally I added “nmap scanme.nmap.org” below the echo for open ports to display the open ports on my network.

```

15
16 # A function to display the IP addresses and ports on this machine
17 ip_ports () {
18     # Display private Ip address using hostname
19     echo "Private IP address: "
20     hostname -I
21
22     # Display public IP address using ifconfig.,e
23     echo -e "\nPublic IP address: "
24     curl ifconfig.me
25
26     # Diplay open ports using nmap
27     echo -e "\nOpen ports: "
28     nmap scanme.nmap.org
29     echo -e "\n"
30 }
31

```

Figure 8 Adding functionality to IP port function

Below is the verification that the above code is functional and works properly.

```
jason-guzman@pop-os:~/Documents/lab-06$ ./guzman-12837.sh

Today's date: 04-15-25
The current time: 22:35:18 PM

Private IP address:
10.0.2.15 fd00::2961:1948:139b:e02c fd00::bc8b:2b2b:ea2f:59ab

Public IP address:
35.147.62.132
Open ports:
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-15 22:35 CDT
Nmap scan report for scanme.nmap.org (45.33.32.156)
Host is up (0.068s latency).
Other addresses for scanme.nmap.org (not scanned): 2600:3c01::f03c:91ff:fe18:bb2f
Not shown: 996 filtered ports
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
9929/tcp  open  nping-echo
31337/tcp open  Elite

Nmap done: 1 IP address (1 host up) scanned in 5.90 seconds
```

Figure 9 Testing IP address functionality

BREAKPOINT 5

To create the functionality of the cipher function I used the provided code and formatted it for readability and added comments.

```
31
32 # A function to encrypt and decrypt a string using a simple Caesar cipher
33 cipher () {
34     run=true
35     # Create a loop that runs until !run
36     while $run; do
37         read -p "Let's encrypt a string! Enter a string of your choice (or '0' to quit): " input
38         # If else statement that continues or terminates the loop depending on user input
39         if [[ "$input" == "0" ]]; then
40             echo "Thank you! Bye!!"
41             run=false
42         else
43             encrypted=$(echo "$input" | tr 'a-zA-Z' 'd-za-cD-ZA-C')
44             decrypted=$(echo "$encrypted" | tr 'a-zA-Z' 'x-za-wX-ZA-W')
45             echo "Encrypted: $encrypted"
46             echo "Decrypted: $decrypted"
47         fi
48     done
49 }
```

Figure 10 Cipher functionality

I then tested the cipher function with 2 user inputs and then "0" to quit the loop.

```
Let's encrypt a string! Enter a string of your choice (or '0' to quit): Test string
Encrypted: Whvw vwulqj
Decrypted: Test string
Let's encrypt a string! Enter a string of your choice (or '0' to quit): another one
Encrypted: dqrwkhu rqh
Decrypted: another one
Let's encrypt a string! Enter a string of your choice (or '0' to quit): 0
Thank you! Bye!!
jason-guzman@pop-os:~/Documents/lab-06$
```

Figure 11 Testing functionality of cipher function

CONCLUSION

A challenge that I faced in this lab was with formatting the provided code for the cipher function. I did not copy the text correctly, so I ran into an unexpected token. I then recopied the code line by line and tested it again. Testing code frequently is a good habit because you can isolate the errors and not have to fix an entire codebase.

REFERENCES

R. Mitra, "Lab 06: Scripting," The University of Texas at San Antonio (2024). Last accessed: 4/15/25

COLLABORATION

I worked on this lab solely with the instruction of Professor Mitra.