Lab 2 - Bash Scripting and Automation

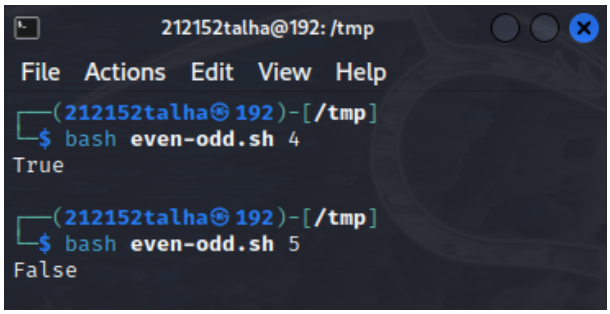
Name: Muhammad Talha Attari
Roll Number: 212152
Course: CY243-L - Penetration Testing Lab
Date: 11/10/2023
Total Questions: 4
Attempted Questions: 4

Task - 1

• Write a bash script that takes a number as an argument and prints whether the number is even or odd. The output should be "True" or "False". Case matters. The file must be inside `/tmp/` directory and named as `even-odd.sh`.

- Commands used

```
#!/bin/bash

if [ $(( $1 % 2 )) -eq 0 ]; then
echo "True"
else
echo "False"
fi
```



Task - 2

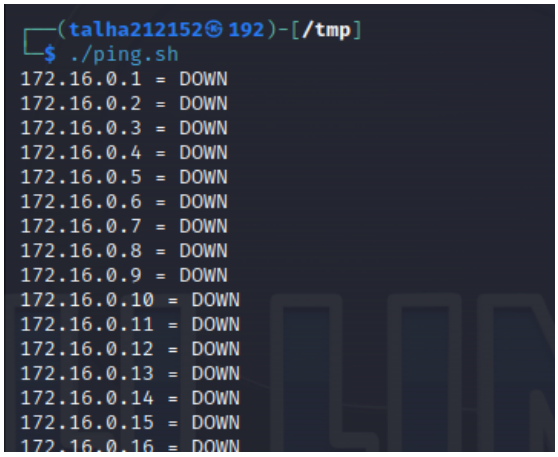
• Using a for loop in bash, try and ping the subnet "172.16.0.0/24" and print the IP addresses that are up. The output should be like: "172.16.0.0 = UP”

> Hint: Use ping -c 1 <ip-address> to ping the IP address once.

The file must be inside `/tmp/` directory and named as `ping.sh`.

- Commands used in `ping.sh`

```
#!/bin/bash

subnet="172.16.0."
start=1
end=254
for ((i=start; i<=end; i++)); do
  ip_address="$subnet$i"
if ping -c 1 -W 1 "ip_address" &> /dev/null; then
  echo "$ip_address = UP"
else
  echo "$ip_address = DOWN"
fi
done
```



When I run this program my output does not look like: “172.16.0.0 = UP” as seen in the screenshot above. I have carefully dry run the above script line-by-line but the subnet keeps responding DOWN.

Task - 3

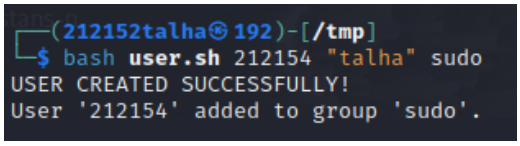
• Create a function called `create_user` that takes two arguments: username and password. The function should create a user with the given username and password. Also, write another function called `add_to_group` that takes two arguments: username and groupname. The function should add the user to the given group. The file must be inside `/tmp/` directory and named as `user.sh`. The username, password, and groupname should be provided from the command line as arguments to the script. Example usage would be:

```
/tmp/user.sh new_user "my-password" sudo
```

Also, add a check to see if the group exists or not.

- Commands used

```
#!/bin/bash

create_user() {
    local usr_name="$1"
    local pass_wd="$2"
    sudo useradd -m -s /bin/bash "$usr_name" && echo "$usr_name:$pass_wd" | sudo chpasswd
    echo "USER CREATED SUCCESSFULLY!"
}

add_to_group() {
    local usr_name="$1"
    local grp_name="$2"
    if sudo getent group "$grp_name" > /dev/null 2>&1; then
        sudo usermod -aG "$grp_name" "$usr_name"
        echo "User '$usr_name' added to group '$grp_name'."
    else
        echo "Group '$grp_name' does not exist."
    fi
}

if [ "$#" -ne 3 ]; then  #(The '#' here contains the number of arguments)
    echo "Error, Enter the following three arguments in order to create a user "
    echo " UserName, Password, GroupName"
    exit 1
fi

usr_name="$1"
pass_wd="$2"
grp_name="$3"


create_user "$usr_name" "$pass_wd"   #(Calling the function to create user)
add_to_group "$usr_name" "$grp_name" #(Calling the function to add new user to group)
```



As you can see in the above screenshot that the username, password and groupname are provided from the command line as arguments to the script.

Task - 4

Part 1: Write a script (can also be in `Python`) that will generate a file called (`num-info.txt`) that contains numbers from 0-1000 and each number is prefixed with an alphabet (only Uppercase) and the next number with next alphabet. (Re-loop after 26). i.e.

```
A0
B1
C2
..
..
..
K998
L999
M1000
```

- Commands used (Script that I came up with)

```
#!/bin/bash

# This is the method I found on google to how to create a file
> num-info.txt

letters=("A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z")

for i in {0..1000}; do
    letter_index=$((i % 26))
#echo "$letter_index'
    letter="${letters[$letter_index]}"
echo "${letter}${i}" >> num-info.txt
echo "${letter}${i}"
done

#Also printing the number of lines in num-info.txt
wc -l num-info.txt
```

Part 2: Extract only the numbers that are prefixed with `B` and `C`. Storing both in seperate files `B-num.txt` and `C-num.txt` respectively.**[Must be in Bash]**

Also, print the number of lines: `wc -1`

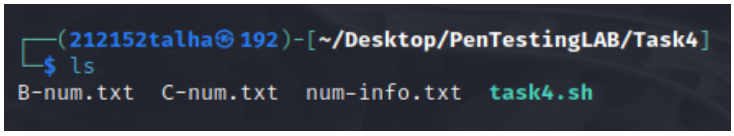
- Commands used (Adding Part 2 in the script)

```
#!/bin/bash

# This is the method I found on google to how to create a file
> num-info.txt

letters=("A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V" "W" "X" "Y" "Z")

for i in {0..1000}; do
    letter_index=$((i % 26))
#echo "$letter_index'
    letter="${letters[$letter_index]}"
echo "${letter}${i}" >> num-info.txt
echo "${letter}${i}"
done

## PART 2 starts from here!

#Also printing the number of lines in num-info.txt
wc -l num-info.txt

#Storing the number extracted with Prefix B and number of lines in B-num.txt
grep '^B' num-info.txt | cut -c2- > B-num.txt
wc -l B-num.txt >> B-num.txt

#Storing the number extracted with Prefix C and number of lines in C-num.txt
grep '^C' num-info.txt | cut -c2- > C-num.txt
wc -l C-num.txt >> C-num.txt
```

In the above script, I first used the grep command to find the required prefixes, after that I used the output of the grep command as an input in **cut** command. The **'-c2-'** in cut command is doing the following:

- **'-c'** stands for "characters", indicating that I am working with individual characters in each line of text.
- **'2-'** specifies the range of characters to keep. In the case of this task, it starts at the second character of line and goes to the end of line meaning that it excludes the first character that is in this is the prefix i.e. B or C.

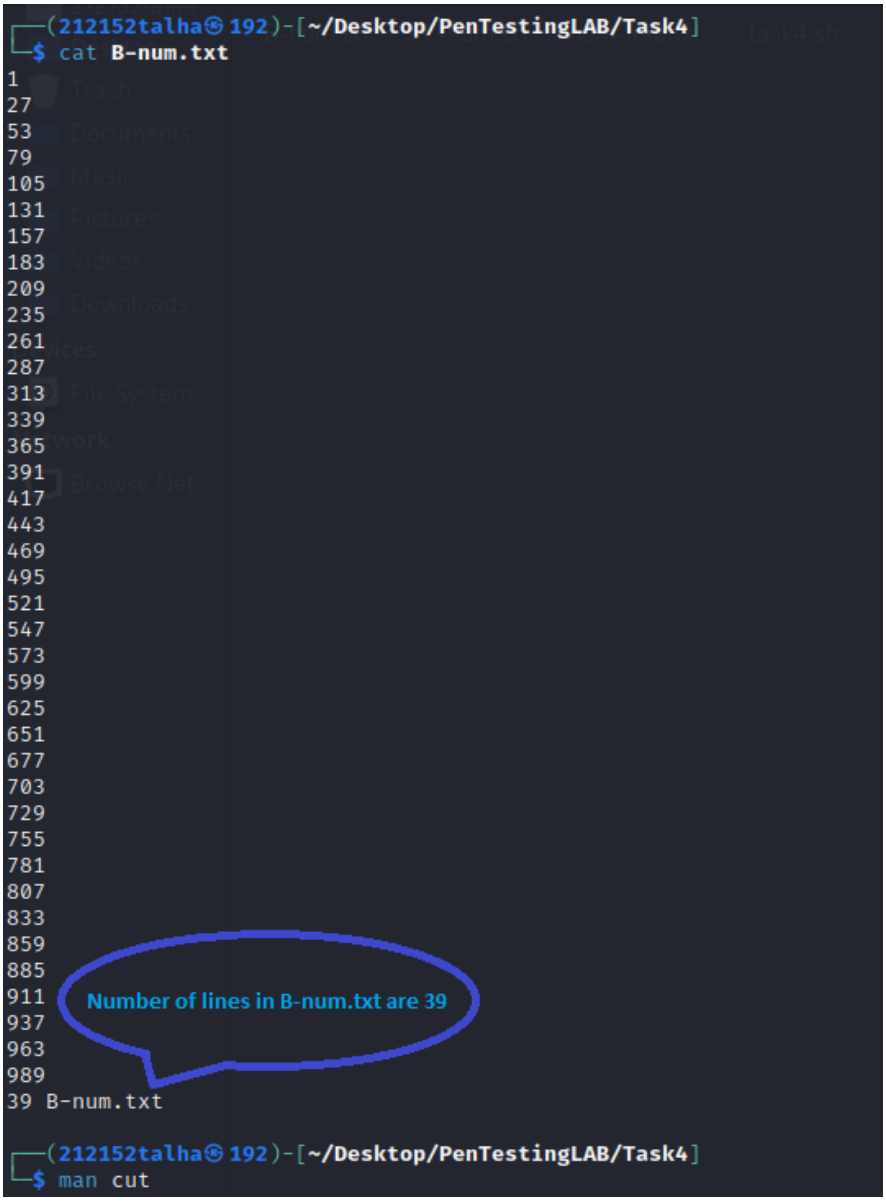
After running the above script, following files are created:



On viewing B-num.txt using the following command, we get the following output as required:

- Commands used:

```
cat B-num.txt
```



On viewing C-num.txt using the following command, we get the following output as required:

- Commands used:

```
cat C-num.txt
```