International Cybersecurity and Digital Forensic Academy

PROGRAMME: CYBERSECURITY AND ETHICAL HACKING INTERNSHIP

ASSIGNMENT

PRESENTED BY

AYILARA BUSARI DARE

IDEAS/24/28133

COURSE CODE: INT301

INT301: Operating Systems Fundamentals – week2 Lab 1: Introduction to Bash Scripting

Part 1: Bash Scripting Basics

1.1 What is a Bash Script?

Bash scripting is a way to automate tasks in a Linux environment using commands that are written in the shell scripting language. Bash (Bourne Again Shell) is one of the most popular shells used in Linux.

1.2 Creating a Simple Script

Let's start by creating a simple script to display text.

1. Open your terminal.
2. Create a new file with a .sh extension:

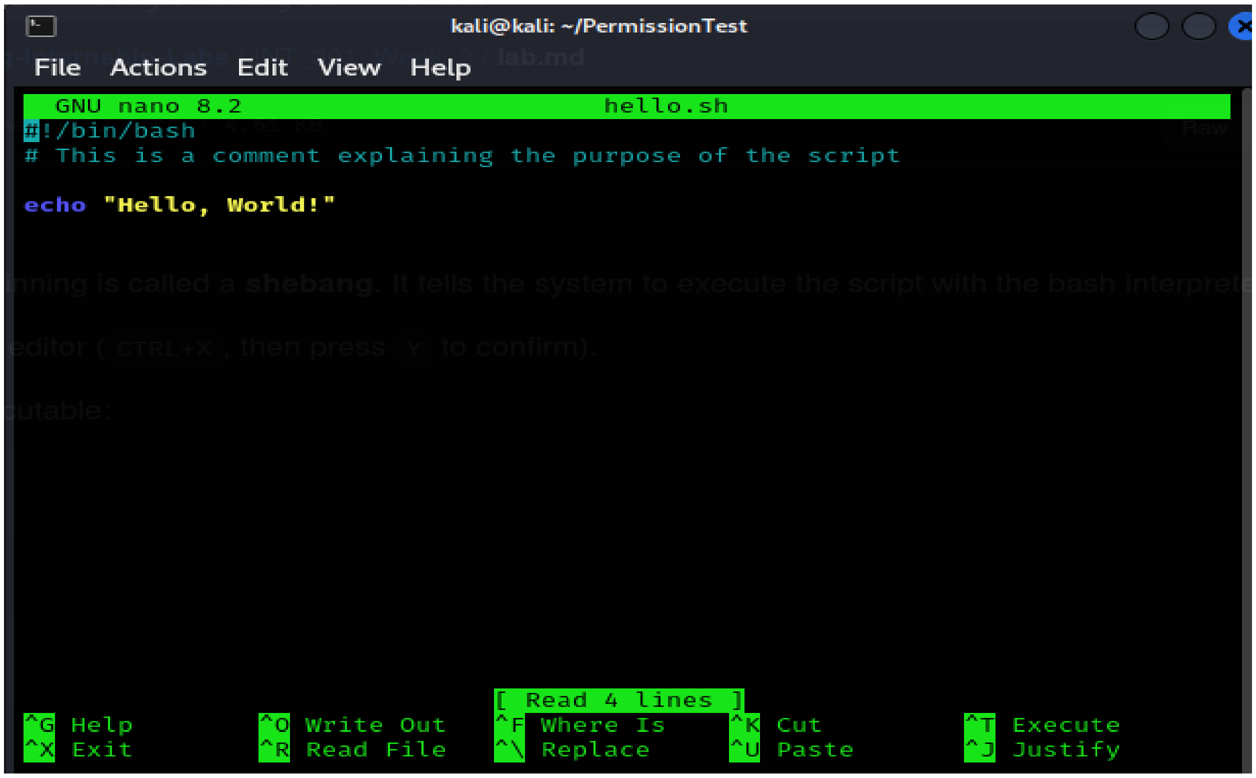
nano hello.sh

Inside the file, write the following lines:

#!/bin/bash

This is a comment explaining the purpose of the script

echo "Hello, World!"

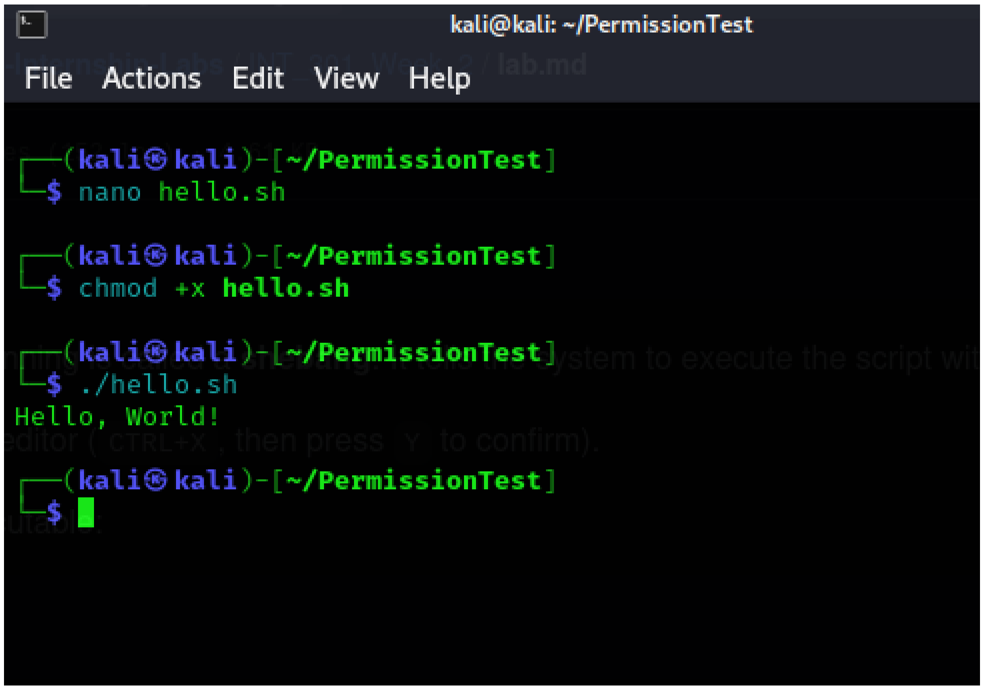


```
  GNU nano 8.2                           hello.sh
#!/bin/bash
# This is a comment explaining the purpose of the script

echo "Hello, World!"

                              [ Read 4 lines ]
^G Help          ^O Write Out     ^F Where Is      ^K Cut           ^T Execute
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify
```

The #!/bin/bash at the beginning is called a **shebang**. It tells the system to execute the script with the bash interpreter. Save the file and exit the text editor (CTRL+X, then press Y to confirm).

To run the script, make it executable: **chmod +x hello.sh**

Now, execute the script: **./hello.sh**



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano hello.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ chmod +x hello.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Hello, World!

┌──(kali㉿kali)-[~/PermissionTest]
└─$
```

1.3 Adding Variables

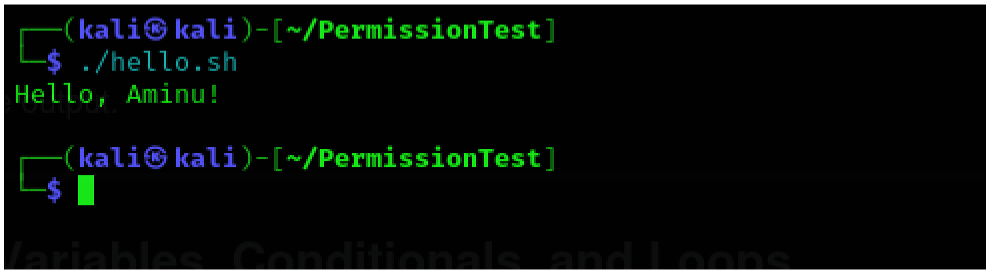
In bash scripting, variables can store data and be used later in the script.

1. Modify hello.sh to include a variable:

   #!/bin/bash

   name="Aminu"

   echo "Hello, $name!"



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Hello, Aminu!

┌──(kali㉿kali)-[~/PermissionTest]
└─$
```

Part 2: Working with Variables, Conditionals, and Loops

2.1 Using Variables

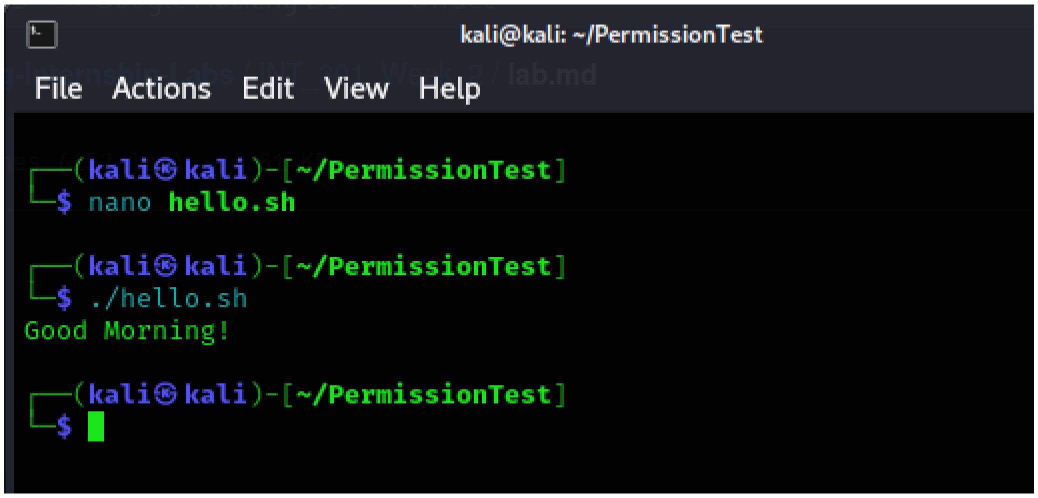
In bash, variables don't need to be declared with a type. You simply assign values to them.

Example:

#!/bin/bash

greeting="Good Morning!"

echo $greeting



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano hello.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Good Morning!

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

2.2 Conditionals

You can use if-else statements to control the flow of your script based on conditions.

Example:

#!/bin/bash

read -p "Enter your age: " age

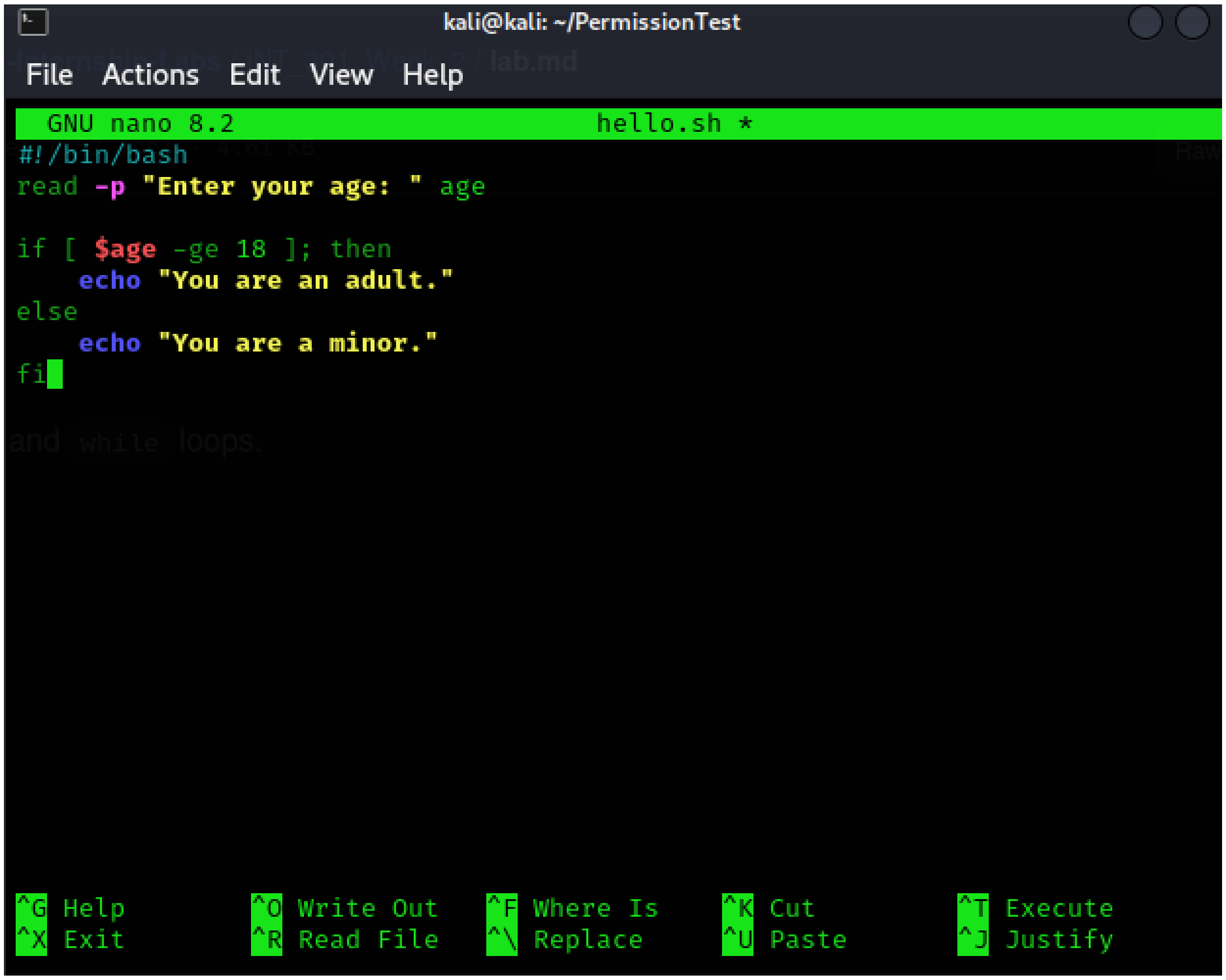
if [$age -ge 18]; then

  echo "You are an adult."

else

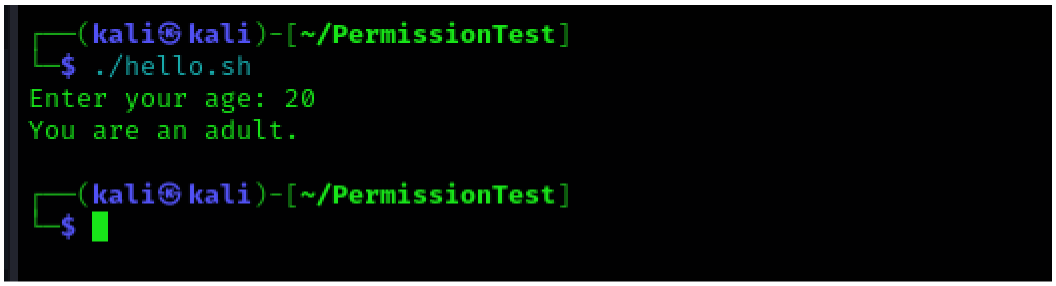
  echo "You are a minor."

Fi



```
  GNU nano 8.2                       hello.sh *
#!/bin/bash
read -p "Enter your age: " age

if [ $age -ge 18 ]; then
    echo "You are an adult."
else
    echo "You are a minor."
fi
```

```
^G Help      ^O Write Out  ^F Where Is   ^K Cut     ^T Execute
^X Exit      ^R Read File  ^\ Replace    ^U Paste   ^J Justify
```



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Enter your age: 20
You are an adult.

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

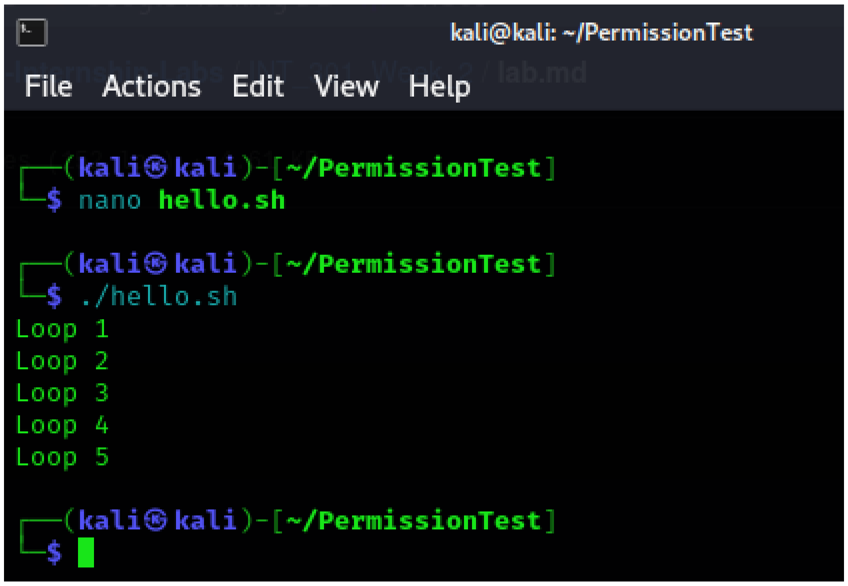
2.3 Loops

Bash scripting supports both for and while loops.

Example of a for loop:

```
#!/bin/bash

for i in {1..5}; do

  echo "Iteration $i"
```

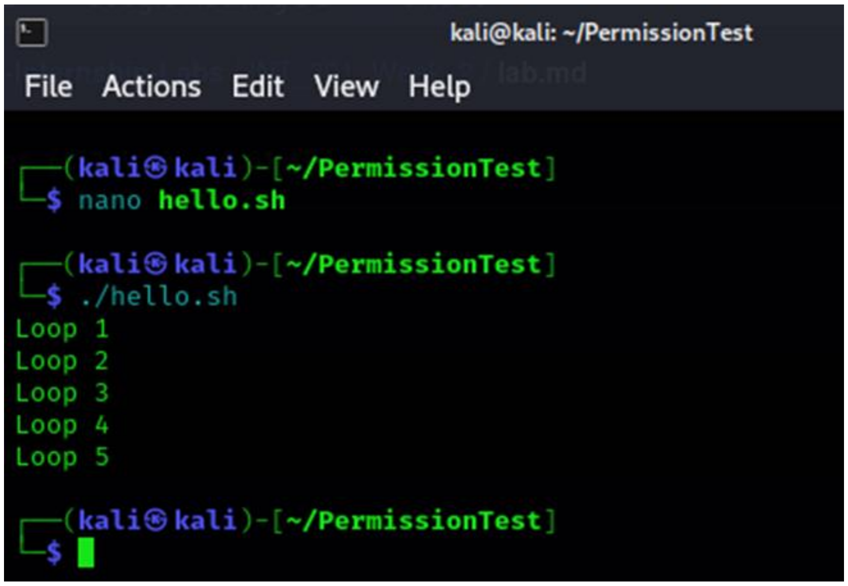
done



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano hello.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

Example of a while loop:

```
#!/bin/bash

count=1

while [ $count -le 5 ]; do

  echo "Loop $count"

  ((count++))

done
```



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano hello.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
Loop 1
Loop 2
Loop 3
Loop 4
Loop 5

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

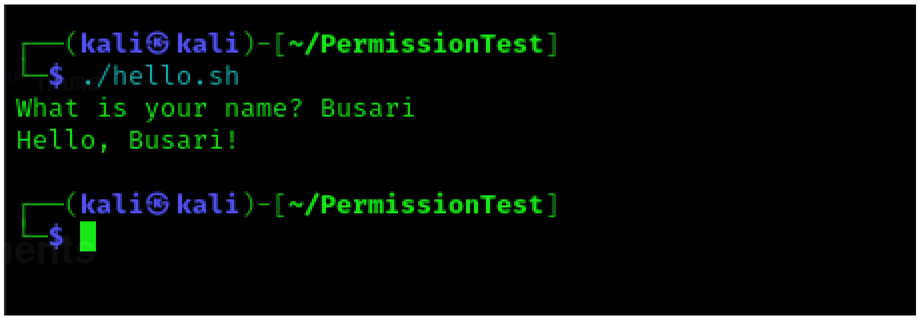
Part 3: Working with User Input

3.1 Reading User Input

Bash scripts can interact with the user by accepting input using the read command.

Example:

```
#!/bin/bash

read -p "What is your name? " name

echo "Hello, $name!"
```



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
What is your name? Busari
Hello, Busari!

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

3.2 Command-Line Arguments

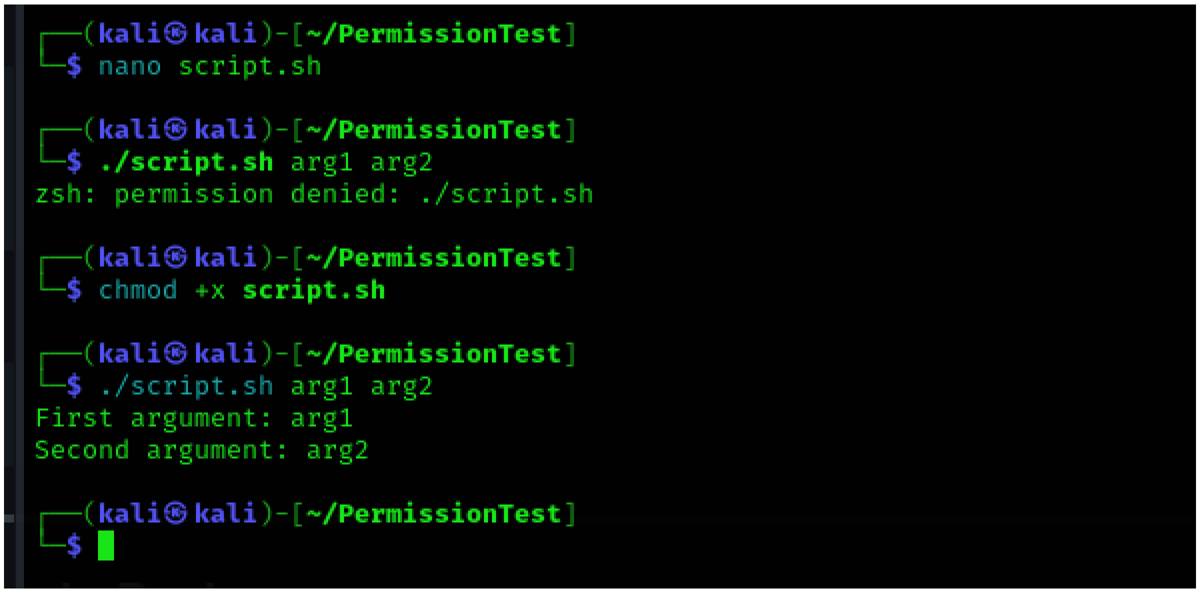
Scripts can also accept arguments when they are run.

Example:

```
#!/bin/bash

echo "First argument: $1"

echo "Second argument: $2"
```

Run it as:

```
./script.sh arg1 arg2
```



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano script.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./script.sh arg1 arg2
zsh: permission denied: ./script.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ chmod +x script.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./script.sh arg1 arg2
First argument: arg1
Second argument: arg2

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

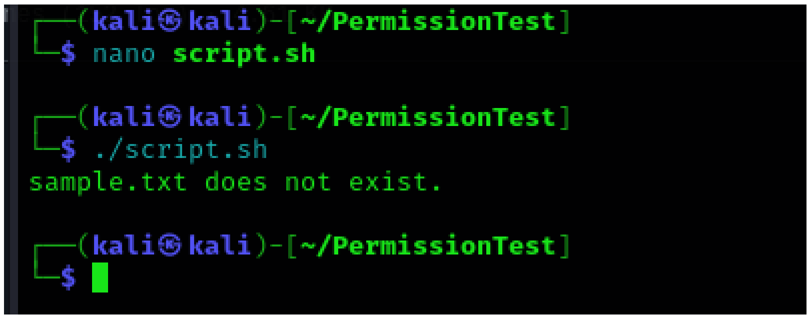
Part 4: File Operations in Bash

4.1 Checking if a File Exists

You can check if a file exists using conditionals.

Example:

```
#!/bin/bash

file="sample.txt"

if [ -e $file ]; then

  echo "$file exists."

else

  echo "$file does not exist."

Fi
```



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano script.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./script.sh
sample.txt does not exist.

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

4.2 Writing to a File

You can redirect output to a file.

Example:

#!/bin/bash

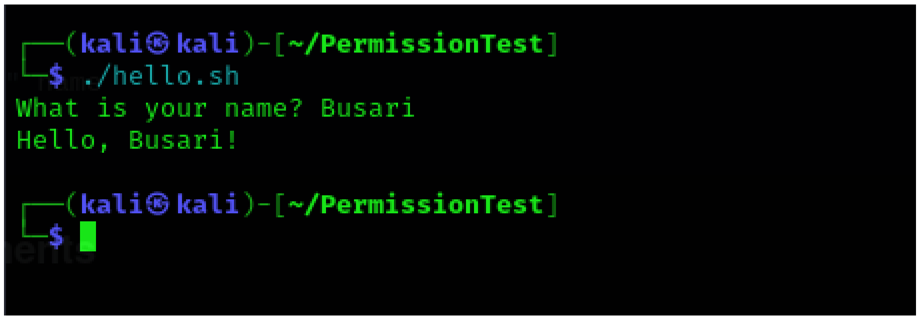
echo "This is some text." > output.txt

Part 5: Exercises

Exercise 1: Create a Script that Greets the User

Write a bash script that:

1. Prompts the user for their name.
2. Outputs a greeting that includes their name.

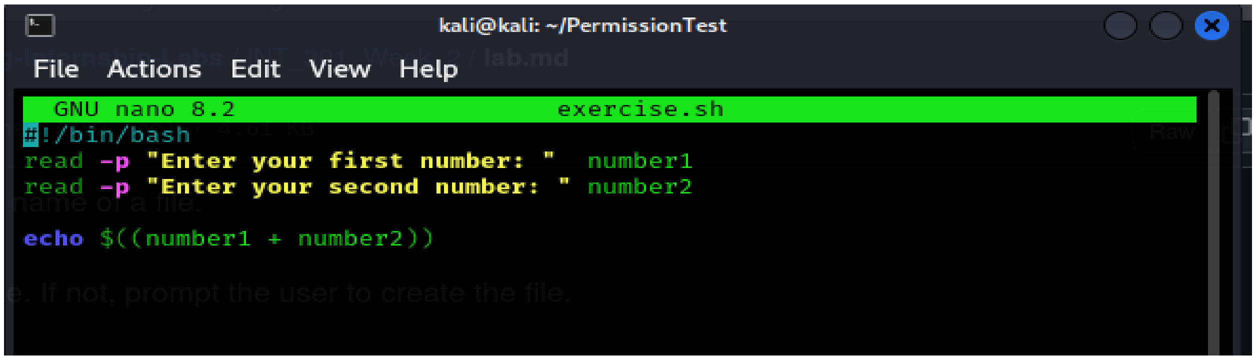


```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./hello.sh
What is your name? Busari
Hello, Busari!

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

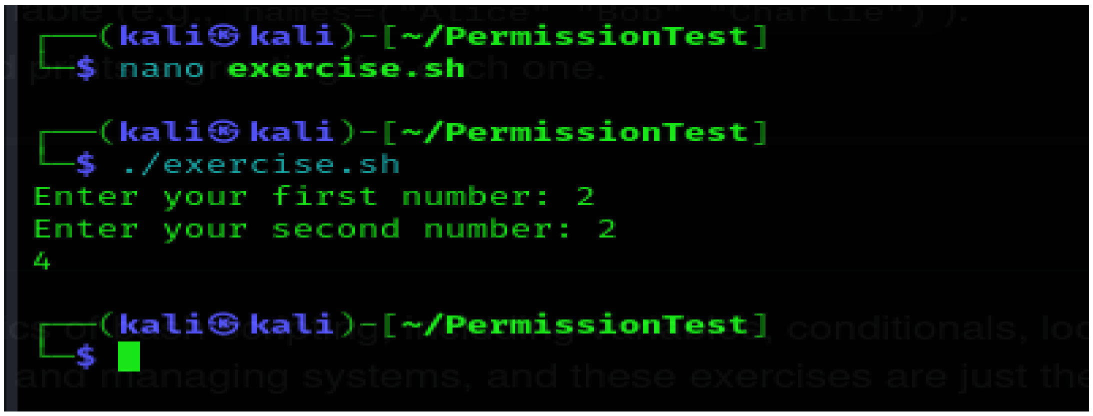
Exercise 2: Simple Calculator

Write a script that:

1. Prompts the user for two numbers.
2. Prompts the user to choose an operation (addition, subtraction, multiplication, or division).
3. Performs the selected operation on the two numbers and displays the result.



```
#!/bin/bash
read -p "Enter your first number: "  number1
read -p "Enter your second number: " number2

echo $((number1 + number2))
```

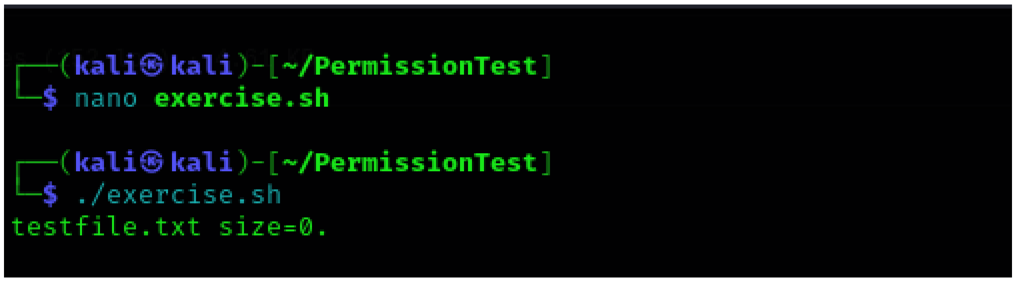


```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano exercise.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./exercise.sh
Enter your first number: 2
Enter your second number: 2
4

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

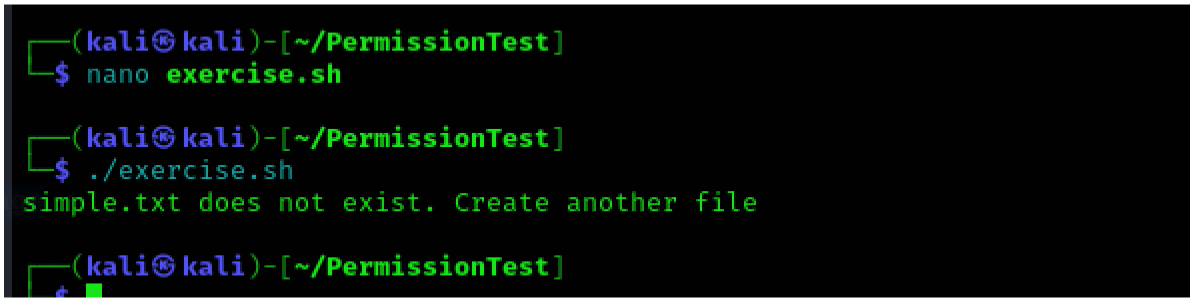
Exercise 3: File Checker

Write a script that:

1. Prompts the user to enter the name of a file.
2. Checks if the file exists.
3. If the file exists, display its size. If not, prompt the user to create the file.



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano exercise.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./exercise.sh
testfile.txt size=0.
```

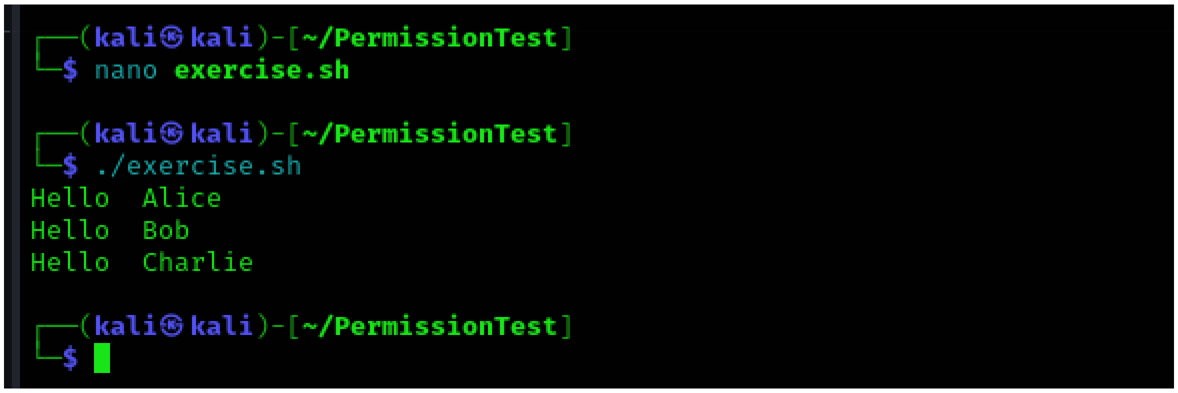


```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano exercise.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./exercise.sh
simple.txt does not exist. Create another file

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

Exercise 4: Loop through a List of Names

Write a script that:

1. Stores a list of names in a variable (e.g., names=("Alice" "Bob" "Charlie")).
2. Loops through the names and prints a greeting for each one.



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano exercise.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./exercise.sh
Hello  Alice
Hello  Bob
Hello  Charlie

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

Lab 2: Automating Linux Command Shell Navigation

Part 1: Automating Directory Navigation

1.1 Basic Navigation

Start by automating the process of navigating directories.

1. Open a terminal and create a bash script:

```
nano auto_navigate.sh
```

In the script, write the following commands to navigate the file system:

```
#!/bin/bash

# Automated navigation script


echo "Navigating to /home directory..."

cd /home

pwd


echo "Listing files in /home..."

ls -la
```

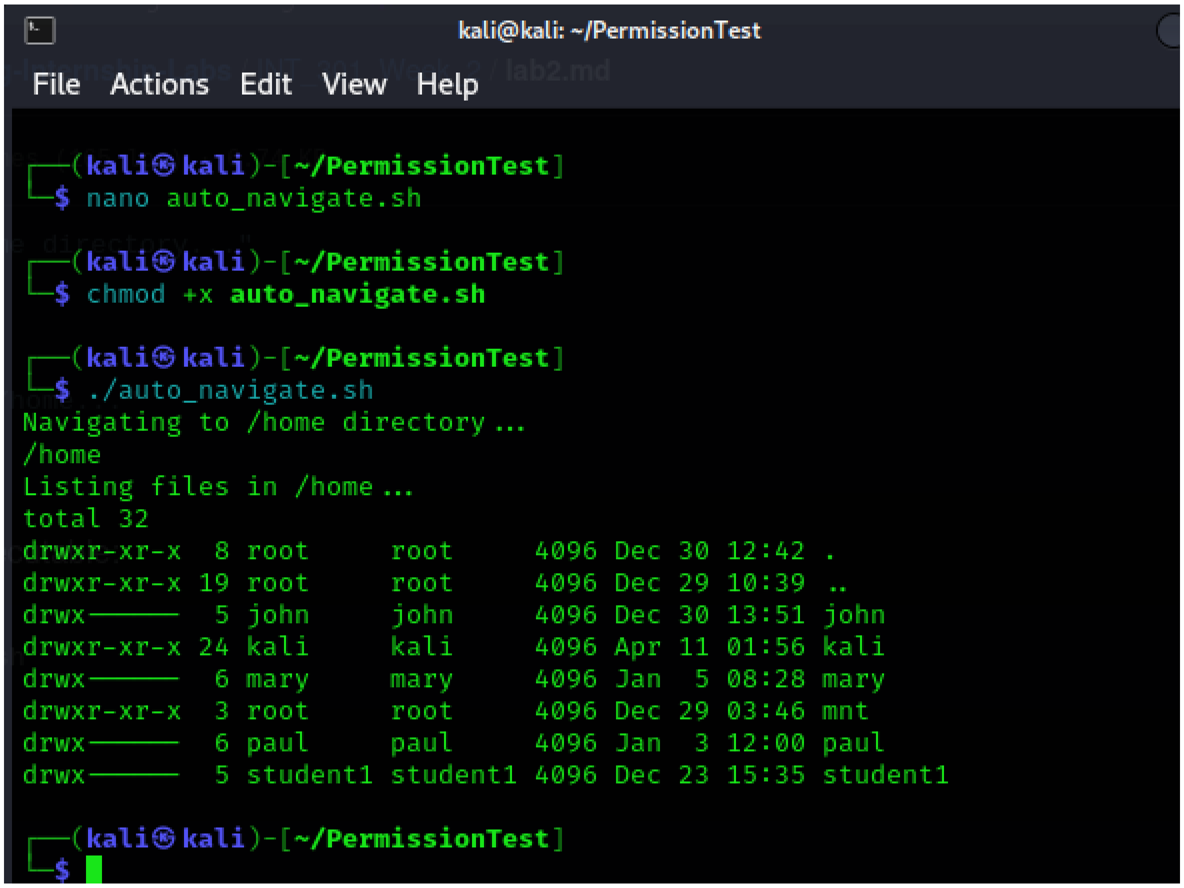
Save and make the script executable:

```
chmod +x auto_navigate.sh
```

Run the script:

```
./auto_navigate.sh
```

This script will automatically navigate to the /home directory, display the current location using pwd, and list all the files and directories within /home using ls -la.



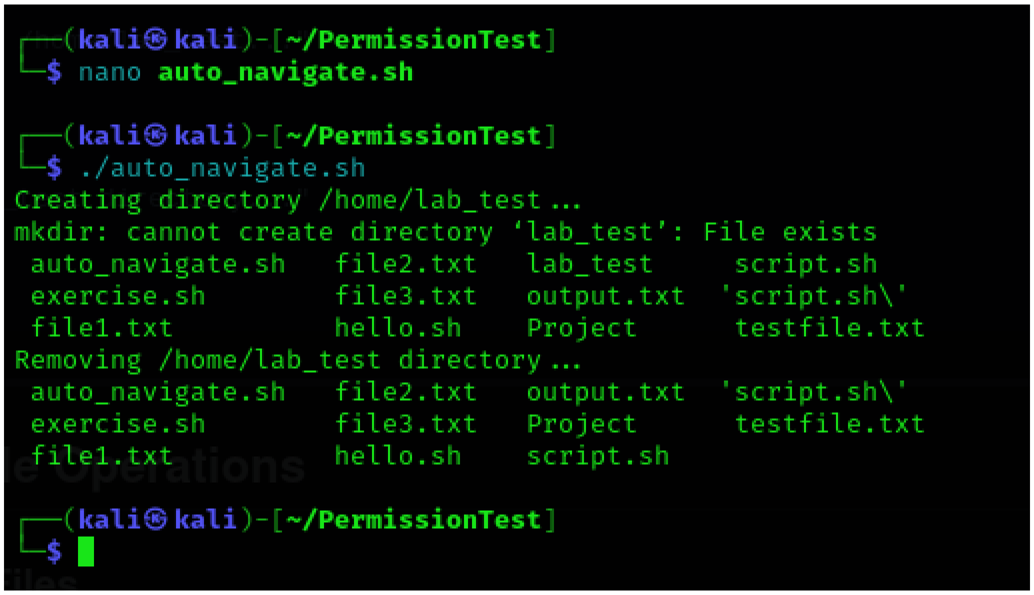
```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ chmod +x auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Navigating to /home directory...
/home
Listing files in /home...
total 32
drwxr-xr-x  8 root     root     4096 Dec 30 12:42 .
drwxr-xr-x 19 root     root     4096 Dec 29 10:39 ..
drwx------  5 john     john     4096 Dec 30 13:51 john
drwxr-xr-x 24 kali     kali     4096 Apr 11 01:56 kali
drwx------  6 mary     mary     4096 Jan  5 08:28 mary
drwxr-xr-x  3 root     root     4096 Dec 29 03:46 mnt
drwx------  6 paul     paul     4096 Jan  3 12:00 paul
drwx------  5 student1 student1 4096 Dec 23 15:35 student1

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

1.2 Automating Directory Creation and Removal

Scripts can automate directory creation and cleanup tasks.

1. Add to your script to create and remove directories:
2. echo "Creating directory /home/lab_test..."
3. mkdir /home/lab_test
4. ls /home
5.
6. echo "Removing /home/lab_test directory..."
7. rmdir /home/lab_test

ls /home



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Creating directory /home/lab_test ...
mkdir: cannot create directory 'lab_test': File exists
 auto_navigate.sh   file2.txt   lab_test      script.sh
 exercise.sh        file3.txt   output.txt   'script.sh\'
 file1.txt          hello.sh    Project       testfile.txt
Removing /home/lab_test directory ...
 auto_navigate.sh   file2.txt   output.txt   'script.sh\'
 exercise.sh        file3.txt   Project       testfile.txt
 file1.txt          hello.sh    script.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

Part 2: Automating File Operations

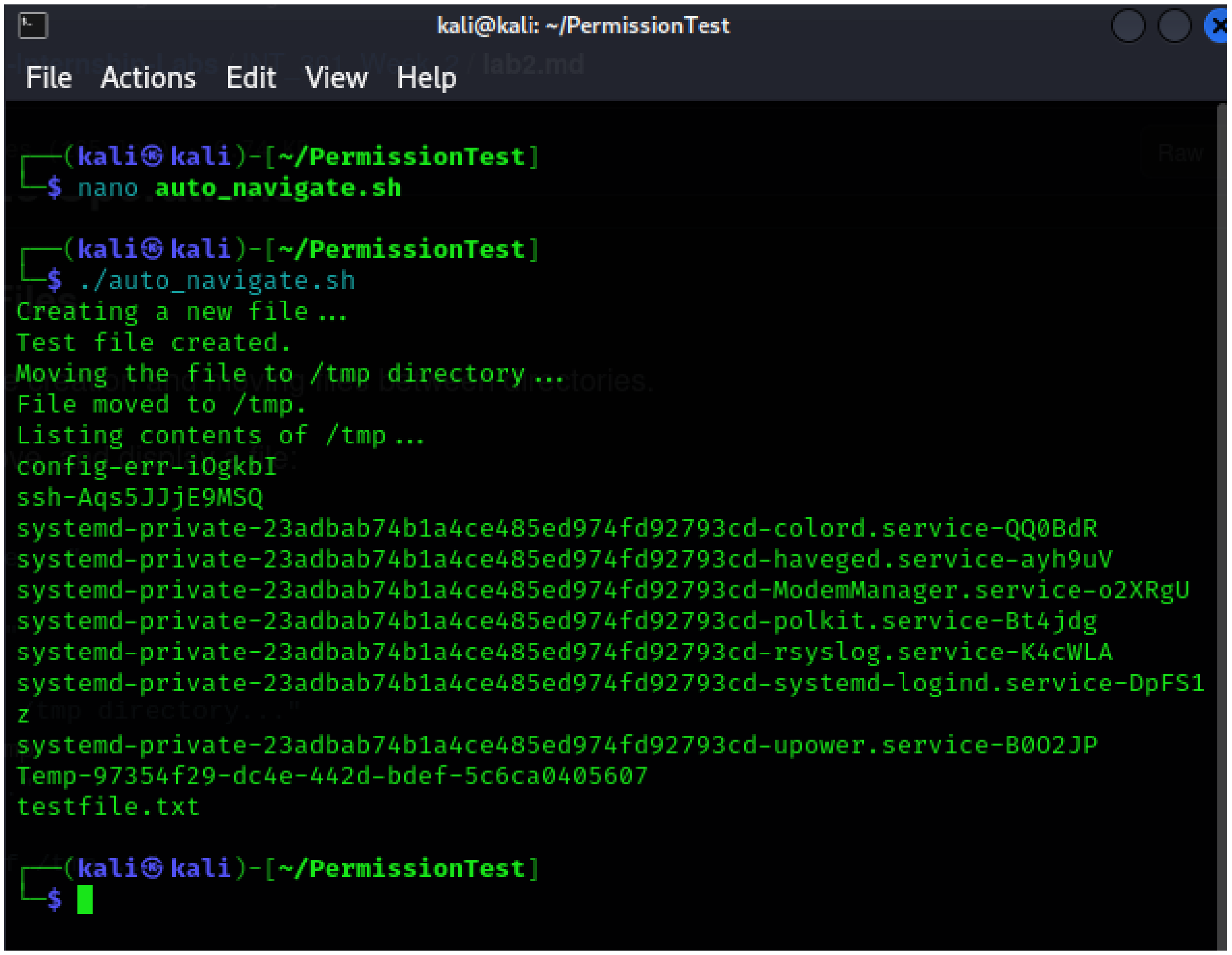
2.1 Creating and Moving Files

We will automate the process of file creation and moving files between directories.

1. Modify the script to create, move, and display a file:
2. echo "Creating a new file..."
3. touch /home/testfile.txt
4. echo "Test file created."
5.
6. echo "Moving the file to /tmp directory..."
7. mv /home/testfile.txt /tmp
8. echo "File moved to /tmp."
9.
10. echo "Listing contents of /tmp..."

ls /tmp

2. Save and rerun the script to see file creation and movement in action.



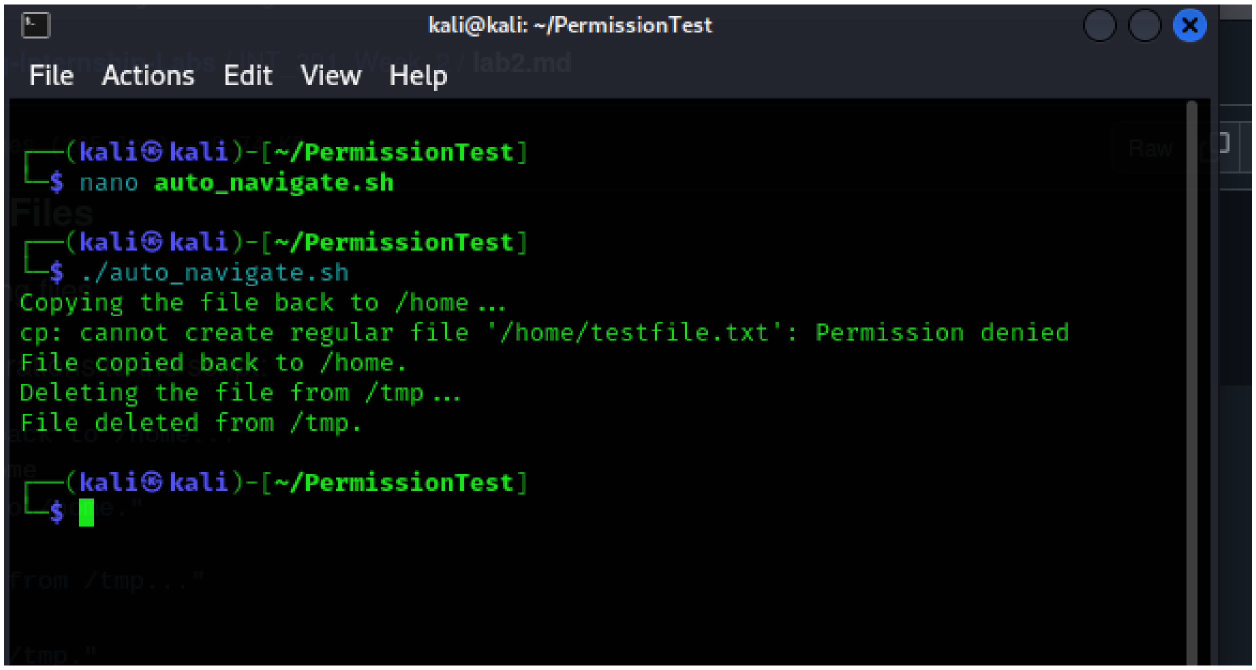
```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Creating a new file ...
Test file created.
Moving the file to /tmp directory ...
File moved to /tmp.
Listing contents of /tmp ...
config-err-iOgkbI
ssh-Aqs5JJjE9MSQ
systemd-private-23adbab74b1a4ce485ed974fd92793cd-colord.service-QQ0BdR
systemd-private-23adbab74b1a4ce485ed974fd92793cd-haveged.service-ayh9uV
systemd-private-23adbab74b1a4ce485ed974fd92793cd-ModemManager.service-o2XRgU
systemd-private-23adbab74b1a4ce485ed974fd92793cd-polkit.service-Bt4jdg
systemd-private-23adbab74b1a4ce485ed974fd92793cd-rsyslog.service-K4cWLA
systemd-private-23adbab74b1a4ce485ed974fd92793cd-systemd-logind.service-DpFS1
z
systemd-private-23adbab74b1a4ce485ed974fd92793cd-upower.service-B0O2JP
Temp-97354f29-dc4e-442d-bdef-5c6ca0405607
testfile.txt

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

2.2 Copying and Deleting Files

Let's automate copying and deleting files.

1. Add copying and deleting operations to the script:
2. echo "Copying the file back to /home..."
3. cp /tmp/testfile.txt /home
4. echo "File copied back to /home."
5.
6. echo "Deleting the file from /tmp..."
7. rm /tmp/testfile.txt

echo "File deleted from /tmp."



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Copying the file back to /home...
cp: cannot create regular file '/home/testfile.txt': Permission denied
File copied back to /home.
Deleting the file from /tmp...
File deleted from /tmp.

┌──(kali㉿kali)-[~/PermissionTest]
└─$ 
```

Part 3: Automating Recursive Directory Traversal

3.1 Using Loops to Navigate Multiple Directories

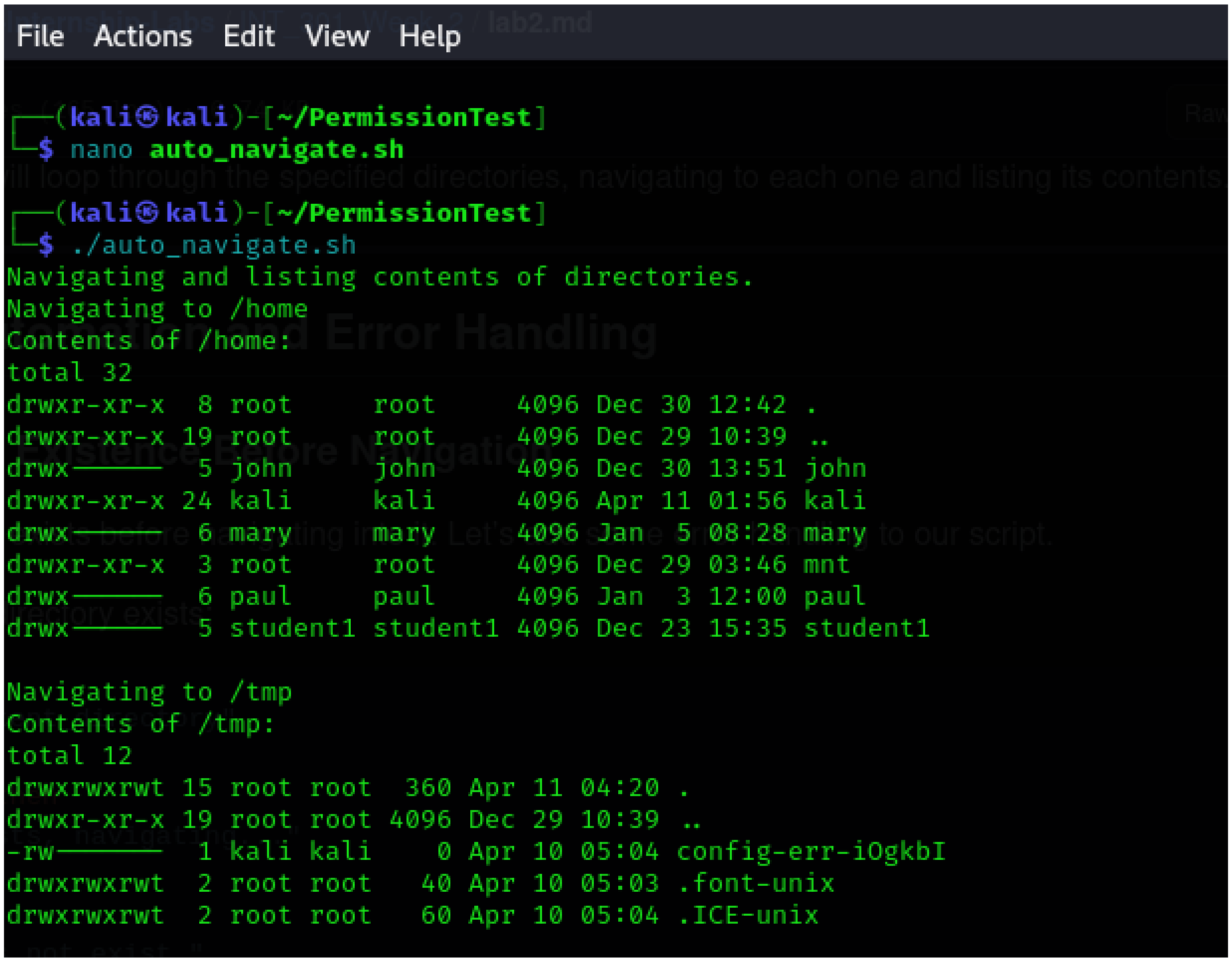
Sometimes, it is necessary to perform actions across multiple directories. A loop can help automate this process.

1. Create a bash script that navigates through several directories:
2. #!/bin/bash
3. echo "Navigating and listing contents of directories."
4.
5. directories=("/home" "/tmp" "/var")
6.
7. for dir in "${directories[@]}"; do
8.   echo "Navigating to $dir"
9.   cd $dir
10.   echo "Contents of $dir:"
11.   ls -la

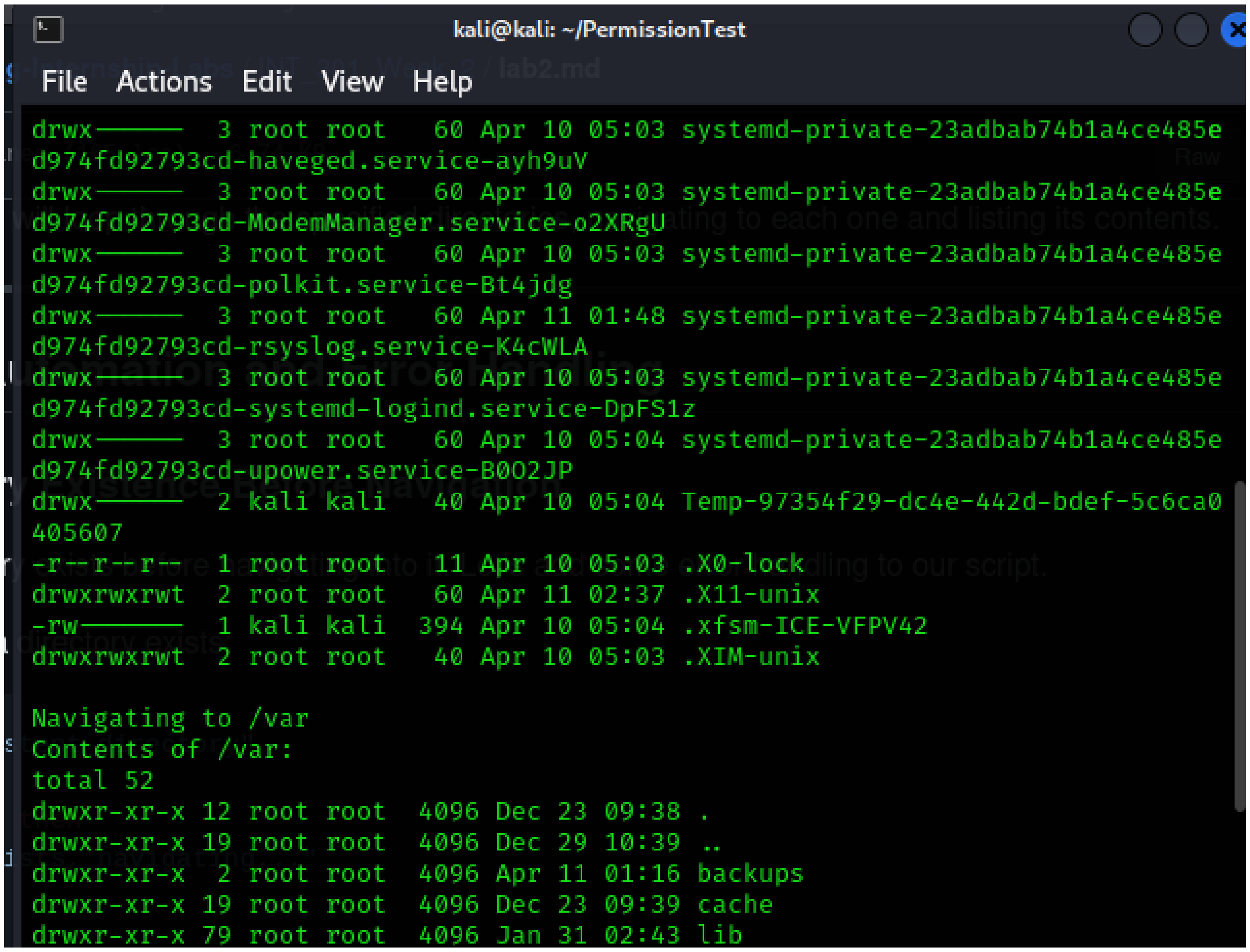
12. echo ""

done

2. Save and run the script. This will loop through the specified directories, navigating to each one and listing its contents.



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ nano auto_navigate.sh

┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Navigating and listing contents of directories.
Navigating to /home
Contents of /home:
total 32
drwxr-xr-x  8 root     root     4096 Dec 30 12:42 .
drwxr-xr-x 19 root     root     4096 Dec 29 10:39 ..
drwx------  5 john     john     4096 Dec 30 13:51 john
drwxr-xr-x 24 kali     kali     4096 Apr 11 01:56 kali
drwx------  6 mary     mary     4096 Jan  5 08:28 mary
drwxr-xr-x  3 root     root     4096 Dec 29 03:46 mnt
drwx------  6 paul     paul     4096 Jan  3 12:00 paul
drwx------  5 student1 student1 4096 Dec 23 15:35 student1

Navigating to /tmp
Contents of /tmp:
total 12
drwxrwxrwt 15 root root  360 Apr 11 04:20 .
drwxr-xr-x 19 root root 4096 Dec 29 10:39 ..
-rw-------  1 kali kali    0 Apr 10 05:04 config-err-iOgkbI
drwxrwxrwt  2 root root   40 Apr 10 05:03 .font-unix
drwxrwxrwt  2 root root   60 Apr 10 05:04 .ICE-unix
```



```
drwx------  3 root root   60 Apr 10 05:03 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-haveged.service-ayh9uV
drwx------  3 root root   60 Apr 10 05:03 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-ModemManager.service-o2XRgU
drwx------  3 root root   60 Apr 10 05:03 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-polkit.service-Bt4jdg
drwx------  3 root root   60 Apr 11 01:48 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-rsyslog.service-K4cWLA
drwx------  3 root root   60 Apr 10 05:03 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-systemd-logind.service-DpFS1z
drwx------  3 root root   60 Apr 10 05:04 systemd-private-23adbab74b1a4ce485e
d974fd92793cd-upower.service-B0O2JP
drwx------  2 kali kali   40 Apr 10 05:04 Temp-97354f29-dc4e-442d-bdef-5c6ca0
405607
-r--r--r--  1 root root   11 Apr 10 05:03 .X0-lock
drwxrwxrwt  2 root root   60 Apr 11 02:37 .X11-unix
-rw-------  1 kali kali  394 Apr 10 05:04 .xfsm-ICE-VFPV42
drwxrwxrwt  2 root root   40 Apr 10 05:03 .XIM-unix

Navigating to /var
Contents of /var:
total 52
drwxr-xr-x 12 root root  4096 Dec 23 09:38 .
drwxr-xr-x 19 root root  4096 Dec 29 10:39 ..
drwxr-xr-x  2 root root  4096 Apr 11 01:16 backups
drwxr-xr-x 19 root root  4096 Dec 23 09:39 cache
drwxr-xr-x 79 root root  4096 Jan 31 02:43 lib
```

Part 4: Conditional Automation and Error Handling

4.1 Checking for Directory Existence Before Navigation

It's important to check if a directory exists before navigating into it. Let's add some error handling to our script.

1. Modify the script to check if a directory exists:
2. directory="/home/nonexistent_directory"
3.
4. if [-d "$directory"]; then
5.   echo "$directory exists, navigating..."
6.   cd $directory
7. else

8.   echo "$directory does not exist."

fi

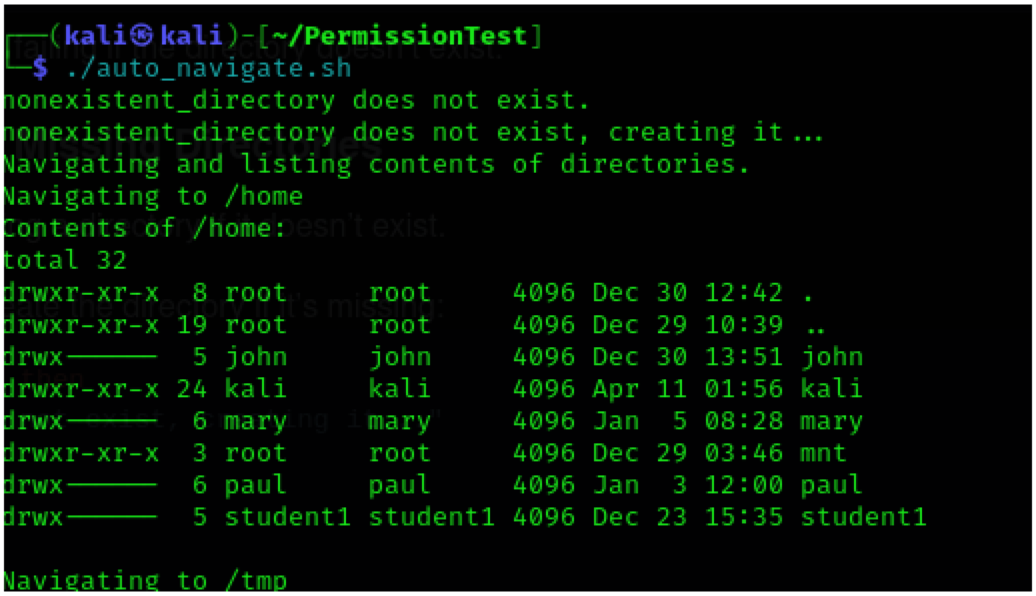
2. This will prevent the script from failing if the directory doesn't exist.

4.2 Automatically Creating Missing Directories

Let's automate the process of creating a directory if it doesn't exist.

1. Modify the previous script to create the directory if it's missing:
2. if [! -d "$directory"]; then
3.   echo "$directory does not exist, creating it..."
4.   mkdir $directory
5. fi

cd $directory



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
nonexistent_directory does not exist.
nonexistent_directory does not exist, creating it ...
Navigating and listing contents of directories.
Navigating to /home
Contents of /home:
total 32
drwxr-xr-x  8 root     root     4096 Dec 30 12:42 .
drwxr-xr-x 19 root     root     4096 Dec 29 10:39 ..
drwx──────  5 john     john     4096 Dec 30 13:51 john
drwxr-xr-x 24 kali     kali     4096 Apr 11 01:56 kali
drwx──────  6 mary     mary     4096 Jan  5 08:28 mary
drwxr-xr-x  3 root     root     4096 Dec 29 03:46 mnt
drwx──────  6 paul     paul     4096 Jan  3 12:00 paul
drwx──────  5 student1 student1 4096 Dec 23 15:35 student1

Navigating to /tmp
```

Part 5: Automating Cleanup and Archiving

5.1 Deleting Old Files Automatically

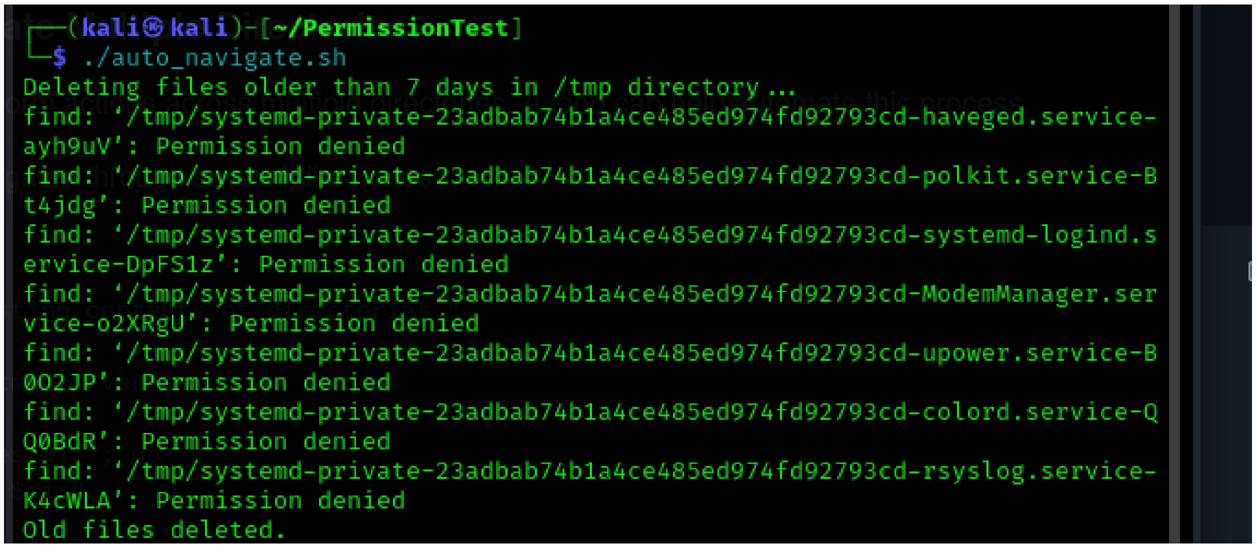
Automate the process of deleting files older than a specified number of days (e.g., 7 days).

1. Add this code to your script:
2. echo "Deleting files older than 7 days in /tmp directory..."

3. find /tmp -type f -mtime +7 -exec rm {} \;

echo "Old files deleted."

This will find and delete any files in /tmp that are older than 7 days.



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Deleting files older than 7 days in /tmp directory ...
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-haveged.service-
ayh9uV': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-polkit.service-B
t4jdg': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-systemd-logind.s
ervice-DpFS1z': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-ModemManager.ser
vice-o2XRgU': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-upower.service-B
0O2JP': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-colord.service-Q
Q0BdR': Permission denied
find: '/tmp/systemd-private-23adbab74b1a4ce485ed974fd92793cd-rsyslog.service-
K4cWLA': Permission denied
Old files deleted.
```

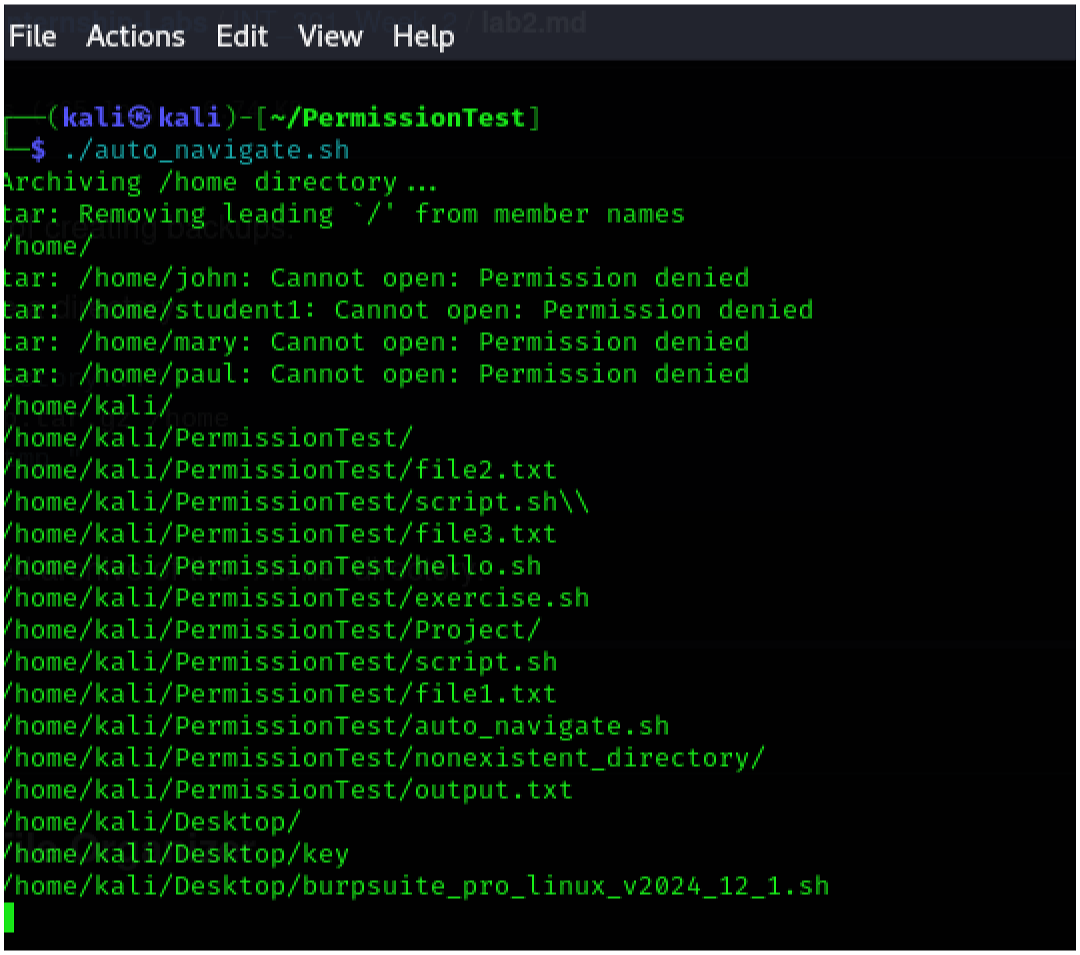
5.2 Archiving Directories

You can also automate the process of creating backups.

1. Add this to your script to archive a directory:
2. echo "Archiving /home directory..."
3. tar -czvf /tmp/home_backup.tar.gz /home

echo "Backup created in /tmp."

This command creates a compressed archive of the /home directory.



```
┌──(kali㉿kali)-[~/PermissionTest]
└─$ ./auto_navigate.sh
Archiving /home directory...
tar: Removing leading `/' from member names
/home/
tar: /home/john: Cannot open: Permission denied
tar: /home/student1: Cannot open: Permission denied
tar: /home/mary: Cannot open: Permission denied
tar: /home/paul: Cannot open: Permission denied
/home/kali/
/home/kali/PermissionTest/
/home/kali/PermissionTest/file2.txt
/home/kali/PermissionTest/script.sh\\
/home/kali/PermissionTest/file3.txt
/home/kali/PermissionTest/hello.sh
/home/kali/PermissionTest/exercise.sh
/home/kali/PermissionTest/Project/
/home/kali/PermissionTest/script.sh
/home/kali/PermissionTest/file1.txt
/home/kali/PermissionTest/auto_navigate.sh
/home/kali/PermissionTest/nonexistent_directory/
/home/kali/PermissionTest/output.txt
/home/kali/Desktop/
/home/kali/Desktop/key
/home/kali/Desktop/burpsuite_pro_linux_v2024_12_1.sh
```

Part 6: Exercises

Exercise 1: Directory and File Organizer

Write a bash script that:

1. Creates directories for the days of the week (Monday, Tuesday, etc.).
2. Moves files from /home that were created on each day of the week into the corresponding directory.

Hints:

- Use the date command to determine when a file was created.
- Use loops to automate the process.

Exercise 2: Automated Log Cleanup

Write a script that:

1. Navigates to the /var/log directory.
2. Deletes all .log files that are older than 5 days.
3. Archives the remaining .log files into a tar archive.

Exercise 3: Directory Search Script

Write a script that:

1. Takes a directory name as an argument.
2. Searches for that directory on the system and outputs its full path.
3. If the directory does not exist, it should prompt the user to create it.

Exercise 4: Automated Backup System

Write a bash script that:

1. Creates a daily backup of the /home directory.
2. The script should create a separate archive for each day of the week (e.g., home_backup_Monday.tar.gz).
3. It should check if a backup for the current day exists, and if so, overwrite it.

---

4. Lab 3: Automating Advanced File Management and Permissions in Linux

Part 1: Automating File Permissions

1.1 Viewing and Modifying Permissions

We will begin by scripting the process of viewing and modifying file permissions.

1. Open a terminal and create a bash script:

nano manage_permissions.sh

In the script, write the following commands to display the permissions of a file:

#!/bin/bash

# Script to manage file permissions

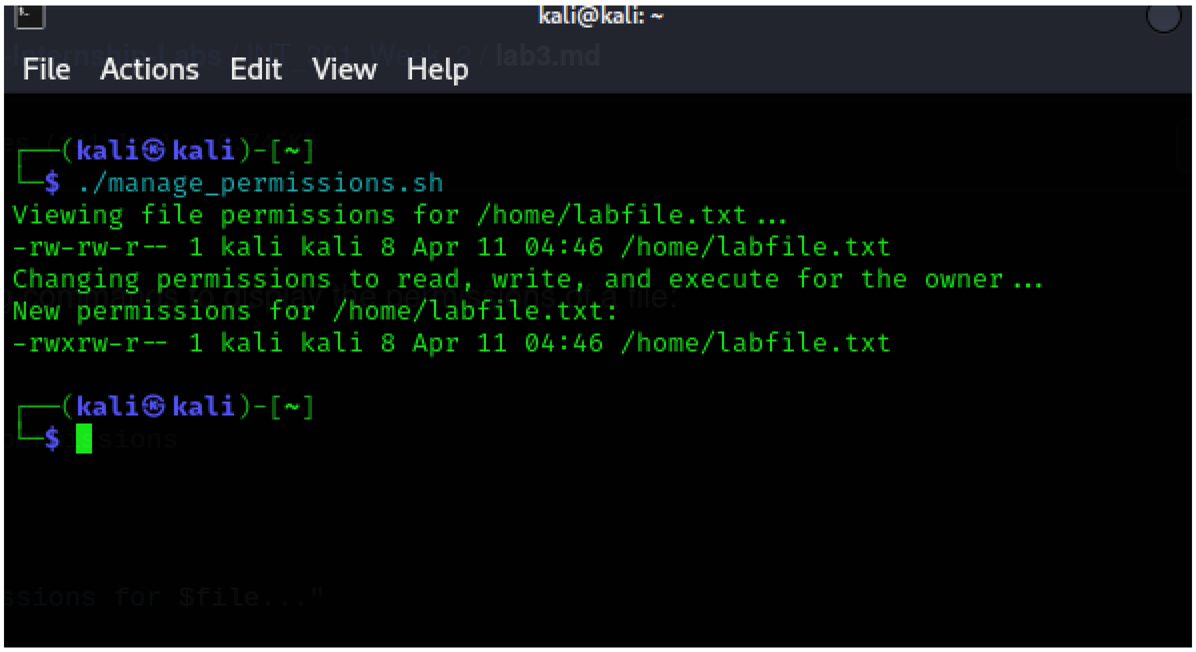
```
file="/home/labfile.txt"


echo "Viewing file permissions for $file..."

ls -l $file


echo "Changing permissions to read, write, and execute for the owner..."

chmod u+rwx $file


echo "New permissions for $file:"

ls -l $file
```

Save and make the script executable:

```
chmod +x manage_permissions.sh
```

Run the script to test permission changes:

```
./manage_permissions.sh
```



```
┌──(kali㉿kali)-[~]
└─$ ./manage_permissions.sh
Viewing file permissions for /home/labfile.txt ...
-rw-rw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
Changing permissions to read, write, and execute for the owner ...
New permissions for /home/labfile.txt:
-rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt

┌──(kali㉿kali)-[~]
└─$ 
```

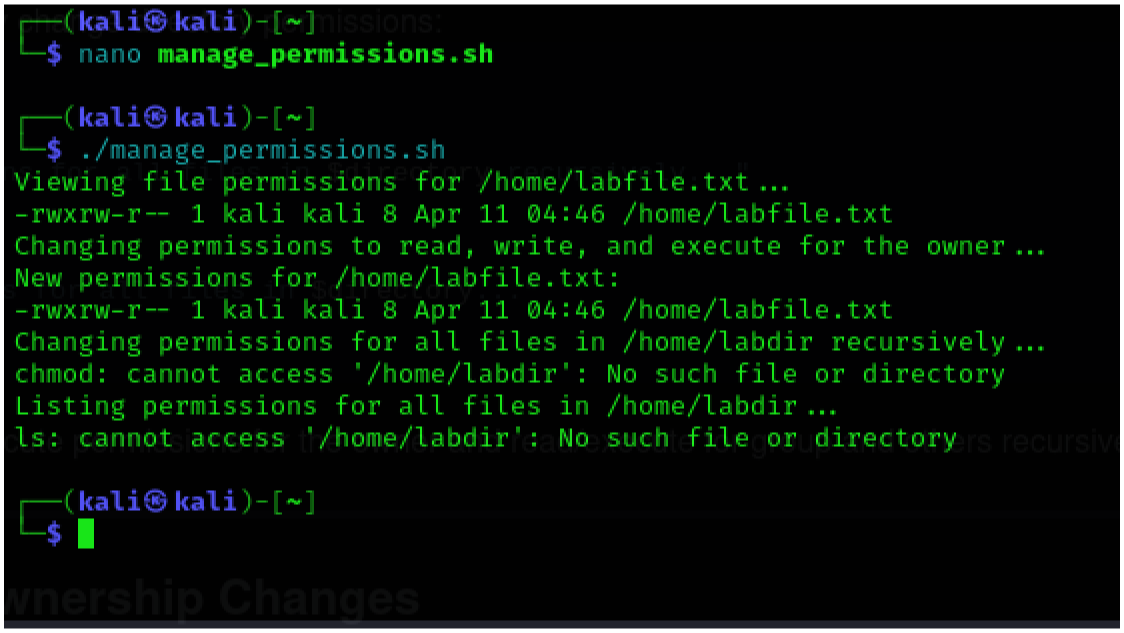
1.2 Automating Recursive Permission Changes

Permissions often need to be set recursively, especially for directories. Let’s automate this.

1. Modify the script to recursively change directory permissions:
2. directory="/home/labdir"
3.
4. echo "Changing permissions for all files in $directory recursively..."
5. chmod -R 755 $directory
6.
7. echo "Listing permissions for all files in $directory..."

ls -l $directory

This will apply read, write, and execute permissions for the owner and read/execute for group and others recursively.



```
┌──(kali㉿kali)-[~]
└─$ nano manage_permissions.sh

┌──(kali㉿kali)-[~]
└─$ ./manage_permissions.sh
Viewing file permissions for /home/labfile.txt...
-rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
Changing permissions to read, write, and execute for the owner...
New permissions for /home/labfile.txt:
-rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
Changing permissions for all files in /home/labdir recursively...
chmod: cannot access '/home/labdir': No such file or directory
Listing permissions for all files in /home/labdir...
ls: cannot access '/home/labdir': No such file or directory

┌──(kali㉿kali)-[~]
└─$ 
```

Part 2: Automating Ownership Changes

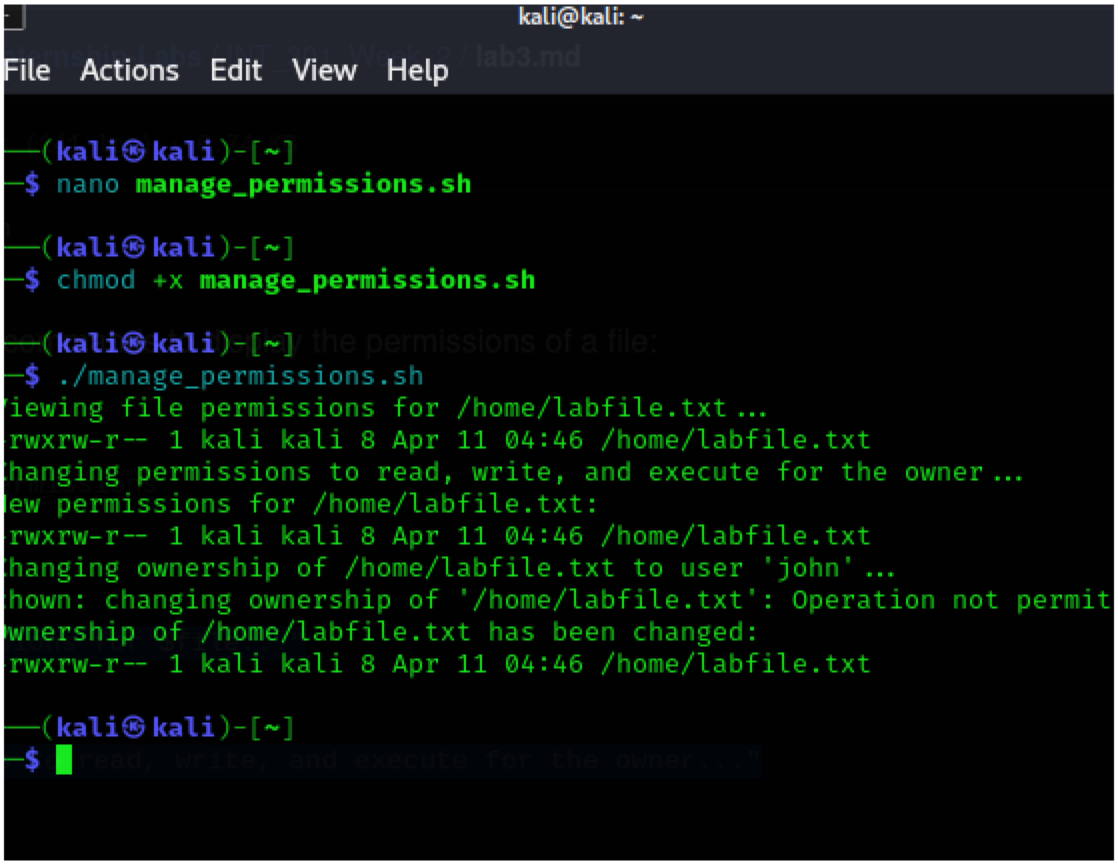
2.1 Changing File Ownership

We will now script ownership changes for files and directories.

1. Add to your script to change ownership of a file:
2. echo "Changing ownership of $file to user 'john'..."
3. chown john $file
4.
5. echo "Ownership of $file has been changed:"

ls -l $file

2. Run the script and ensure the ownership is updated.



```
──(kali㉿kali)-[~]
─$ nano manage_permissions.sh

──(kali㉿kali)-[~]
─$ chmod +x manage_permissions.sh

──(kali㉿kali)-[~]
─$ ./manage_permissions.sh
iewing file permissions for /home/labfile.txt ...
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
hanging permissions to read, write, and execute for the owner ...
ew permissions for /home/labfile.txt:
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
hanging ownership of /home/labfile.txt to user 'john' ...
hown: changing ownership of '/home/labfile.txt': Operation not permit
wnership of /home/labfile.txt has been changed:
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt

──(kali㉿kali)-[~]
─$ 
```

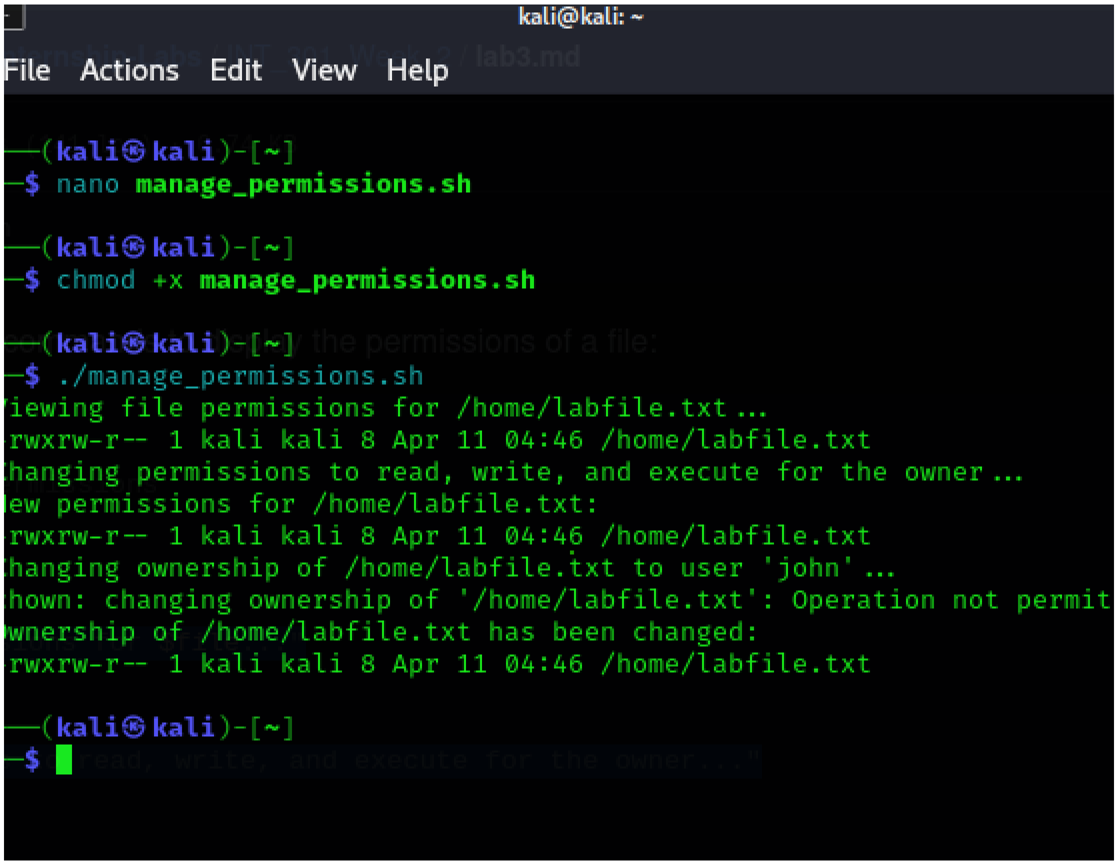
2.2 Recursive Ownership Changes

Ownership changes can also be applied recursively. Here's how to automate that:

1. Modify the script to change ownership recursively:
2. echo "Changing ownership of all files in $directory to user 'john'..."
3. chown -R john $directory
4.
5. echo "Listing ownership for all files in $directory..."

ls -l $directory

This will change ownership of all files and directories within /home/labdir.



```
──(kali㉿kali)-[~]
─$ nano manage_permissions.sh

──(kali㉿kali)-[~]
─$ chmod +x manage_permissions.sh

──(kali㉿kali)-[~]
─$ ./manage_permissions.sh
iewing file permissions for /home/labfile.txt ...
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
hanging permissions to read, write, and execute for the owner ...
ew permissions for /home/labfile.txt:
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt
hanging ownership of /home/labfile.txt to user 'john' ...
hown: changing ownership of '/home/labfile.txt': Operation not permit
wnership of /home/labfile.txt has been changed:
rwxrw-r-- 1 kali kali 8 Apr 11 04:46 /home/labfile.txt

──(kali㉿kali)-[~]
─$ 
```

Part 3: Automating Permission Errors and Conditional Handling

3.1 Handling Permission Errors

Sometimes, permission or ownership changes may fail. Let’s add error handling to our script.

1. Modify your script to handle permission errors:
2. if [-e $file]; then
3.   chmod u+rwx $file
4.   echo "Permissions changed successfully."
5. else
6.   echo "Error: $file does not exist!"

fi

This script checks if the file exists before attempting to change permissions and outputs an error if the file is not found.

Part 4: Automating Access Control Lists (ACLs)

4.1 Managing File ACLs

Access control lists (ACLs) allow more granular permission control beyond the traditional user/group/other model.

1. Add the following to your script to apply ACLs to a file:
2. echo "Applying ACL for user 'john' to read the file..."
3. setfacl -m u:john:r /home/labfile.txt
4.
5. echo "Viewing ACL for $file..."

getfacl /home/labfile.txt

2. Save and run the script. This will grant user john read access to labfile.txt.

4.2 Removing ACLs

You can also automate the removal of ACLs:

1. Modify the script to remove ACLs:
2. echo "Removing ACL for user 'john'..."
3. setfacl -x u:john /home/labfile.txt
4.
5. echo "ACL removed. Current ACL for $file:"

getfacl /home/labfile.txt

Part 5: Automating File Archiving and Compression with Permissions

5.1 Retaining Permissions in Archives

It’s important to retain permissions and ownership information when archiving files. Let’s automate the process of archiving while keeping permissions intact.

1. Add this code to your script to archive and compress a directory:
2. echo "Archiving and compressing the /home/labdir directory..."
3. tar -czvf /tmp/labdir_backup.tar.gz /home/labdir --preserve-permissions
4.

echo "Backup created with preserved permissions."

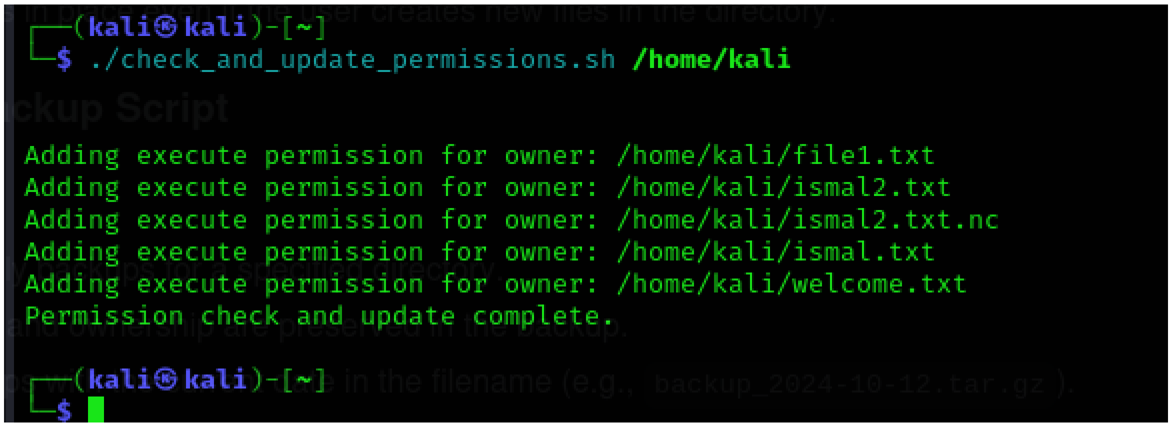
2. This command ensures that when the /home/labdir directory is restored from the backup, the permissions and ownership settings are retained.

Part 6: Exercises

Exercise 1: Automated Permission Checker

Write a bash script that:

1. Takes a directory as an input.
2. Checks the permissions of all files in the directory.
3. If any file is missing execute permissions for the owner, the script should automatically apply u+x.

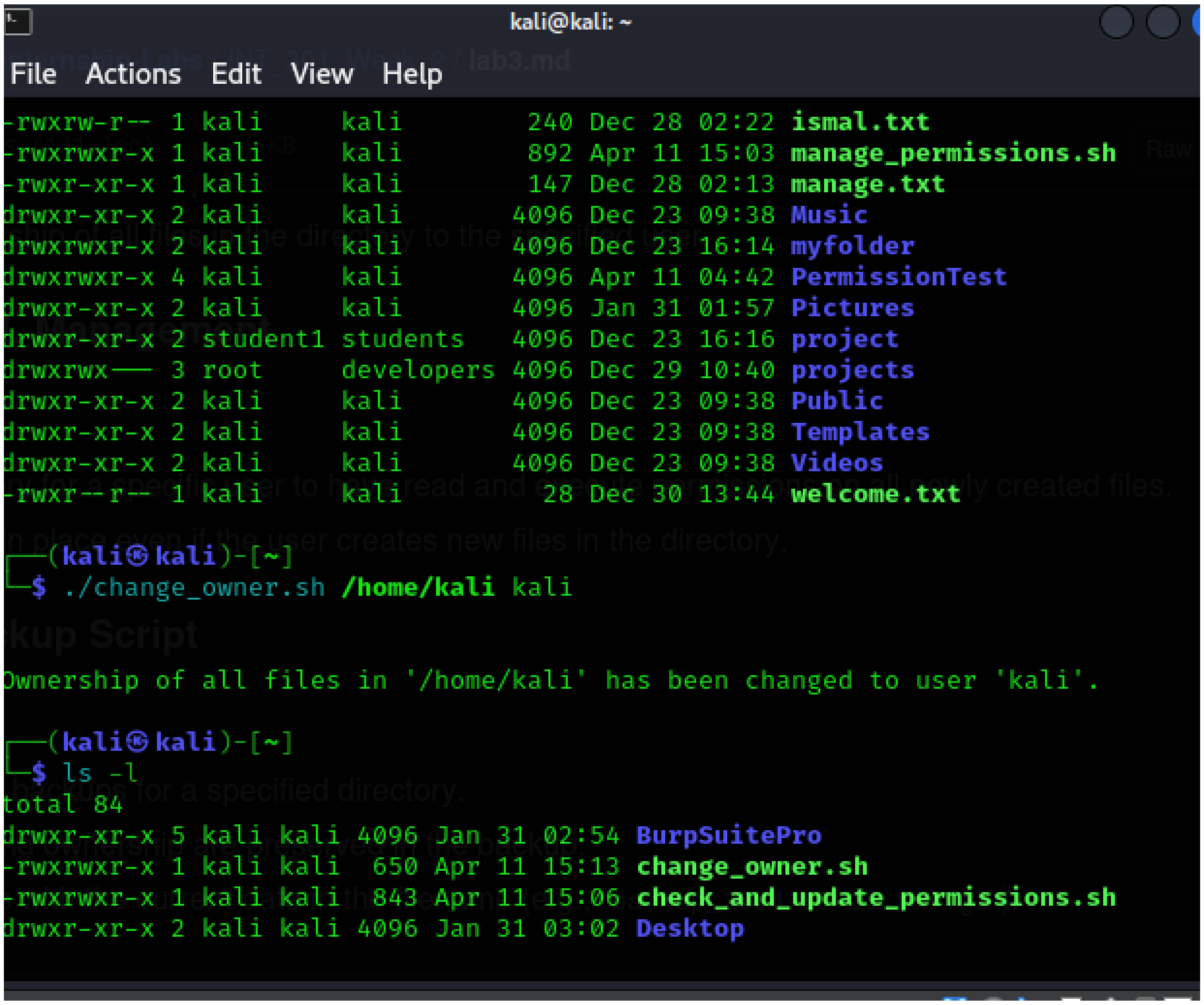


```
┌──(kali㉿kali)-[~]
└─$ ./check_and_update_permissions.sh /home/kali

Adding execute permission for owner: /home/kali/file1.txt
Adding execute permission for owner: /home/kali/ismal2.txt
Adding execute permission for owner: /home/kali/ismal2.txt.nc
Adding execute permission for owner: /home/kali/ismal.txt
Adding execute permission for owner: /home/kali/welcome.txt
Permission check and update complete.

┌──(kali㉿kali)-[~]
└─$ 
```

Exercise 2: Bulk Ownership Change

Write a bash script that:

1. Accepts a directory path and username as input.
2. Recursively changes the ownership of all files in the directory to the specified user.

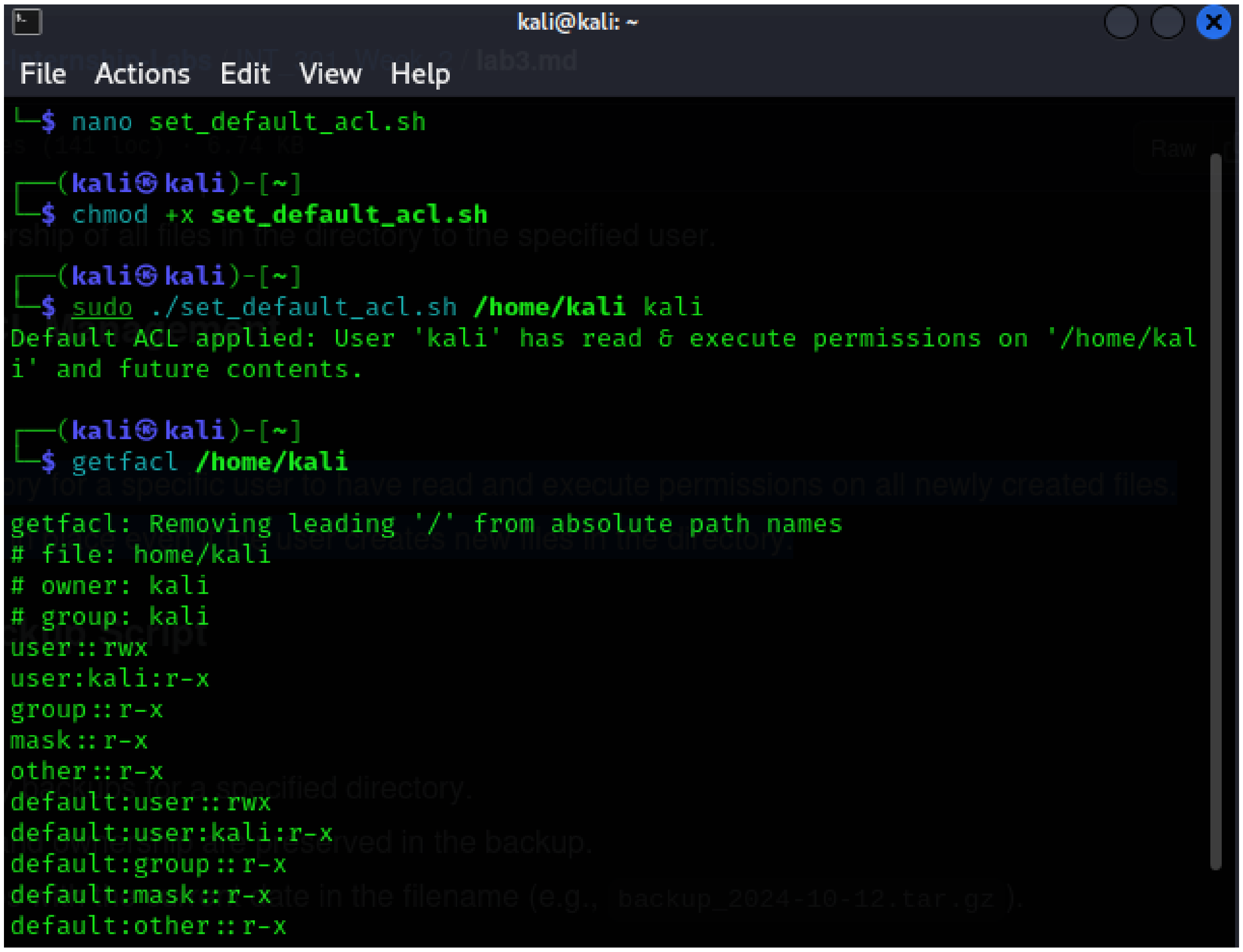


```
-rwxrw-r-- 1 kali     kali        240 Dec 28 02:22 ismal.txt
-rwxrwxr-x 1 kali     kali        892 Apr 11 15:03 manage_permissions.sh
-rwxr-xr-x 1 kali     kali        147 Dec 28 02:13 manage.txt
drwxr-xr-x 2 kali     kali       4096 Dec 23 09:38 Music
drwxrwxr-x 2 kali     kali       4096 Dec 23 16:14 myfolder
drwxrwxr-x 4 kali     kali       4096 Apr 11 04:42 PermissionTest
drwxr-xr-x 2 kali     kali       4096 Jan 31 01:57 Pictures
drwxr-xr-x 2 student1 students   4096 Dec 23 16:16 project
drwxrwx--- 3 root     developers 4096 Dec 29 10:40 projects
drwxr-xr-x 2 kali     kali       4096 Dec 23 09:38 Public
drwxr-xr-x 2 kali     kali       4096 Dec 23 09:38 Templates
drwxr-xr-x 2 kali     kali       4096 Dec 23 09:38 Videos
-rwxr--r-- 1 kali     kali         28 Dec 30 13:44 welcome.txt

┌──(kali㉿kali)-[~]
└─$ ./change_owner.sh /home/kali kali

Ownership of all files in '/home/kali' has been changed to user 'kali'.

┌──(kali㉿kali)-[~]
└─$ ls -l
total 84
drwxr-xr-x 5 kali kali 4096 Jan 31 02:54 BurpSuitePro
-rwxrwxr-x 1 kali kali  650 Apr 11 15:13 change_owner.sh
-rwxrwxr-x 1 kali kali  843 Apr 11 15:06 check_and_update_permissions.sh
drwxr-xr-x 2 kali kali 4096 Jan 31 03:02 Desktop
```

Exercise 3: Automated ACL Management

Write a script that:

1. Sets a default ACL on a directory for a specific user to have read and execute permissions on all newly created files.
2. Ensures that the ACL remains in place even if the user creates new files in the directory.

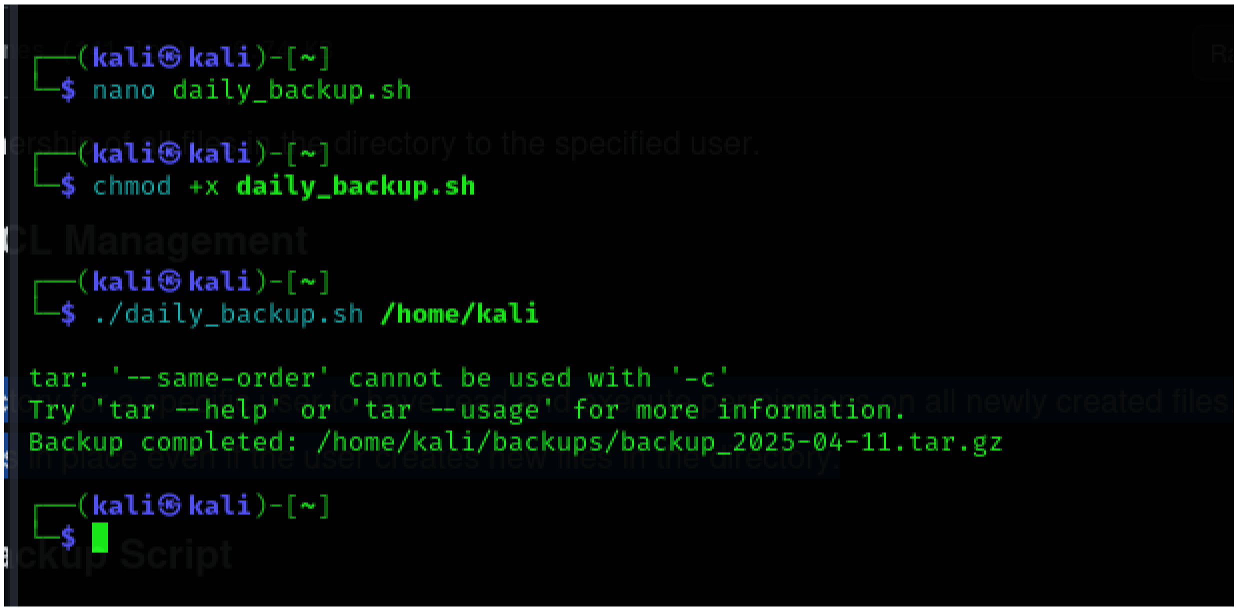


```
└─$ nano set_default_acl.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x set_default_acl.sh

┌──(kali㉿kali)-[~]
└─$ sudo ./set_default_acl.sh /home/kali kali
Default ACL applied: User 'kali' has read & execute permissions on '/home/kal
i' and future contents.

┌──(kali㉿kali)-[~]
└─$ getfacl /home/kali

getfacl: Removing leading '/' from absolute path names
# file: home/kali
# owner: kali
# group: kali
user::rwx
user:kali:r-x
group::r-x
mask::r-x
other::r-x
default:user::rwx
default:user:kali:r-x
default:group::r-x
default:mask::r-x
default:other::r-x
```

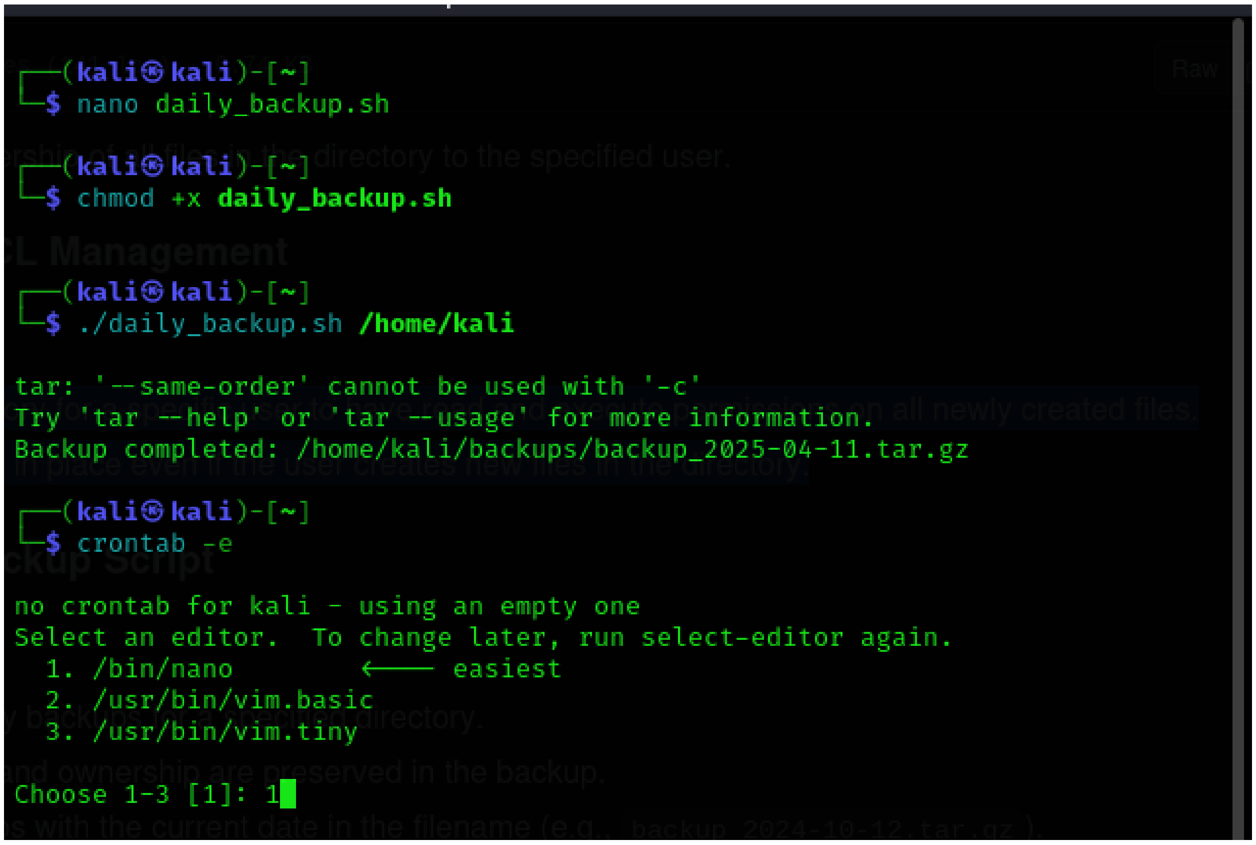
Exercise 4: Scheduled Backup Script

Write a script that:

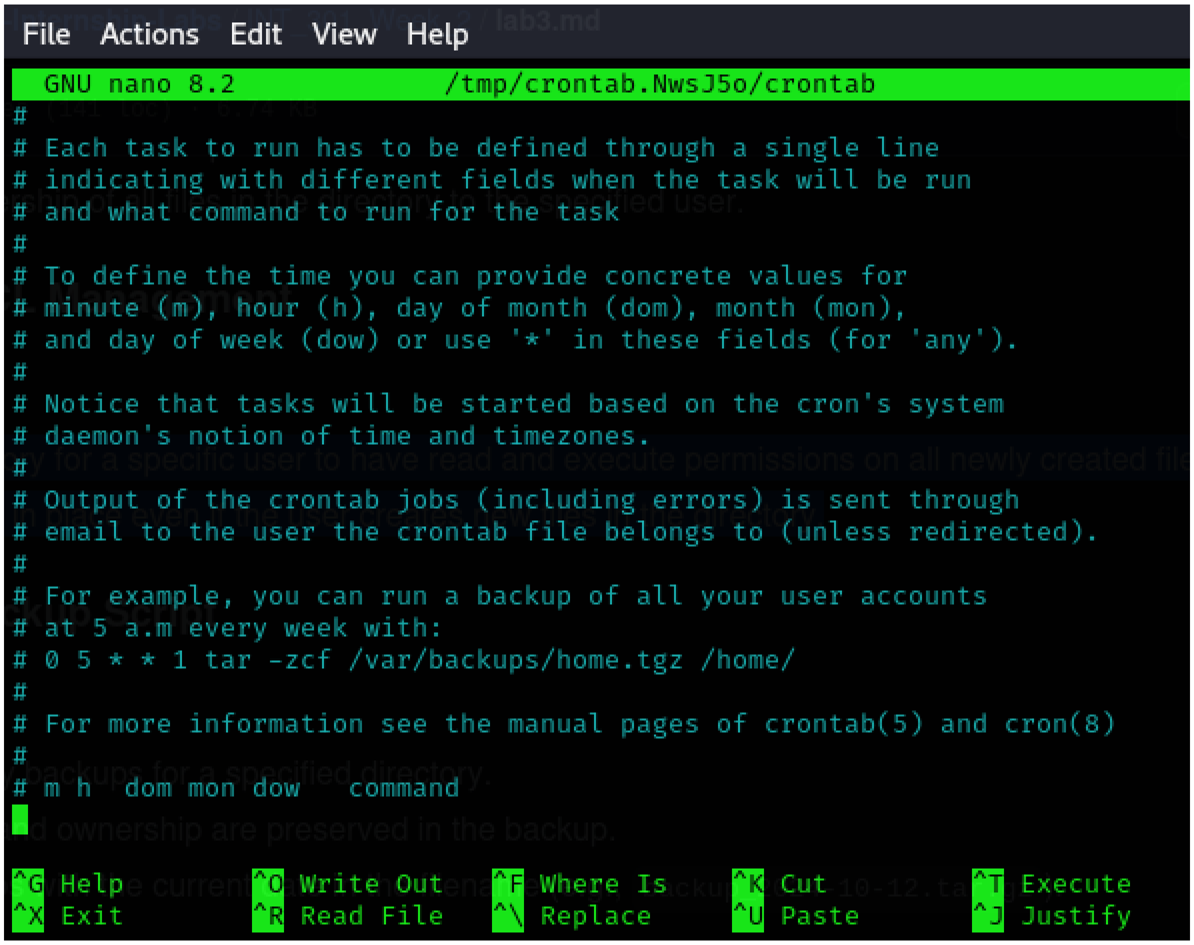
1. Automates the creation of daily backups for a specified directory.
2. Ensures that file permissions and ownership are preserved in the backup.
3. The script should save backups with the current date in the filename (e.g., backup_2024-10-12.tar.gz).



```
┌──(kali㉿kali)-[~]
└─$ nano daily_backup.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x daily_backup.sh

┌──(kali㉿kali)-[~]
└─$ ./daily_backup.sh /home/kali

tar: '--same-order' cannot be used with '-c'
Try 'tar --help' or 'tar --usage' for more information.
Backup completed: /home/kali/backups/backup_2025-04-11.tar.gz

┌──(kali㉿kali)-[~]
└─$ 
```



```
┌──(kali㉿kali)-[~]
└─$ nano daily_backup.sh

┌──(kali㉿kali)-[~]
└─$ chmod +x daily_backup.sh

┌──(kali㉿kali)-[~]
└─$ ./daily_backup.sh /home/kali

tar: '--same-order' cannot be used with '-c'
Try 'tar --help' or 'tar --usage' for more information.
Backup completed: /home/kali/backups/backup_2025-04-11.tar.gz

┌──(kali㉿kali)-[~]
└─$ crontab -e

no crontab for kali - using an empty one
Select an editor.  To change later, run select-editor again.
  1. /bin/nano        <---- easiest
  2. /usr/bin/vim.basic
  3. /usr/bin/vim.tiny

Choose 1-3 [1]: 1
```



```
  GNU nano 8.2              /tmp/crontab.NwsJ5o/crontab
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify
```