



CHRIST

(DEEMED TO BE UNIVERSITY)

B A N G A L O R E • I N D I A

SCHOOL OF ENGINEERING AND TECHNOLOGY

NAME: EVAN K S

REGISTER NUMBER: 2462067

CLASS: 3 BTCSAIML B

COURSE CODE: CSHO331CSP

ETHICAL HACKING

Assignment 5: File Creation and Permission REPORT

**AIM : Create a file and set specific permissions using numbers
(751)**

Introduction

This document outlines the process of creating a file and assigning it specific permissions using **octal notation** in a Linux operating system. The objective is to create a file and set its permissions to the octal value 751, demonstrating a clear understanding of file access control for the owner, group, and others.

Methodology

The process involves three main steps: creating the file, setting the permissions, and verifying the result.

1. File Creation

A new, empty file named myfile.txt is created using the touch command.

```
touch myfile.txt
```

2. Setting File Permissions

Permissions are assigned using the chmod command with the octal value 751.

chmod 751 myfile.txt This sets the permissions as follows:

Owner (7): Read, Write, Execute (rwx)

Group (5): Read, Execute (r-x)

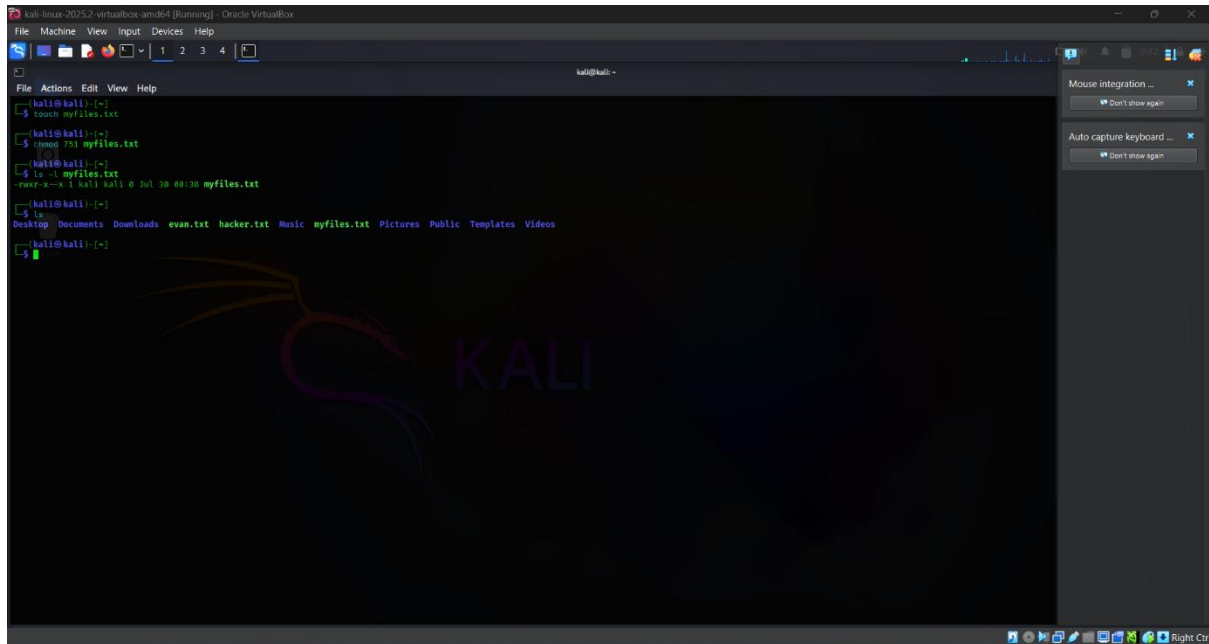
Others (1): Execute only (--x)

3. Verification

To verify the new file permissions, the `ls -l` command is used.

`ls -l myfile.txt` The expected output will be similar to this:

```
-rwxr-x--x 1 user user 0 Aug 1 10:30 myfile.txt
```

A screenshot of a Kali Linux terminal window. The terminal shows the following commands and output:

```
kali@kali:~$ touch myfile.txt
kali@kali:~$ chmod 751 myfile.txt
kali@kali:~$ ls -l myfile.txt
-rwxr-x--x 1 kali kali 0 Jul 30 00:30 myfile.txt
```

The terminal window has a dark background with a Kali Linux logo. The file manager sidebar on the right shows the file 'myfile.txt' in the 'Downloads' directory.

The file is created and permission is set using numbers (751).

Findings & Conclusion

The file `myfile.txt` was created successfully.

Permissions 751 were correctly applied.

The `ls -l` output confirmed that the owner has full access, the group can read and execute, and others can only execute.

This permission scheme is ideal when all users must be able to run a file, but only the owner should have full control. This task successfully demonstrated the use of octal notation to manage file permissions securely in a Linux environment.

Code Submission

Here is the complete script for the operations:

Create the file

```
touch myfile.txt
```

Set permissions

```
chmod 751 myfile.txt
```

Verify permissions

```
ls -l myfile.txt
```

Notes: Octal Notation in File Permissions

In Linux, file permissions are controlled using **octal notation**, a shorthand based on the **base-8 number system (0–7)**. This notation makes it easier to represent read (r), write (w), and execute (x) permissions for:

- **Owner**
- **Group**
- **Others**

Each permission has a numeric value:

Permission	Symbol	Value
Read	r	4
Write	w	2
Execute	x	1

You add the values to get the octal digit:

Symbolic	Binary	Octal	Meaning
rwX	111	7	Read, write, execute
rw-	110	6	Read, write
r-X	101	5	Read, execute
r--	100	4	Read only
--X	001	1	Execute only

Example: `chmod 751 myfile.txt`

This sets permissions as:

Role	Octal	Symbolic	Description
Owner	7	rwX	Read, write, execute
Group	5	r-X	Read and execute only
Others	1	--X	Execute only

When you run:

```
ls -l myfile.txt
```

You'll see:

```
-rwxr-x--x
```

This **octal format** helps administrators manage file access clearly and securely.

Conclusion

This assignment demonstrated how to create a file and set permissions using octal notation in Linux. By applying `chmod 751`, we ensured that the file owner has full control (read, write, execute), the group can only read and execute, and others can execute only. The verification step using `ls -l` confirmed that the permissions were applied correctly. This approach is efficient, secure, and provides a clear, numeric method for managing file access, making it particularly useful for situations where precise control over file usage is required.