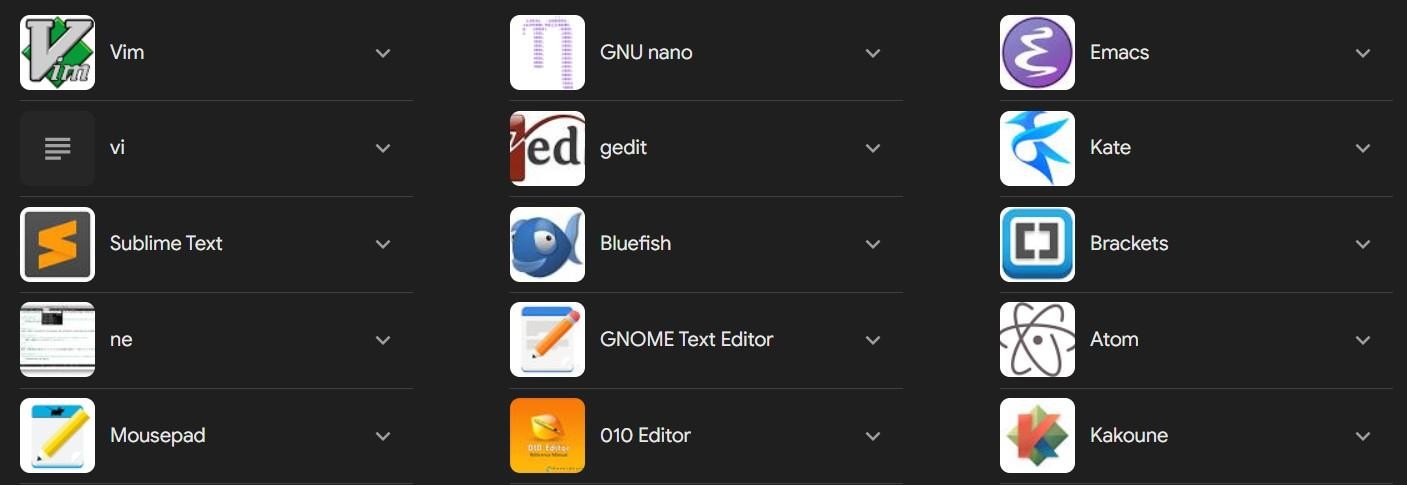
# Experiment – 5

## Linux Editors with Command Specification

### Introduction

Linux offers a diverse range of text editors that cater to different users' needs. Whether you are writing code, editing configuration files, or creating simple text documents, these editors provide robust functionality for text manipulation. In this assignment, we explore some of the most widely used Linux text editors, focusing on their command specifications and the unique features that set them apart.



### Vim (Vi Improved)

Vim is an advanced text editor built for programmers, based on the vi editor, with powerful features for efficient text manipulation. Vim supports a modal interface, where users switch between different modes for navigation, insertion, and command execution.

### Command Specifications:

* + **Modes:** o Normal mode (default): Use for navigating and manipulating text.
    - Insert mode: Press i to insert text.

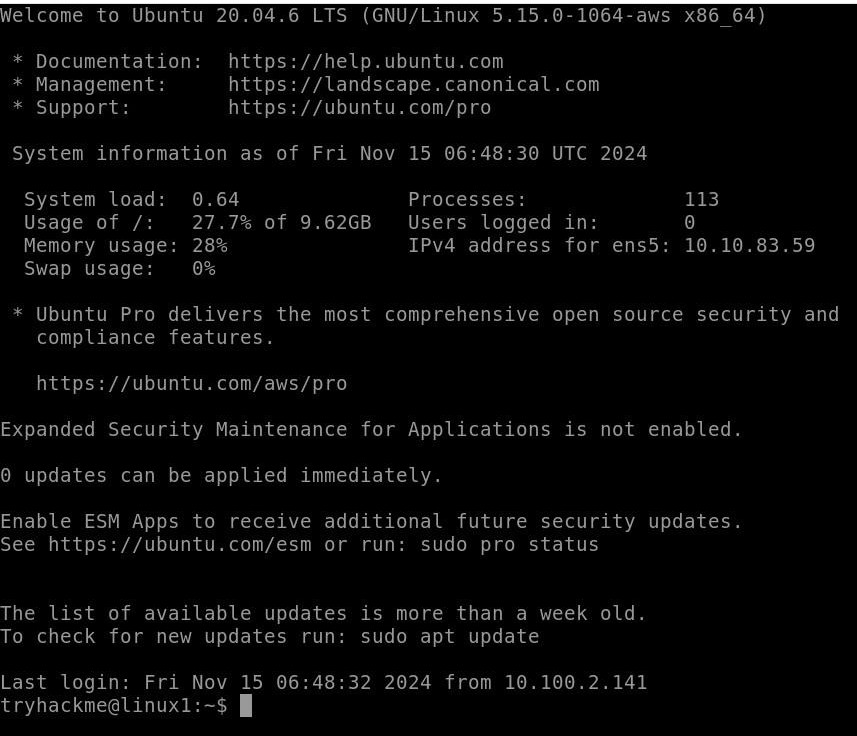
### File Operations:

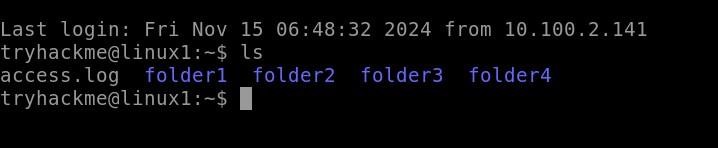
o :w: Save the file. o :q: Quit Vim. o :wq: Save and quit. o :q!: Quit without saving.

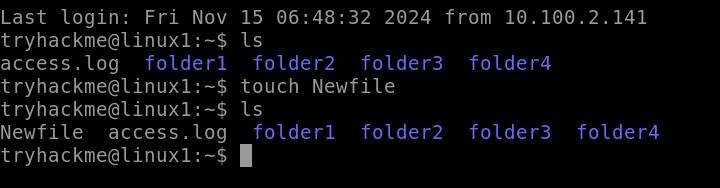
* + - :e filename: Open another file in the same session.
  + **Navigation:** o h, j, k, l: Move left, down, up, and right, respectively. o gg: Go to the start of the file. o G: Go to the end of the file. o w: Jump forward one word. o b: Jump backward one word.
    - Ctrl + f: Page down.
  + **Search & Replace:** o /pattern: Search forward for a pattern.
    - ?pattern: Search backward for a pattern.

### Editing:

* + - dd: Delete a line. o yy: Copy (yank) a line.
    - p: Paste the copied content. o u: Undo the last operation.
    - Ctrl + r: Redo the undone operation.

• 





### Nano

Nano is a simple, easy-to-use text editor for beginners. It is designed with simplicity in mind and displays commands at the bottom of the screen, making it user-friendly for new Linux users.

### Command Specifications:

* + **File Operations:**

o Ctrl + O: Save the file. o Ctrl + X: Exit Nano. o Ctrl + R: Read a file into the current buffer.

* + - Ctrl + T: Open the spell-checker.

### Navigation:

* + - Ctrl + A: Move to the beginning of the line.
    - Ctrl + E: Move to the end of the line.
    - Ctrl + Y: Scroll up one page.

### Editing:

* + - Ctrl + K: Cut the current line. o Ctrl + U: Paste the cut line. o Ctrl + ^: Begin selecting text (for cutting/copying).
    - Ctrl + Shift + 6: Cancel text selection.

### Search & Replace:

* + - Ctrl + W: Search for a string. o Ctrl + \: Search and replace.

### Emacs

GNU Emacs is one of the most versatile and extensible text editors available on Linux. It offers a variety of features beyond basic text editing, including email handling, web browsing, and even games. Emacs is known for its extensibility through Lisp programming.

***Command Specifications:***

* + **File Operations:** o Ctrl + X, Ctrl + S: Save the current file. o Ctrl + X, Ctrl + C: Exit Emacs.
    - Ctrl + X, Ctrl + F: Open a file.
  + **Navigation:** o Ctrl + A: Move to the beginning of the line.
    - Ctrl + E: Move to the end of the line.

**Editing:** o Ctrl + K: Kill (cut) the rest of the line. o Ctrl + Y: Yank (paste) the cut content. o Ctrl + Space: Set a mark (start selection). o Ctrl + W: Cut the selected text.

* + - Meta + W: Copy the selected text.

### Search & Replace:

* + - Ctrl + S: Incremental search forward. o Ctrl + R: Incremental search backward.
    - Meta + %: Search and replace.

### Gedit

Gedit is a graphical text editor that comes by default with the GNOME desktop environment. It offers a simple, intuitive interface for basic and moderate text editing tasks, while also supporting plugins for extended functionality.

### Kate

Kate is an advanced text editor that is part of the KDE desktop environment. It is a feature-rich editor tailored for developers, with syntax highlighting, multi-document editing, and robust search and replace options.

# Experiment – 6

## Create a file called WLCC.txt with some lines and display how many lines, words and characters are present

### Introduction :-

Here’s a brief overview of the concepts related to counting lines, words, and characters in a text file:

### Text Files:

A text file is a standard file format that contains unformatted text. It can be created and edited with any text editor.

### Line Count:

The line count refers to the number of lines in a text file. A line is typically defined as a sequence of characters ending with a newline character. This is useful for understanding the structure of the content.

### Word Count:

The word count indicates how many words are present in the text. Words are usually defined as sequences of characters separated by spaces or punctuation. This metric is often used in writing to meet specific requirements.

### Character Count:

The character count includes all characters in the text, including letters, numbers, punctuation, and whitespace. This count is important for understanding the length of the text and is relevant for formatting constraints.

### Using “WC” Command :--

* The wc (word count) command is a Unix/Linux utility that counts lines, words, and characters in files. It can take options to specify what to count:

o -l: Count lines o -w:

Count words o -c:

Count characters

* Using wc without options will return all three counts.

**Steps involves : --**

1. **Create the File**: You can use any text editor to create the file. Here's an example of what to put in WLCC.txt:
2. **Count Lines, Words, and Characters**: If you're using a command line interface, you can use the following commands (assuming you're in the directory where WLCC.txt is located):

**Lines**: Use wc -l wlcc.txt

**Words**: Use wc -w wlcc.txt

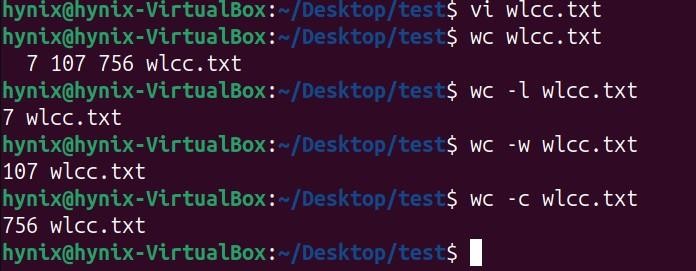
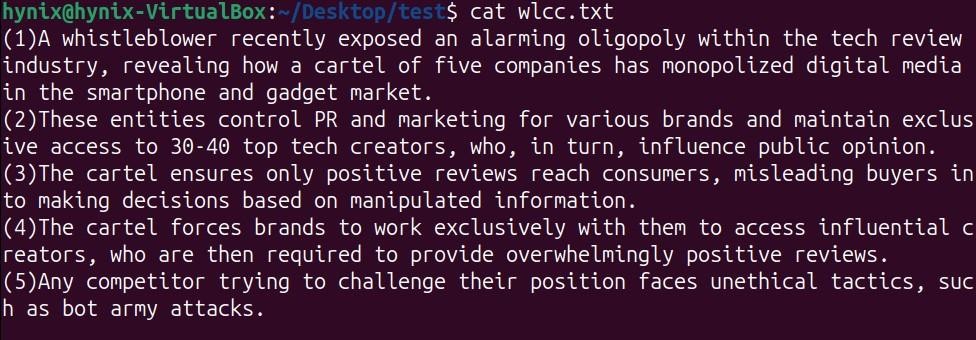
**Characters**: Use wc -c wlcc.txt

You can combine whole above code into one command **:**

### >> wc wlcc.txt

1. **Example Output**: The output will look something like this (the actual numbers may vary based on your content):

**Contents of wlcc.txt –**

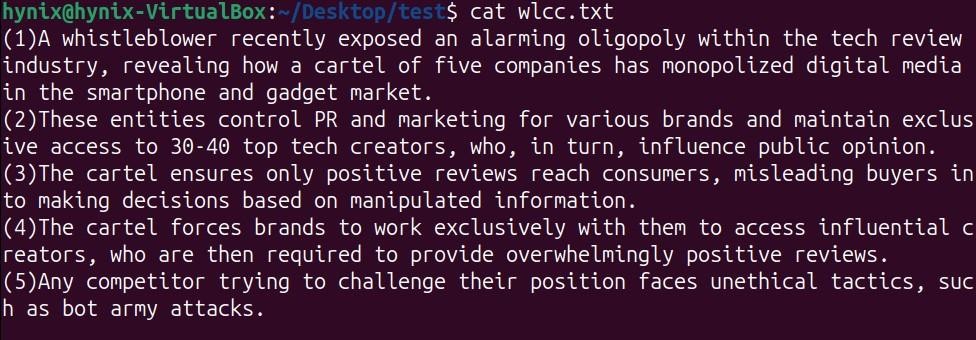


# Experiment – 7

## Append 10 more lines in the file WLCC.txt and display

On Linux, while working with files in a terminal sometimes we need to append the same data of a command output or file content. Append means simply add the data to the file without erasing existing data

Here given below, is the already present content in wlcc.txt file shown ---

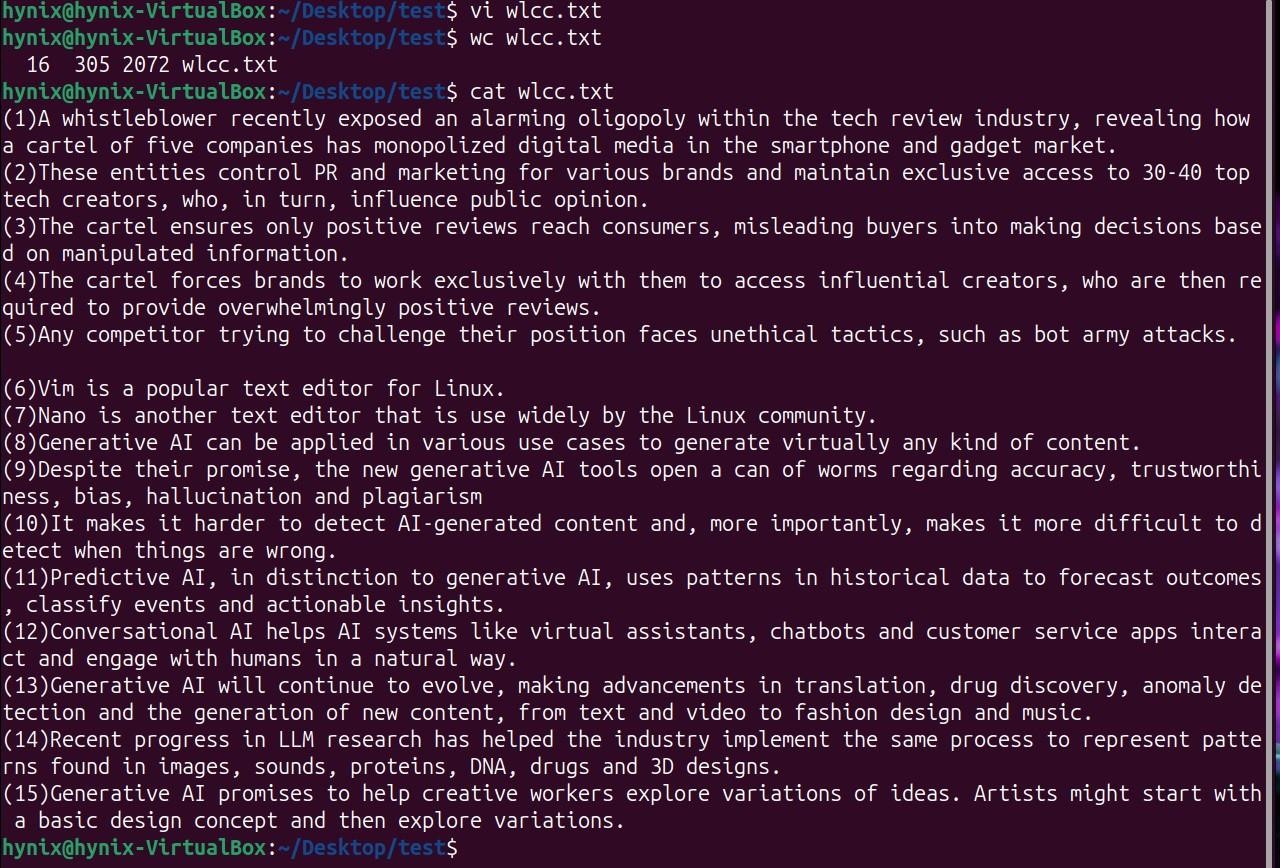


Contents of wlcc.txt -

Now in order to add new content line in above wlcc.txt file then we have to follow Steps involved :-

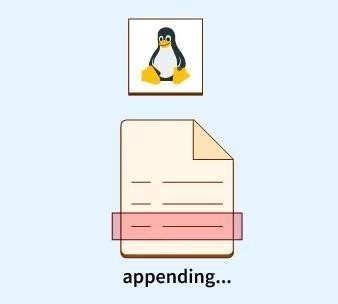
* 1. Opened the file wlcc.txt in editor.
  2. After adding those lines pressed ESC and typed :wq to save the state of the file and quit.

Appended wlcc.txt -



### Benefits of Appending :--

* **Data Preservation**: Appending allows users to add information without losing existing data, which is crucial for logs, configuration files, and documents.
* **Simplicity**: The method is straightforward and can be done with simple command-line instructions.



Appending content to files in Linux is a fundamental operation that allows for efficient data management. By understanding how to use redirection operators and commands like echo, users can manipulate files effectively without risking data loss.

# Experiment – 8

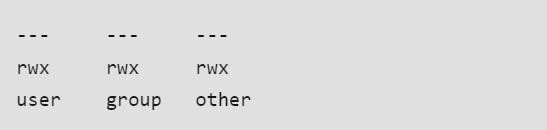
## Study and Use of commands for changing file permissions

File permissions are core to the security model used by Linux systems. They determine who can access files and directories on a system and how. This article provides an overview of Linux file permissions, how they work, and how to change them.

### Basics of File Permissions

In Linux, every file and directory has permissions that dictate who can read, write, or execute them. These permissions are set for three types of users:

1. **Owner**: The user who created the file.
2. **Group**: A group of users who have certain permissions on the file.
3. **Others**: All other users on the system.



### Permission Types

* + **Read (r)**: Allows the user to read the contents of the file.
  + **Write (w)**: Allows the user to modify the contents of the file.
  + **Execute (x)**: Allows the user to run the file as a program

### Viewing Permissions

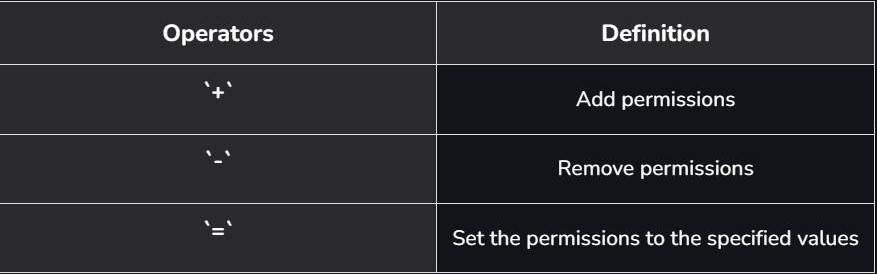
You can view file permissions using the ls -l command:

>> ls -l filename

### Changing Permissions

To change file permissions, you can use the **chmod** command.

1. **Symbolic Mode:** Use letters to specify changes.



Examples:

# Add execute permission for the owner

>> chmod u+x filename

# Remove write permission for the group

>> chmod g-w filename

# Set read permission for others

>> chmod o=r filename

1. **Numeric Mode**: Use numbers to set permissions.

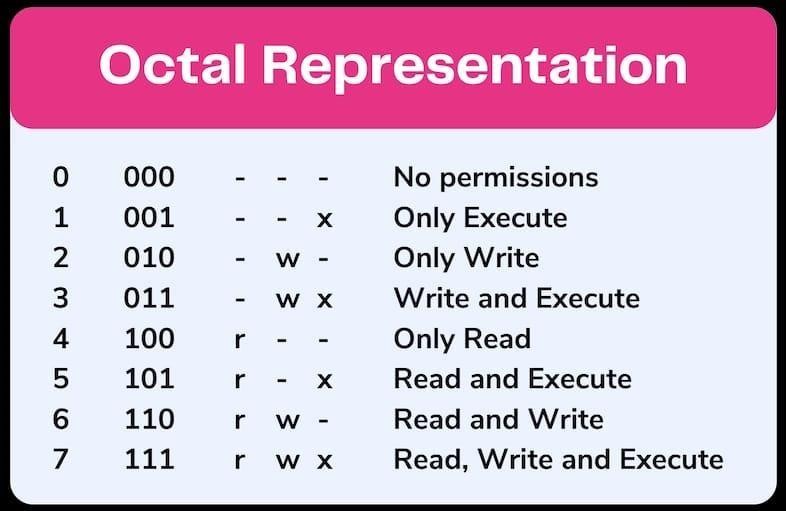
{ Read: 4, Write: 2, Execute: 1 } Each permission level can be summed:

* + 7 (4+2+1) = read, write, execute
  + 6 (4+2) = read, write

### Reading the Security Permissions in Linux

* + **“rw-“**: the first three characters `rw-`. This means that the owner of the file can “read” it (look at its contents) and “write” it (modify its contents).
  + **“r-x”**: the second set of three characters “r-x”. This means that the members of the group can only read and execute the files.
  + **“r–“**: The final three characters “r–” show the permissions allowed to other users who have a UserID on this Linux system.

### Octal Notation in linux :-



Using the octal notations table instead of ‘r’, ‘w’, and ‘x’. Each digit octal notation can be used for either of the group ‘u’, ‘g’, or’o’.

So, the following work is the same.

chmod ugo+rwx [file\_name] chmod 777 [file\_name]

Both of them provide full read write and execute permission (code=7) to all the group. The same is the case with this.

chmod u=r,g=wx,o=rx [file\_name]

chmod 435 [file\_name]

Both the codes give read (code=4) user permission, write and execute (code=3) for the group and read and execute (code=5) for others.

chmod 775 [file\_name] chmod ug+rwx,o=rx [file\_name]

Both the commands give all permissions (code=7) to the user and group, read and execute (code=5) for others.

**Commands** :---

( Numeric Mode )

**1. -rw-----**

Only the User has read and write permissions



1. **drwx----**

Only the Owner has read, write and execute permissions



1. **-rwxr-xr-x-**



**4. -rw-r--r--**



1. **-rw-rw-r—**



**( Named Mode )**

* + Allows everybody to read **file.txt**.



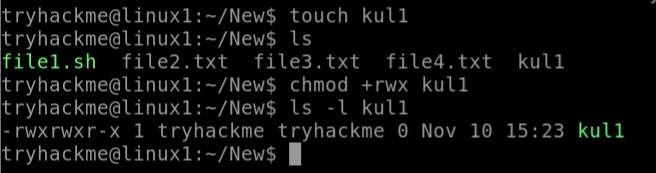
* + Strips everybody of all permissions, except for the owner who retains any former permissions.



**Example :-**

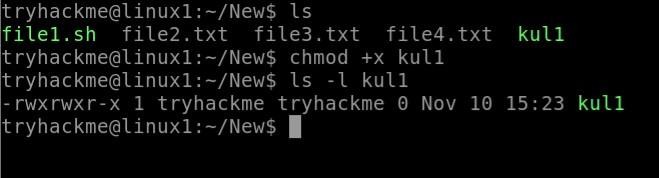
Example 1. Add read, write, and execute permissions to a file:

**Syntax:-** $ chmod +rwx filename



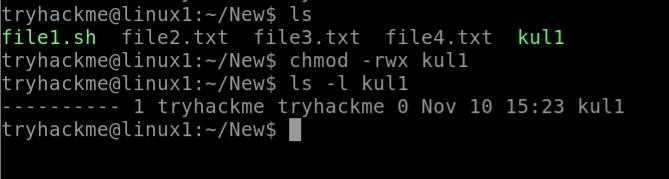
**Example 2.** Add execute permissions to a file:

**Syntax:-**$ chmod +x filename



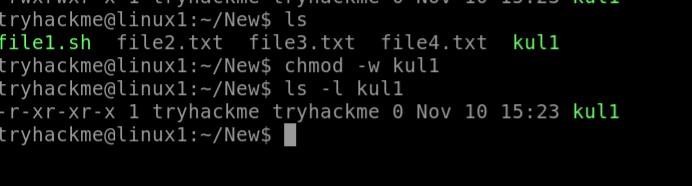
**Example 3.** Remove all permissions from a file:

**Syntax:-** $ chmod -rwx filename



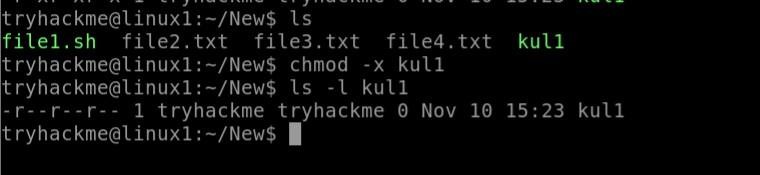
**Example 4.** Remove write permissions from a file:

**Syntax:-** $ chmod -w filename



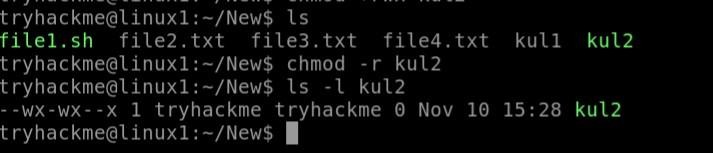
**Example 5.** Remove execute permissions from a file:

**Syntax:-**$ chmod -x filename



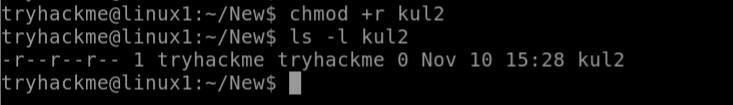
**Example 6.** Remove read permissions from a file:

**Syntax:-**$ chmod -r filename



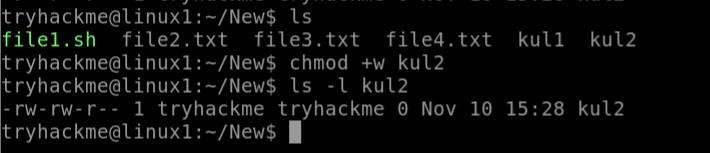
**Example 7.** Add read permissions to a file:

**Syntax:-**$ chmod +r filename



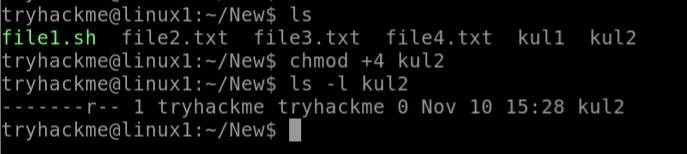
**Example 8.** Add write permissions to a file:

**Syntax:-**$ chmod +w filename



**Example 9.** Add 4(read) permissions to a file:

**Syntax:-**$ chmod +4 filename



**Example 10.** Add 2(write) permissions to a file:

**Syntax:-**$ chmod +2 filename

