

Project File

**Linux System Administration**

**STUDENT PARTCULARS**

**GROUP NUMBER:**

**MEMBER NAME MEMBER ROLL NO**

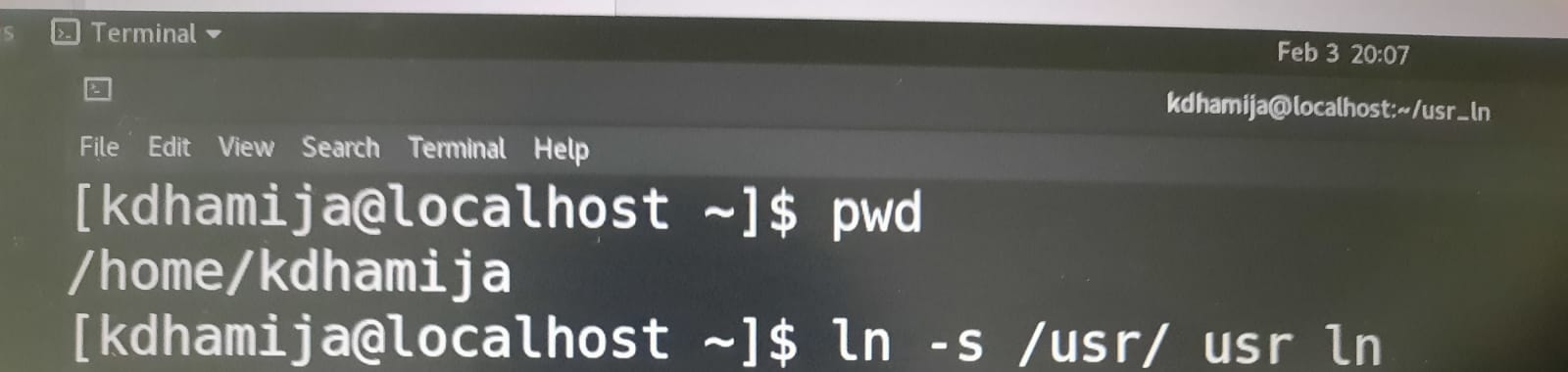
1. Kritika Dhamija 2110990801
2. Priyanshu Singh 2110991088
3. Ravi Ranjan Kumar Ojha 2110991142
4. Rishab 2110991156
5. Vaibhav Kansal 2110991861

**Index**

|  |  |  |
| --- | --- | --- |
| **S. No.** | **Title of Task** | **Page No.** |
| 1 | ls, cd, pwd | 4-6 |
| 2 | mkdir,rm,mv,cp,cat,nl,rmdir | 7-22 |
| 3 | more,less,head,tail | 23-27 |
| 4 | vi editor | 28-30 |
| 5 | Linux Programs | 31-37 |

* **PWD Command**

**PWD** (print working directory) command **writes to standard output the full path name of your current directory (from the root directory).**

**Example 1:** Get Working Directory Path

The command outputs the current working directory absolute path. In this case, the home directory path.

### Example 2: Using the -L Option

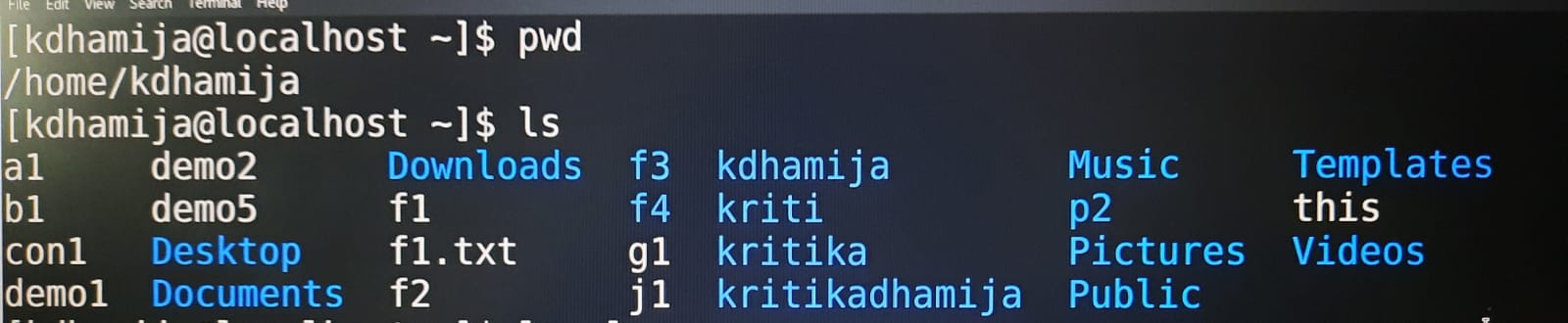
The **-L** option instructs **pwd** to print the working directory path, including any symlinks.

### Example 3: Using the -P Option

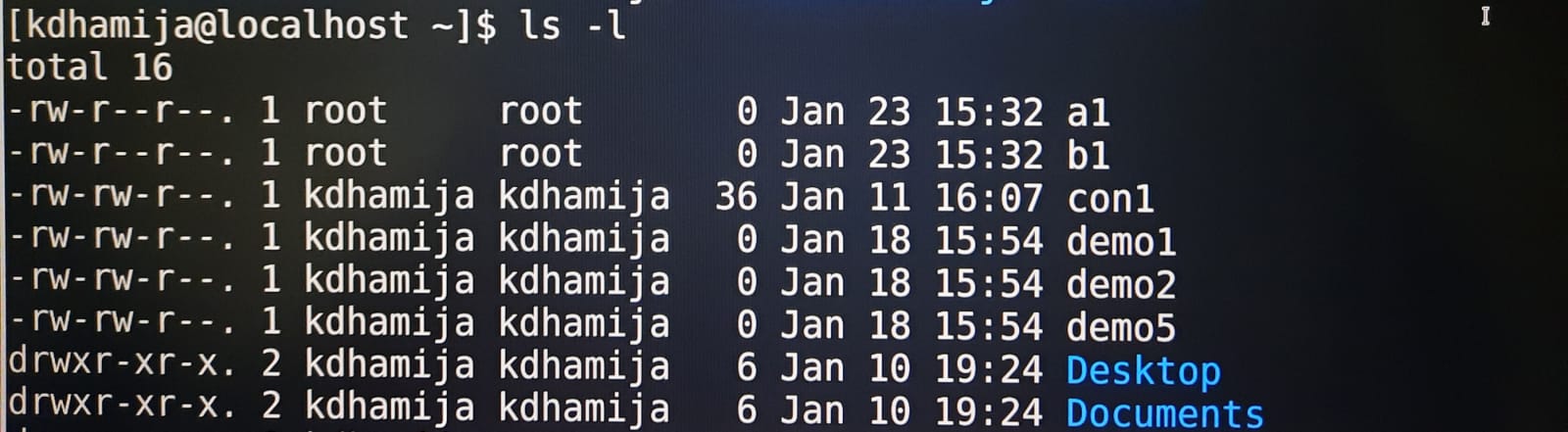
### 

### LS Command

The **ls** is the list command in Linux. It will show the full list or content of your directory.

**Example 1:** ls

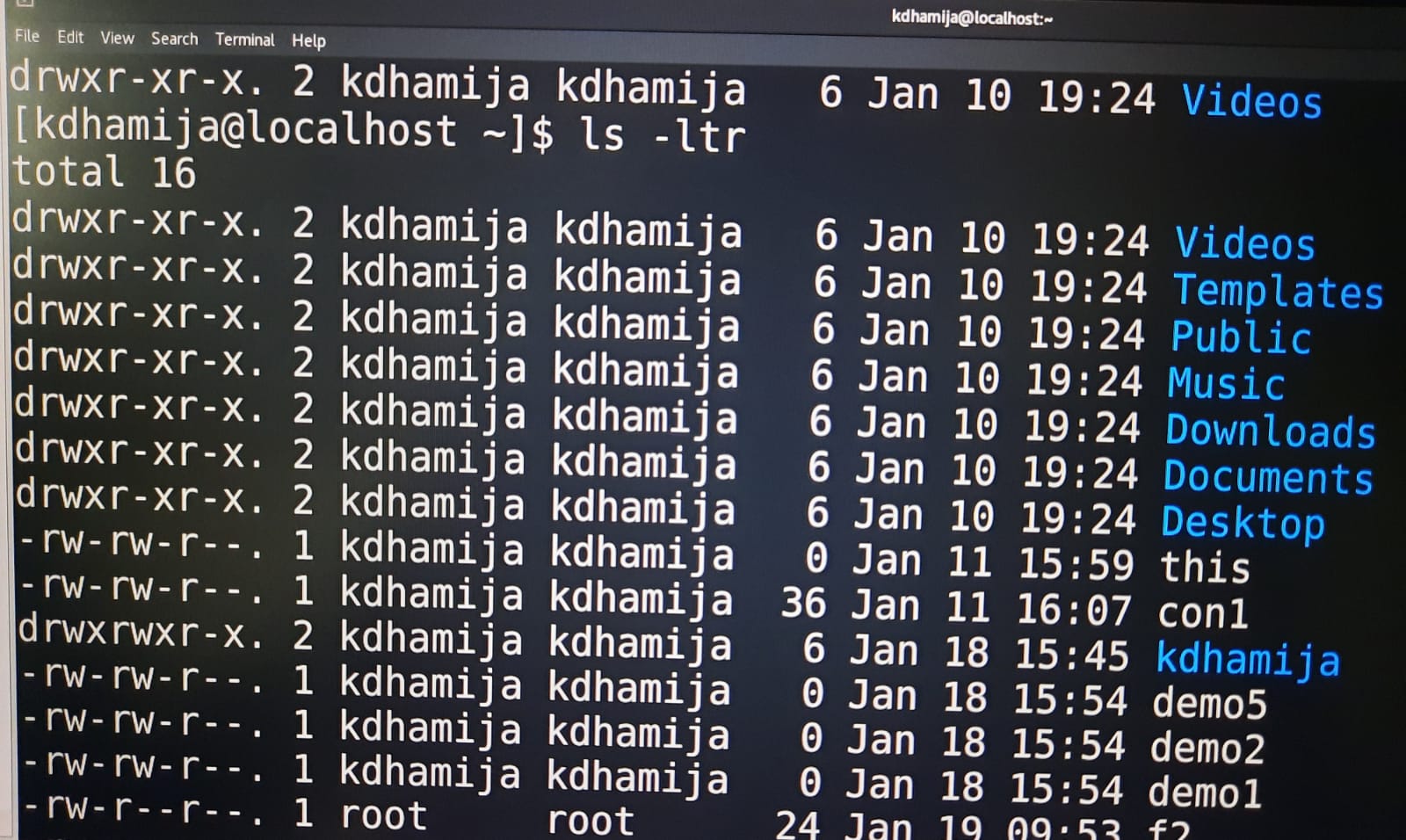
**Example 2:** ls -l

It will show the list in a long list format.

**Example 3:** ls -lt

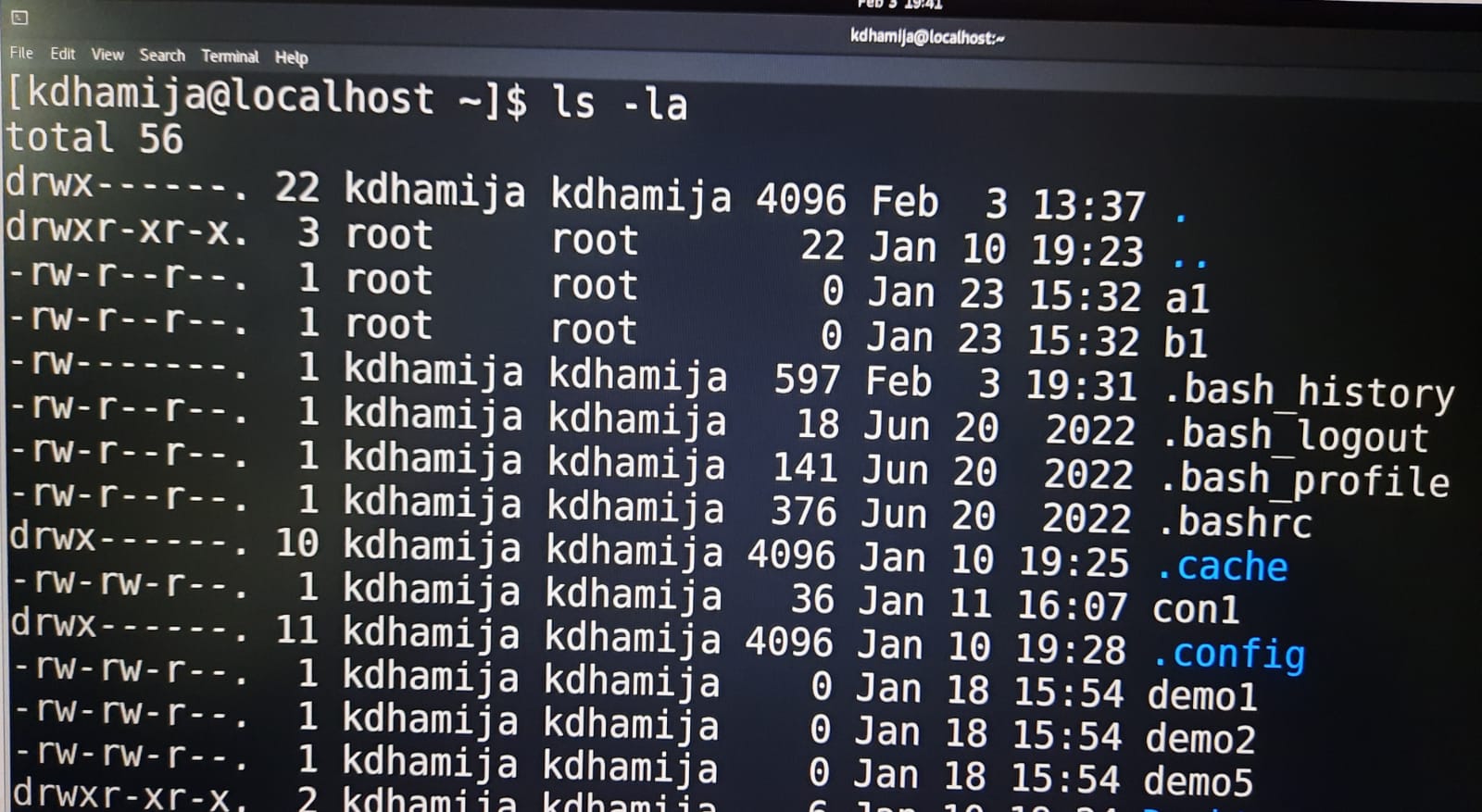
It will sort the list by displaying recently modified filed at top.

**Example 4:** ls -ltr

View Reverse Output Order by Date.

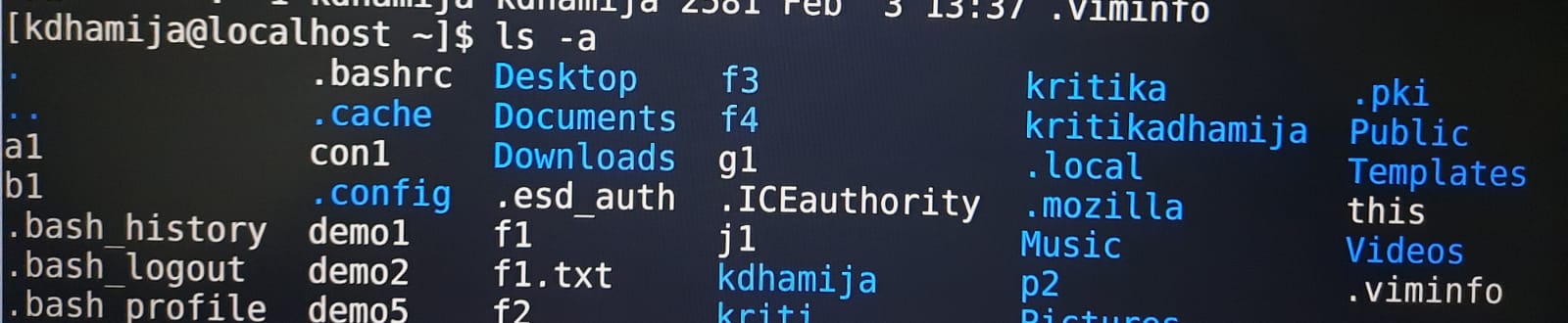
**Example 5:** ls -la

Show file details



**Example 6:** ls -a

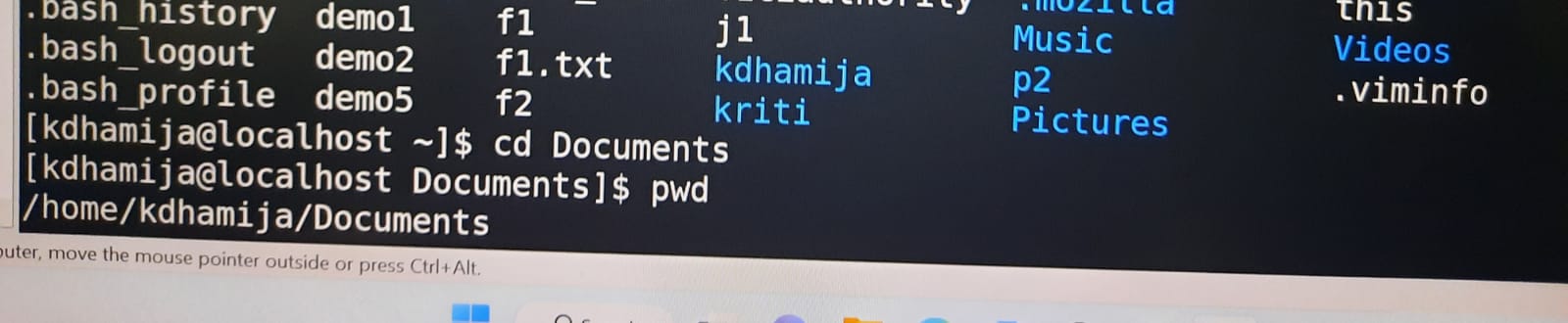
list all files including hidden file starting with '.'.



* **CD Command**

cd command is **used to change the current working directory** ( i.e. , in which the current user is working).

**Example :**



* **MKDIR command**

The mkdir stands for 'make directory'. With the help of mkdir command, you can create a new directory wherever you want in your system. Just type **"mkdir directory\_name** , in place of directory\_name type the name of new directory, you want to create and then press enter.

**SYNTAX :** $ mkdir directory\_name

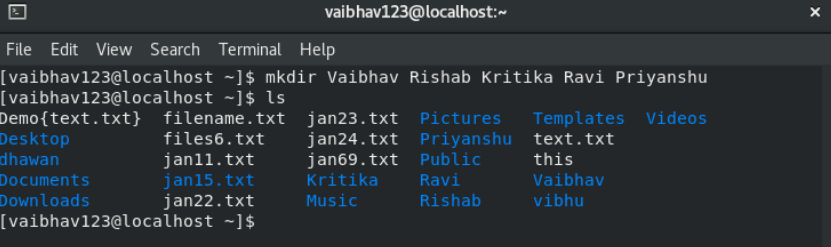
**OUTPUT:** will create a new directory



## **To make multiple directories**

**SYNTAX :** $ mkdir directory\_name1 directory\_name2 directory\_name3 ……

**OUTPUT:** will create a new directories

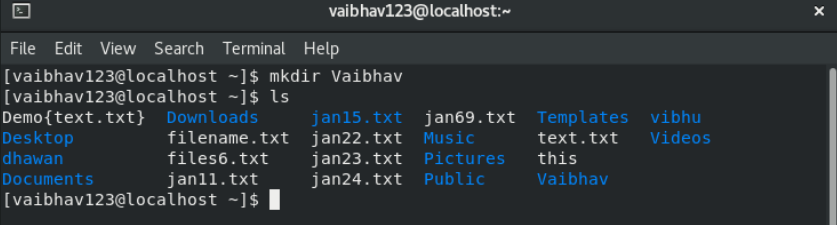


Add directory including its sub directory

With the help of mkdir -p command you can create sub-directories of a directory. It will create parent directory first, if it doesn't exist. But if it already exists, then it will not print an error message and will move further to create sub-directories.

**SYNTAX :** $ mkdir -p directory\_name/sub-directory\_name

**OUTPUT:** will create a new sub directory in existing directory

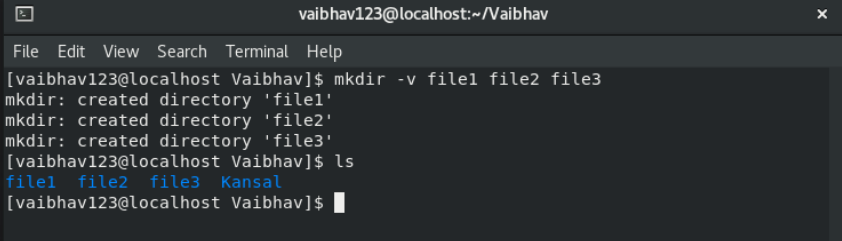


# **Print a message for each created directory**

'mkdir -v' command will print a message with every new file created.

**SYNTAX :** $ mkdir -v file1 file2 file3

**OUTPUT:** will create files in existing directory



In above example, I have created **'file1 file2 file3'** and you can see the message for every individual file.

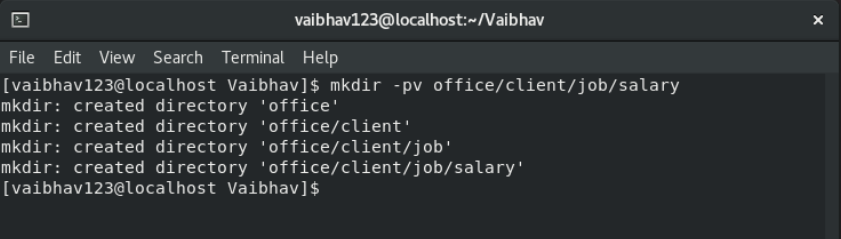
If files already exist then it will give an error message as shown below.



Both the commands 'mkdir -pv' and 'mkdir -vp' are same. By combining -p you can create a long list files together like **"office/client/job/salary"**with a printed message for each file.

**SYNTAX :** $ mkdir -pv

**OUTPUT:** will create long list files

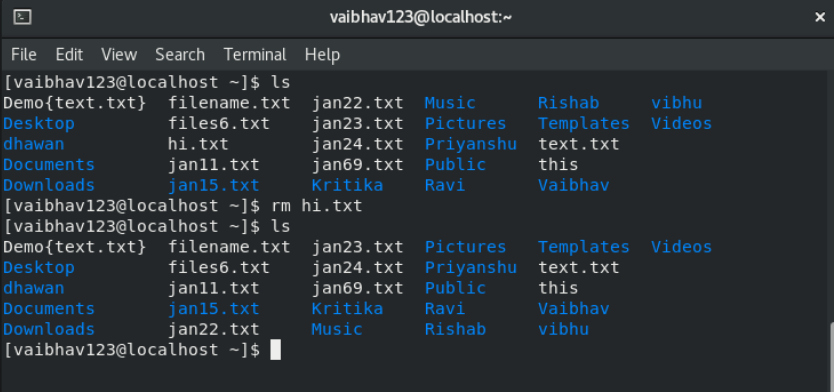


* **RM command**

The **rm** command removes the entries for a specified file, group of files, or certain select files from a list within a directory. User confirmation, read permission, and write permission are not required before a file is removed when you use the **rm** command. However, you must have write permission for the directory containing the file.

**SYNTAX :** $ rm file\_name

**OUTPUT:** will remove the file

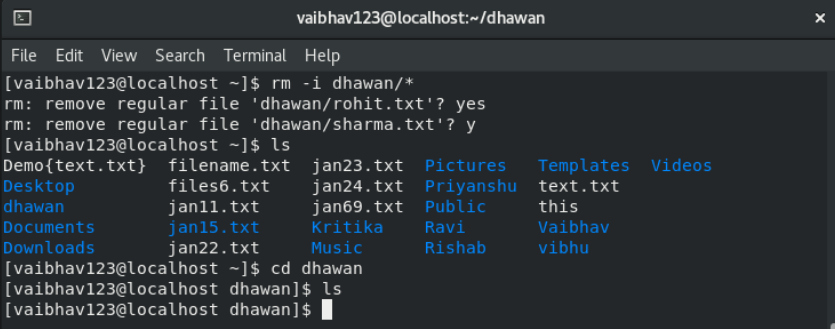


**Remove a file interactively.**

To delete all the files in the mydir directory, one by one After each file name displays, type y and press Enter to delete the file. Or to keep the file, just press Enter.

**SYNTAX :** $ rm -i directory\_name/\*

**OUTPUT:** will remove the files of directory

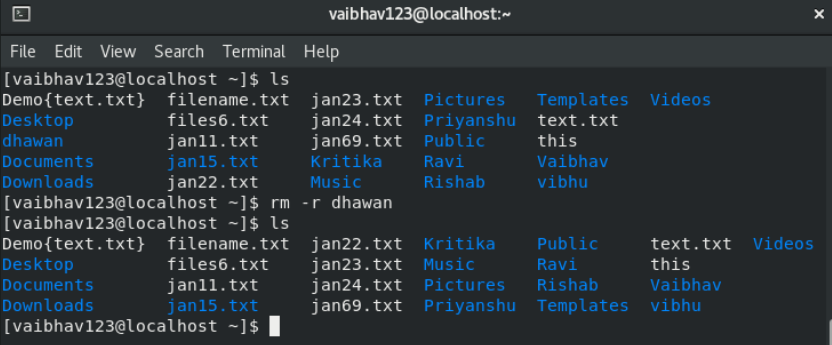


**To delete a directory recursively**

With rm '-r' option, you can delete a directory having sub directories inside it. So you don't need to delete sub-directories manually

**SYNTAX :** $ rm -r directory\_name

**OUTPUT:** will remove the directory



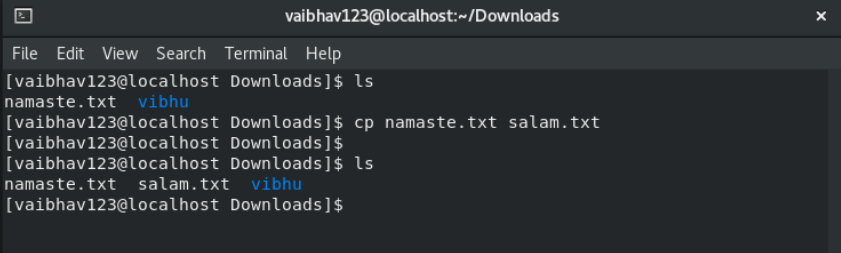
* **CP command**

cp' means copy. 'cp' command is used to copy a file or a directory.

To copy a file into the same directory syntax will be,

**SYNTAX :** $ cp existing\_file\_name newfile\_name

**OUTPUT:** is used to copy a file

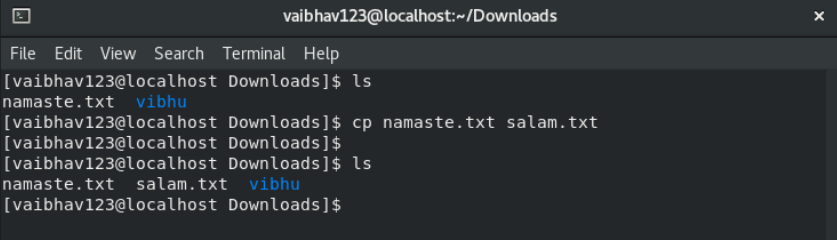


To copy a directory along with its sub directories.

Option **'r'** with the copy command can be used to copy a directory including all its content from a source directory to the destination directory.

**SYNTAX :** $ cp -r source\_directory destination\_directory

**OUTPUT:** is used to copy a directory with its sub directories

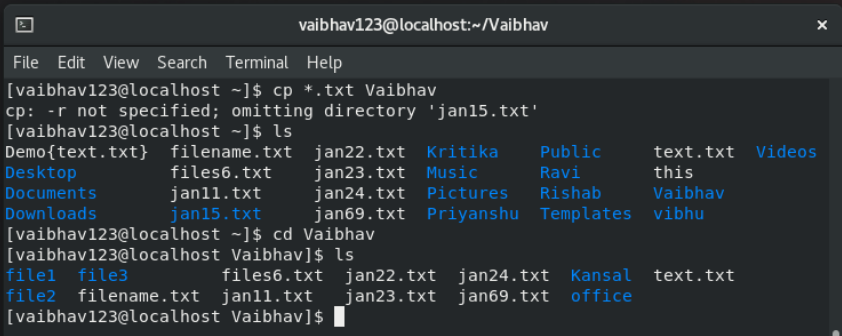


To copy multiple file or directories in a directory

Multiple files or directories can be copied to a destination directory at once. In this case, target must be a directory. To copy multiple files you can use**wildcards** (cp \*.extension) having same pattern.

**SYNTAX :** $ cp \*.<extension> destination\_directory

**OUTPUT:** is used to copy multiple directories or files in directory

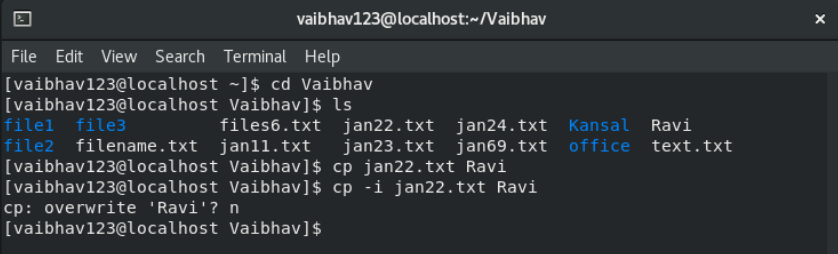


Asks for confirmation

The cp '-i' option allows you to confirm once before overwriting your file.

**SYNTAX :** $ cp -i file\_name destination\_directory

**OUTPUT:** is used to copy file before confirming once

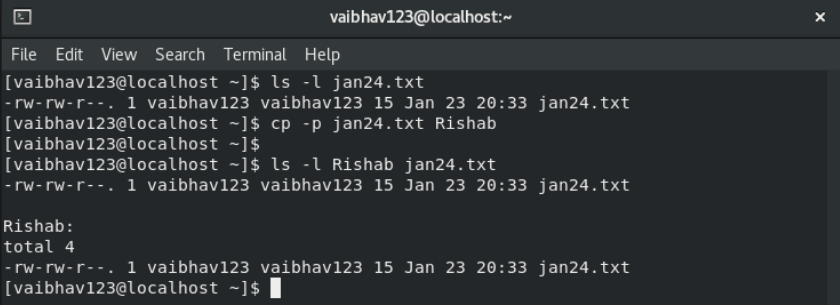


Preserves attribute of a file

The cp '-p' option is used to preserve the properties and attributes of a file. You can also preserve the selected properties which you want.

**SYNTAX :** $ cp -p file\_name destination\_directory

**OUTPUT:** is used to preserve attributes of file



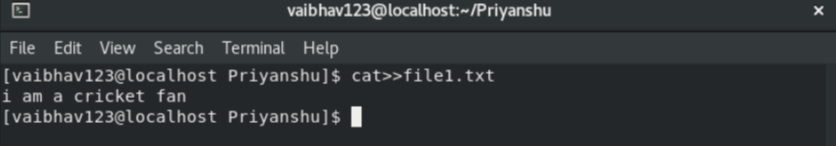
* **CAT Command**

**Cat** (concatenate) command is very frequently used in Linux. It reads data from the file and gives their content as output. It helps us to create, view, concatenate files. So let us see some frequently used cat commands.

**Create a file**

**SYNTAX :** $ cat>>file\_name

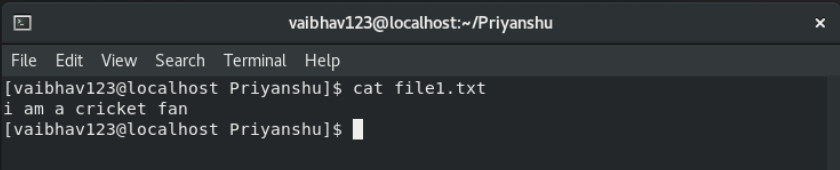
**OUTPUT:** will create a new file



**To view a single file**

**SYNTAX :** $ cat file\_name

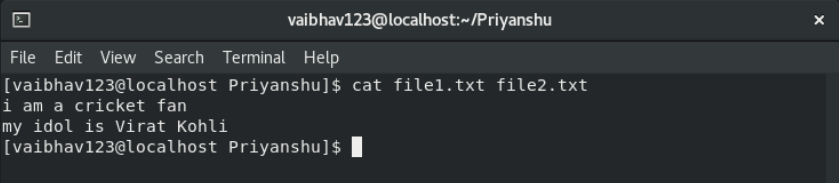
**OUTPUT:** will show the contents of file



**To view multiple files**

**SYNTAX :** $ cat file\_name1 file\_name2 ….

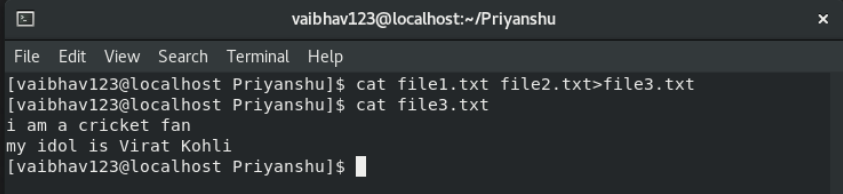
**OUTPUT:** will show the contents of files



**Copy the contents of one file to another file**

**SYNTAX :** $ cat [filename-whose-contents-is-to-be-copied] > [destination-filename]

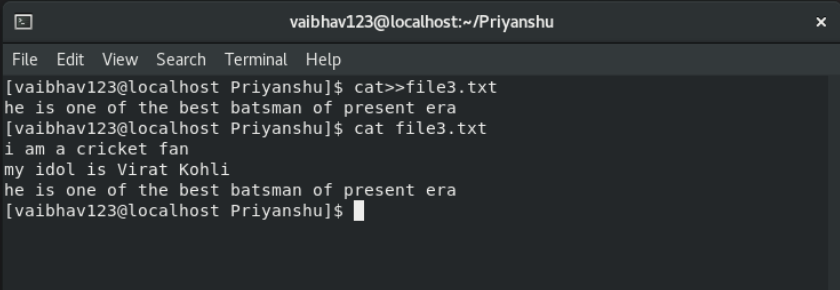
**OUTPUT:** content will be copied to the destination



**Cat command can append the contents of one file to the end of another file**

**SYNTAX :** $ cat file1 >> file2

**OUTPUT:** Will append the contents of one file to the end of another file

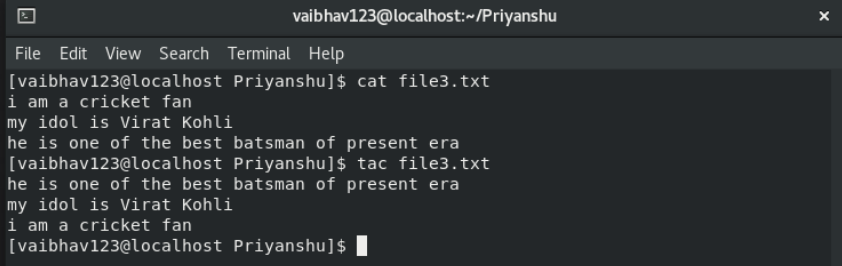


**Reverse the content of the file**

 Tac command in Linux is used to concatenate and print files in reverse. This command will write each FILE to standard output, the last line first. When no file is specified then this command will read the standard input.

**SYNTAX :** $ tac file\_name

**OUTPUT:** Will append the contents of one file to the end of another file in reverse order



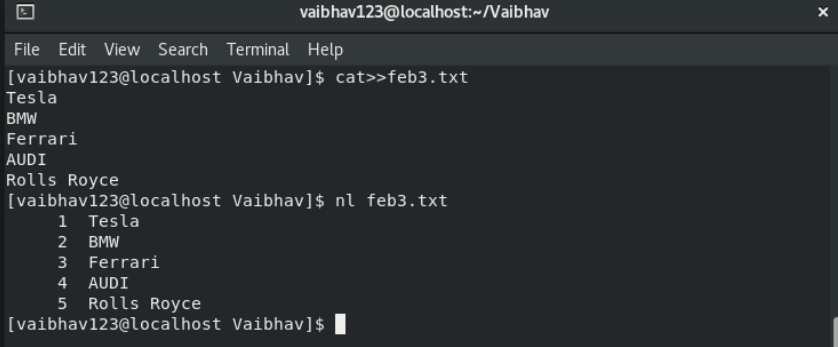
* **NL command**

Linux offers a lot of text formatting command line tools, with each of them having a different purpose. One such utility is **nl**, which lets you number lines in files. In this tutorial, we will discuss the basics of this command

**To display a file with line numbers**

**SYNTAX :** $ nl file\_name

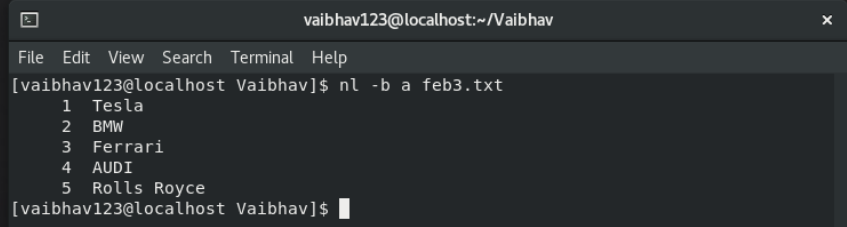
**OUTPUT:** Numbers all non-empty lines.



**To number all lines (including empty lines also)**

**SYNTAX :** $ nl -b a file\_name

**OUTPUT:** Numbers all lines whether empty or non-empty

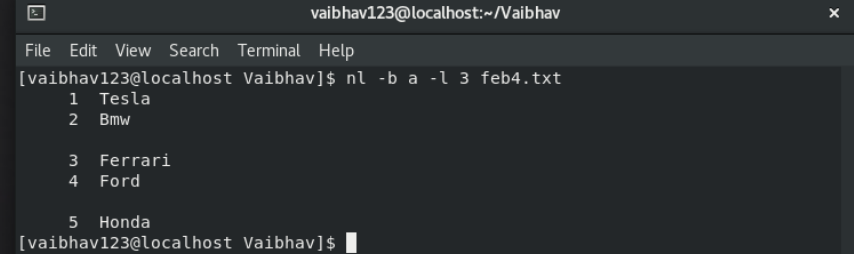


**Count multiple, consecutive, non-empty lines as one**

l optionis used to count all non-empty lines as a single logical line to nl command.Linux considers **NUMBER** consecutive empty lines as a single logical line for numbering, and only numbers the last one. If any empty consecutive lines less than **NUMBER** occur, it will discard them.

**SYNTAX :** $ nl -b a -l 3 file\_name

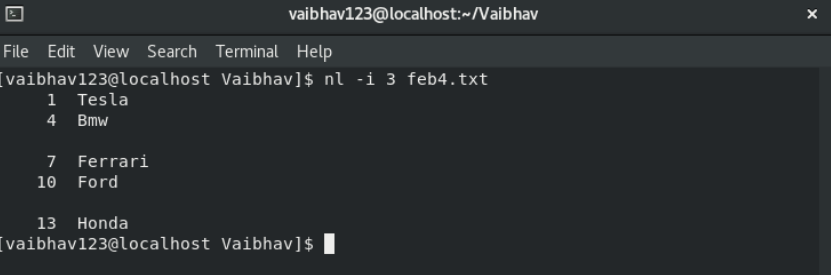
**OUTPUT:** Numbers all lines each and every



**Override default increment**

**SYNTAX :** $ nl -i 3 file\_name

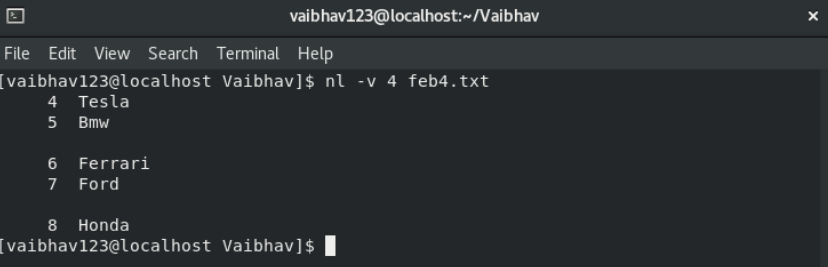
**OUTPUT:** The first line number is 1 and cannot be changed using **-i**.



**To make the starting line number different**

**SYNTAX :** $ nl -v 4 file\_name

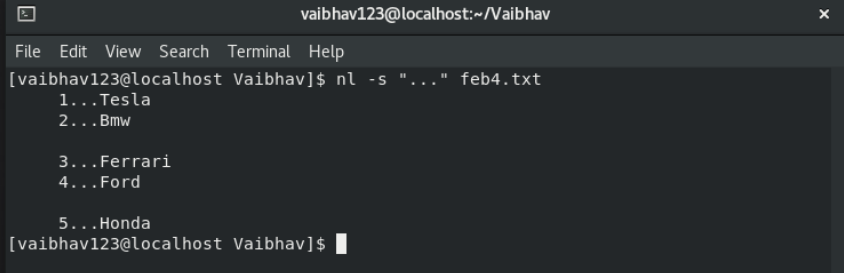
**OUTPUT:** The default line number is 1. This can be changed using the -v option..



**Add a string literal after line numbers**

**SYNTAX :** $ nl -s "..." file\_name

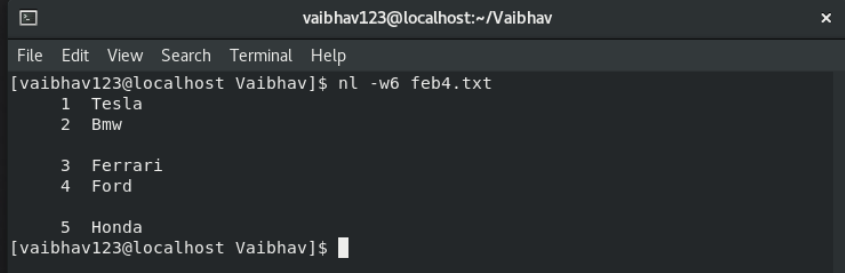
**OUTPUT:** Any STRING literal can be added after a line number using the -s option.



**Change column for line numbers**

**SYNTAX :** $ nl -w 2 file\_name

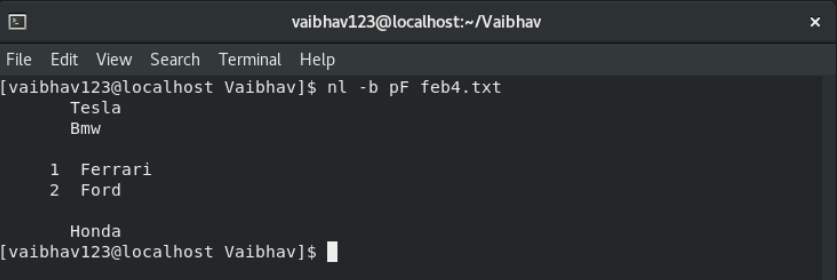
**OUTPUT:** Different columns can be used to display the file output using the -w option. The default column number is 1.



**To number all logical lines that match the specified REGEX**

**SYNTAX :** $ nl -b pF file\_name

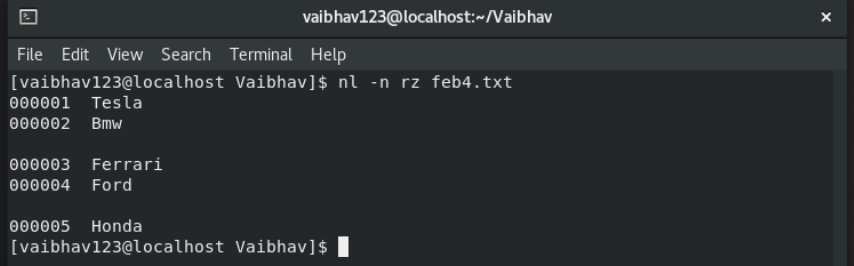
**OUTPUT:** The following command will number those lines that begin with **F**.



**To print the lines using a different number format**

**SYNTAX :** $ nl -n rz file\_name

**OUTPUT:** The default numbering format is rz

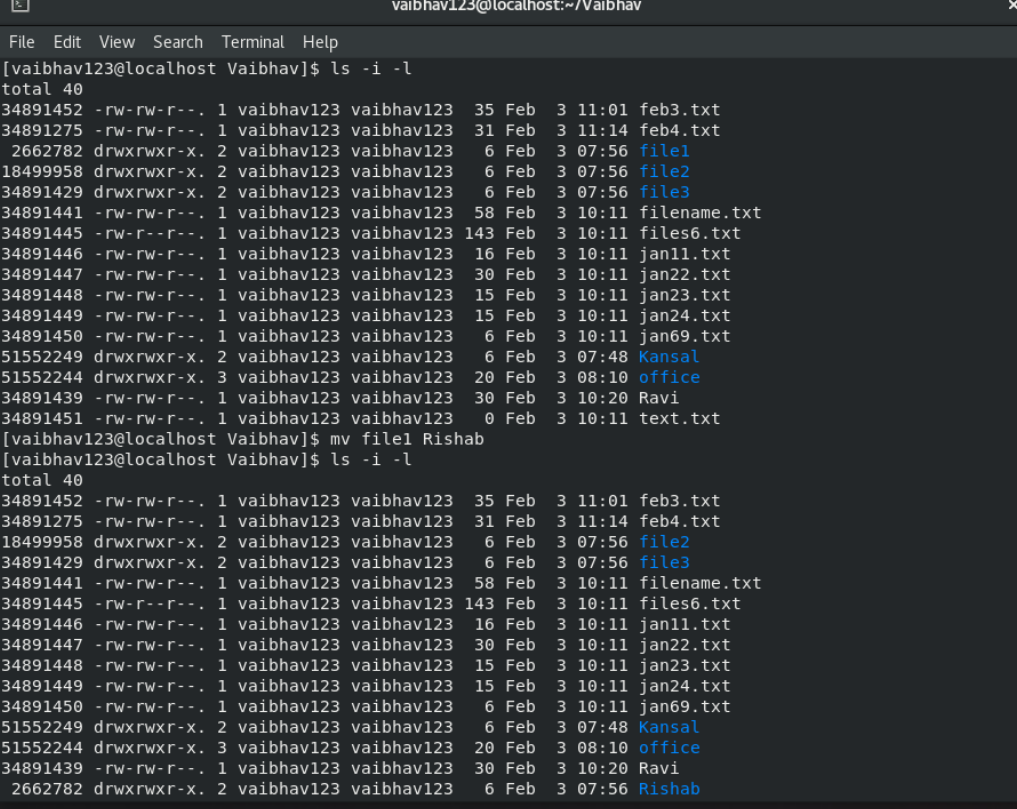


* **MV command**

Linux mv command is used to move existing file or directory from one location to another. It is also used to rename a file or directory. If you want to rename a single directory or file then **'mv'** option will be better to use.

**SYNTAX :** $ mv file\_name directory\_name

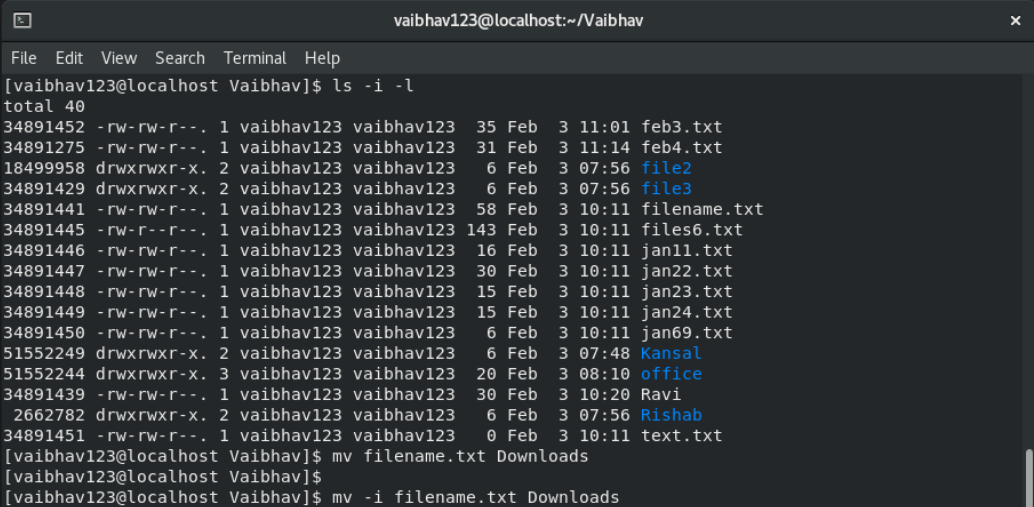
**OUTPUT:** move the file from one point to another



**Asks for permission to over write**

**SYNTAX :** $ mv -i file\_name directory\_name

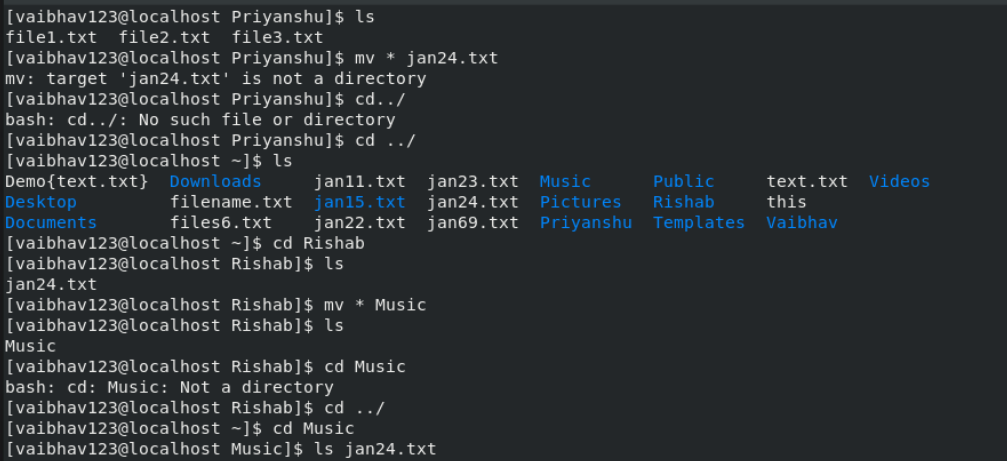
**OUTPUT:** it will silently over write the existing file.But if you'll use **'i'** option then it will first ask for your permission to over write it.



**Move multiple files to a specific directory**

**SYNTAX :** $ mv \* directory\_name

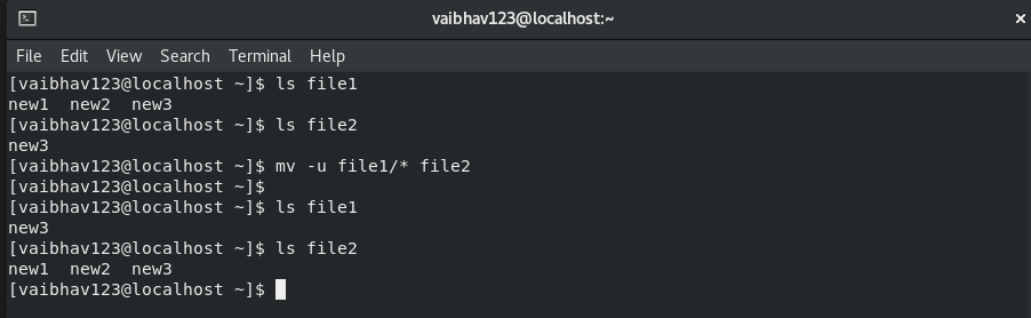
**OUTPUT:** move all the files from the current directory to another specified directory at once.



**Only move those files that doesn't exist**

**SYNTAX :** $ mv -u file\_name

**OUTPUT:** moves all the files to the destination directory

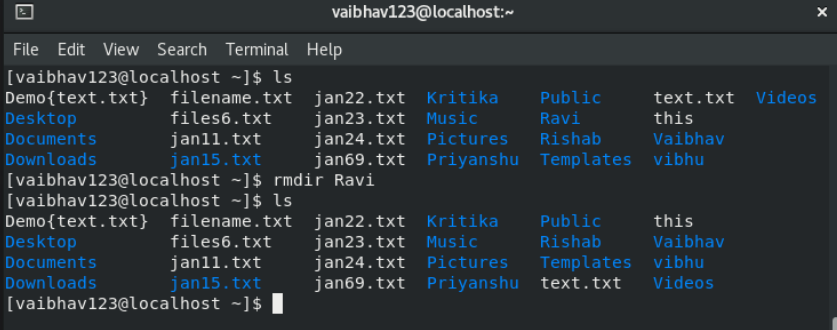


* **RMDIR command**

This command is used to delete a directory. But will not be able to delete a directory including a sub-directory. It means, a directory has to be empty to be deleted.

**SYNTAX :** $ rmdir directory\_name

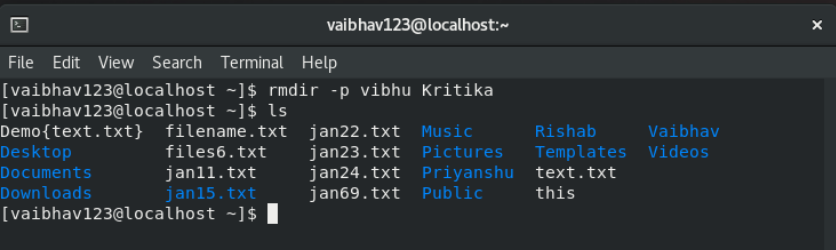
**OUTPUT:** Deletes the directory



**Delete a directory including its sub-directories**

**SYNTAX :** $ rmdir -p directory\_name

**OUTPUT:** Deletes the directory with its sub-directory



* **HEAD Command**

The Linux head command prints the first lines of one or more files (or piped data) to standard output. By default, it shows the first 10 lines. However, head provides several arguments you can use to modify the output.

**SYNTAX:**

head [filename]

**Output:**

It shows the first 10 lines



**SYNTAX:**

head -n [--lines] [filename]

**Output:**

show the specified number of lines



**SYNTAX:**

head -c [file name]

**Output:**

Count the number of file in bites

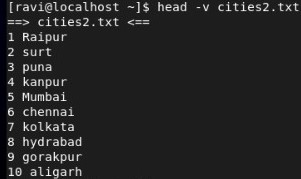


**SYNTAX:**

head -v [filename]

**Output:**

show the file name tag



* **TAIL Command**

It is the complementary of head command. The tail command ,as the name implies, print the last N number of data of the given input. By default it print the last 10 lines of the specified files.

**SYNTAX:**

tail [filename]

**Output:**

print the last N number of data

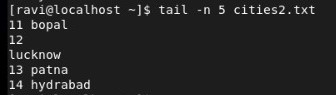


**SYNTAX:**

tail -n [--lines] [filename]

**Output:**

print last 5 lines



* **MORE Command**

**More** is a command to view (but not modify) the contents of a text file one screen at a time. Programs of this sort are called pagers. More is a very basic pager. Which means, it is an application that allows to navigate a file page-by-page

**SYNTAX:**

more [filename]

**Output:**

* Enter key : To scroll down page line by line.
* Space bar : To go to next page. Displays next k lines of text which defaults to screen size.



* **LESS Command**

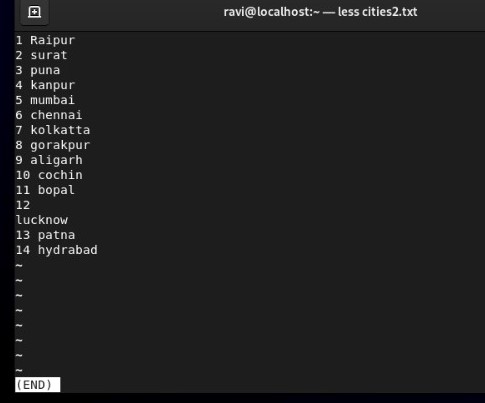
The less command is a Linux terminal pager that shows [a file's contents](https://phoenixnap.com/glossary/what-is-a-file) one screen at a time. It is useful when dealing with a large text file because it doesn't load the entire file but accesses it page by page, resulting in fast loading speeds.

**SYNTAX:**

Less [filename]

**Output:**

Running the command without options outputs the input file's contents in the default manner.



* **VI commands:**

1. **i** – Insert at cursor (goes into insert mode)
2. **a** – Write after cursor (goes into insert mode)
3. **A** – Write at the end of line (goes into insert mode)
4. **ESC** – Terminate insert mode
5. **u** – Undo last change
6. **U** – Undo all changes to the entire line
7. **o** – Open a new line (goes into insert mode)
8. **dd** – Delete line
9. **3dd** – Delete 3 lines.
10. **D** – Delete contents of line after the cursor
11. **C** – Delete contents of a line after the cursor and insert new text. Press ESC key to end insertion.
12. **dw** – Delete word
13. **4dw** – Delete 4 words
14. **cw** – Change word
15. **x** – Delete character at the cursor
16. **r** – Replace character
17. **R** – Overwrite characters from cursor onward
18. **s** – Substitute one character under cursor continue to insert
19. **S** – Substitute entire line and begin to insert at the beginning of the line
20. **~** – Change case of individual character

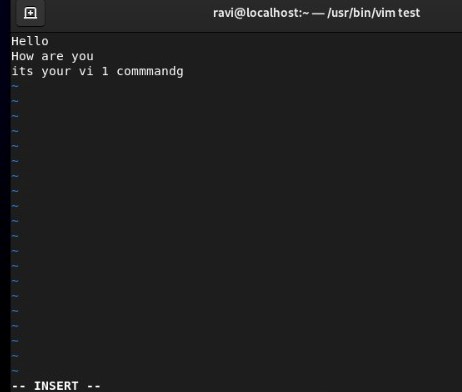
**SYNTAX:**

vi [filename]

* This mode is for inserting text in the file.
* You can switch to the Insert mode from the command mode **by pressing ‘i’ on the keyboard**
* Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
* To return to the command mode and save the changes you have made you need to press the Esc key

**Output:**

* This mode is for inserting text in the file.
* You can switch to the Insert mode from the command mode **by pressing ‘i’ on the keyboard**
* Once you are in Insert mode, any key would be taken as an input for the file on which you are currently working.
* To return to the command mode and save the changes you have made you need to press the Esc key

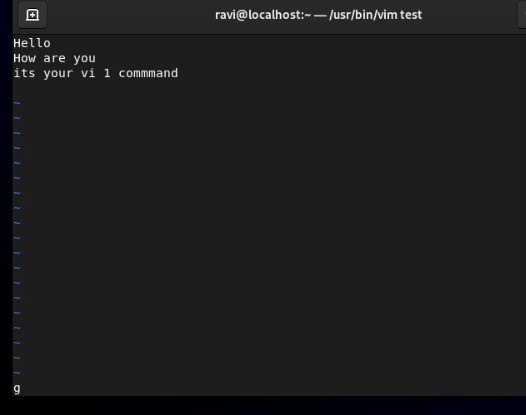


**SYNTAX:**

Esc

**Output:**

Terminate insert mode.

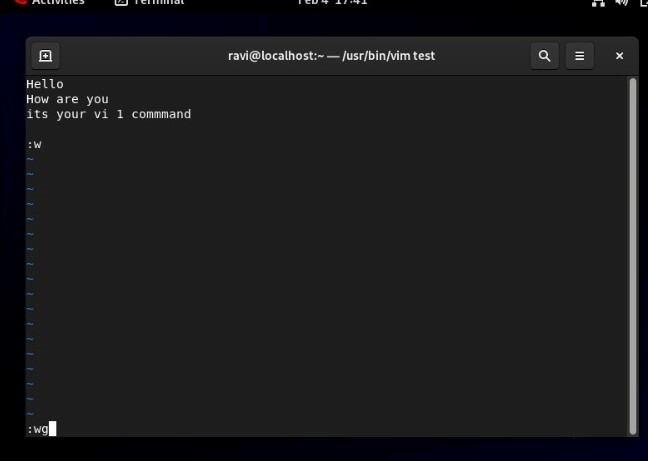


1. **Command:**

**:w**

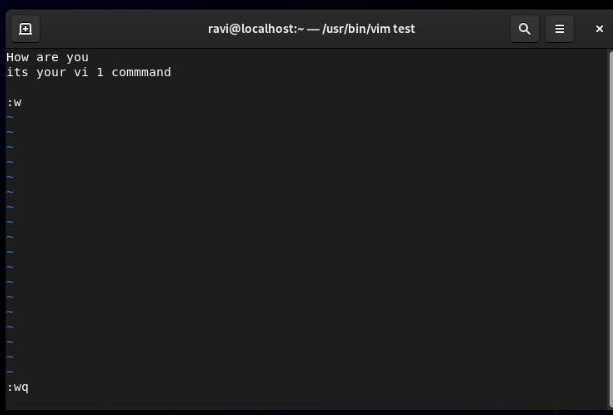
**Output:**

**Save data**

**Command:**

:wq

**Output:**

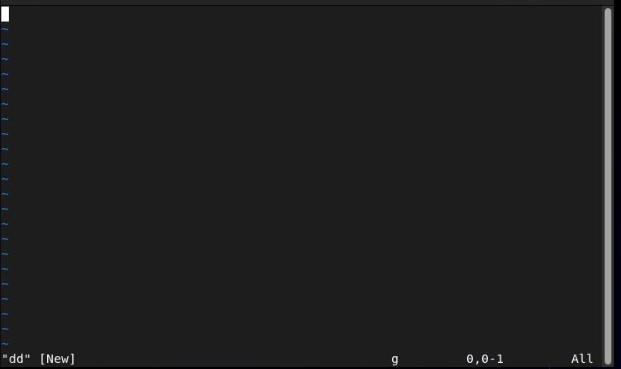
To quit

**Command:**

:dd

**Output:**

Delete line

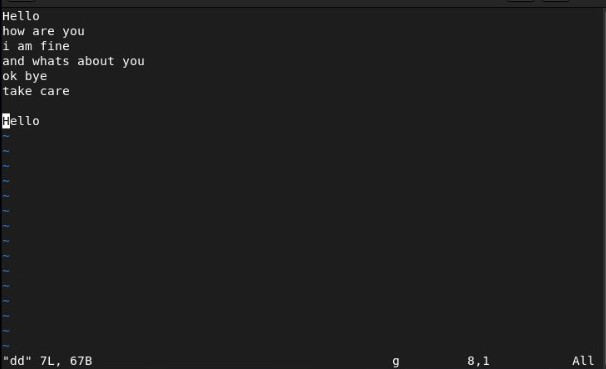


**Command:**

:yy :v

**Output:**

Copy and Paste

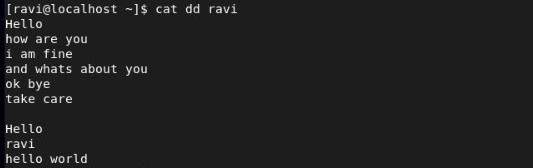


**Command:**

Cat dd ravi

**Output:**

Show delete and new both data



**Command:**

<n>dd

**Output:**

Delete the specified N number of lines



**Command:**

Cat dd ravi

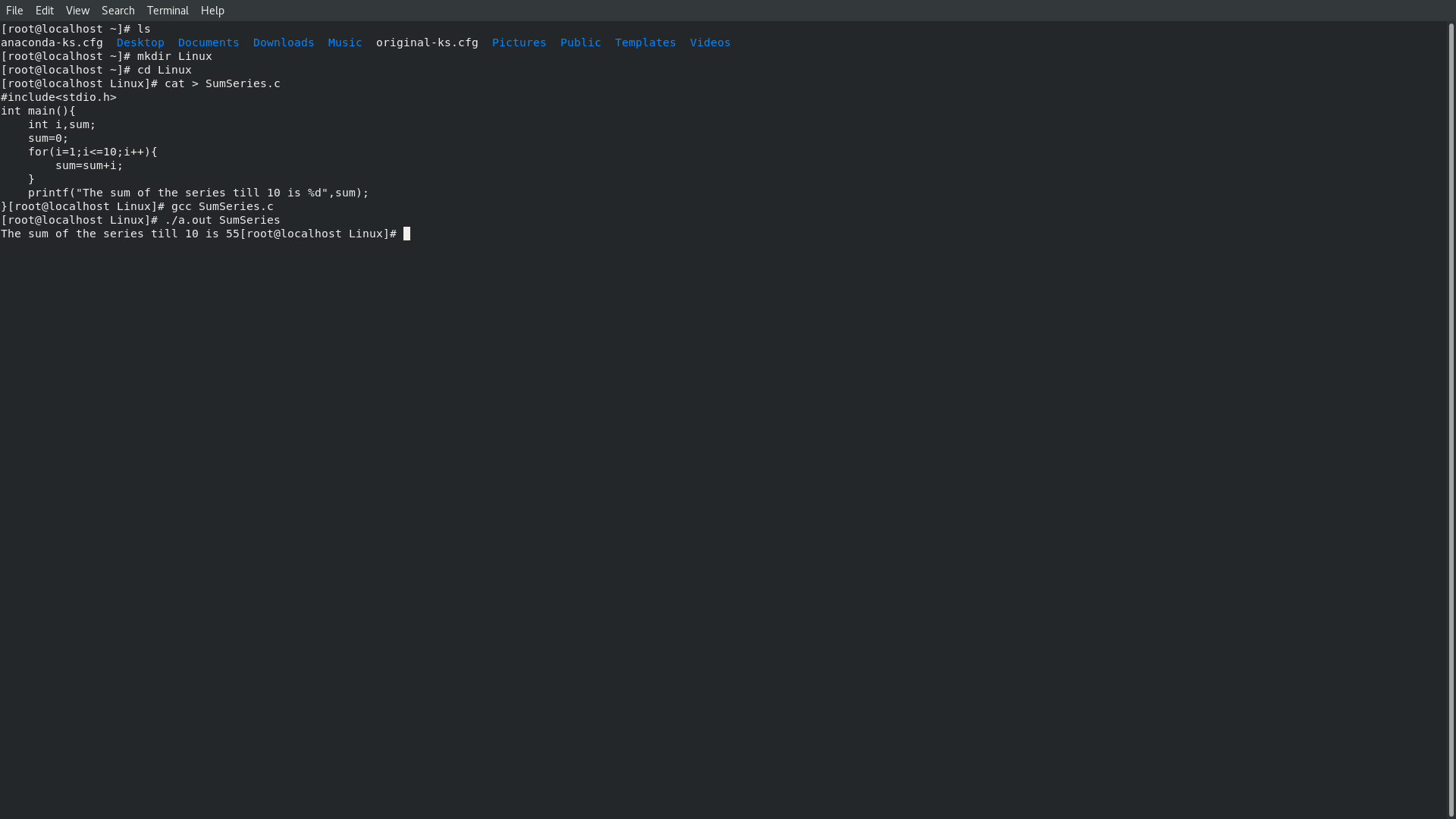
**Output:**

Delete 4 lines



**Programs in Linux**

* 1. Write a Linux program to print the sum of series till 10.



* **Step 1:** Create a .c file using cat command:
* **Step 2:** Enter the line of code for the respective program. (In this case, it is for printing the sum of series till 10.
* **Step 3:** Compile the code using the command: gcc [filename].
* **Step 4:** Display the output of the code using the ./a.out [filename] command.

SYNTAX:

cat > SumSeries.c

#include<stdio.h>

int main(){

int i, sum;

sum=0;

for(i=0;i<=10;i++){

sum=sum+i;

}

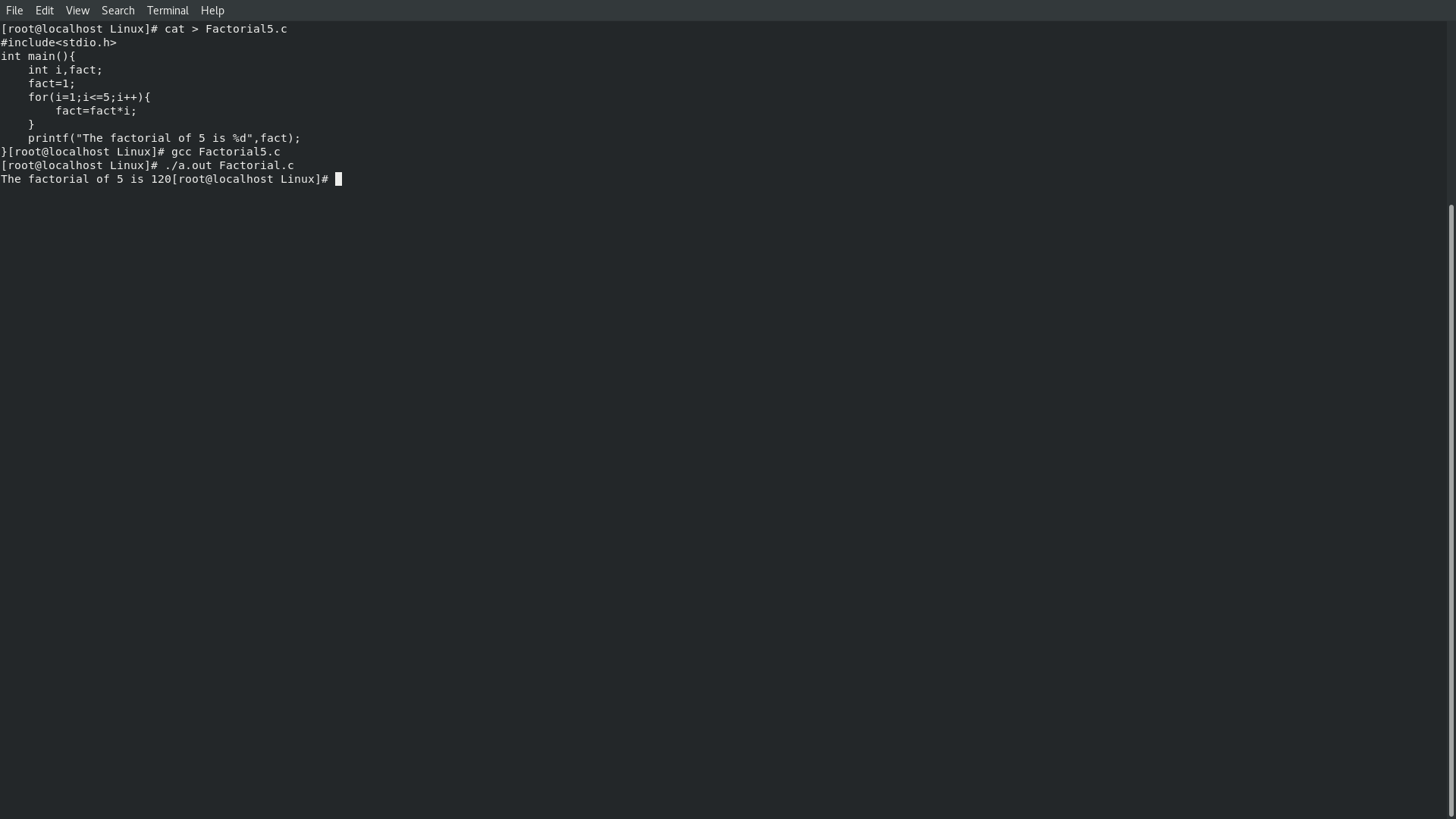
printf(“The sum of the series till 10 is %d”, sum);

}

CTRL + D to save

gcc SumSeries.c

./a.out SumSeries.c

* 1. Write a Linux program to find the factorial of 5.
  + **Step 1:** Create a .c file using cat command:
  + **Step 2:** Enter the line of code for the respective program. (In this case, it is for printing the factorial of 5.
  + **Step 3:** Compile the code using the command: gcc [filename].
  + **Step 4:** Display the output of the code using the ./a.out [filename] command.

SYNTAX:

cat > Factorial5.c

#include<stdio.h>

int main(){

int i,fact;

fact = 1;

for(i=0;i<=5;i++){

fact=fact\*i;

}

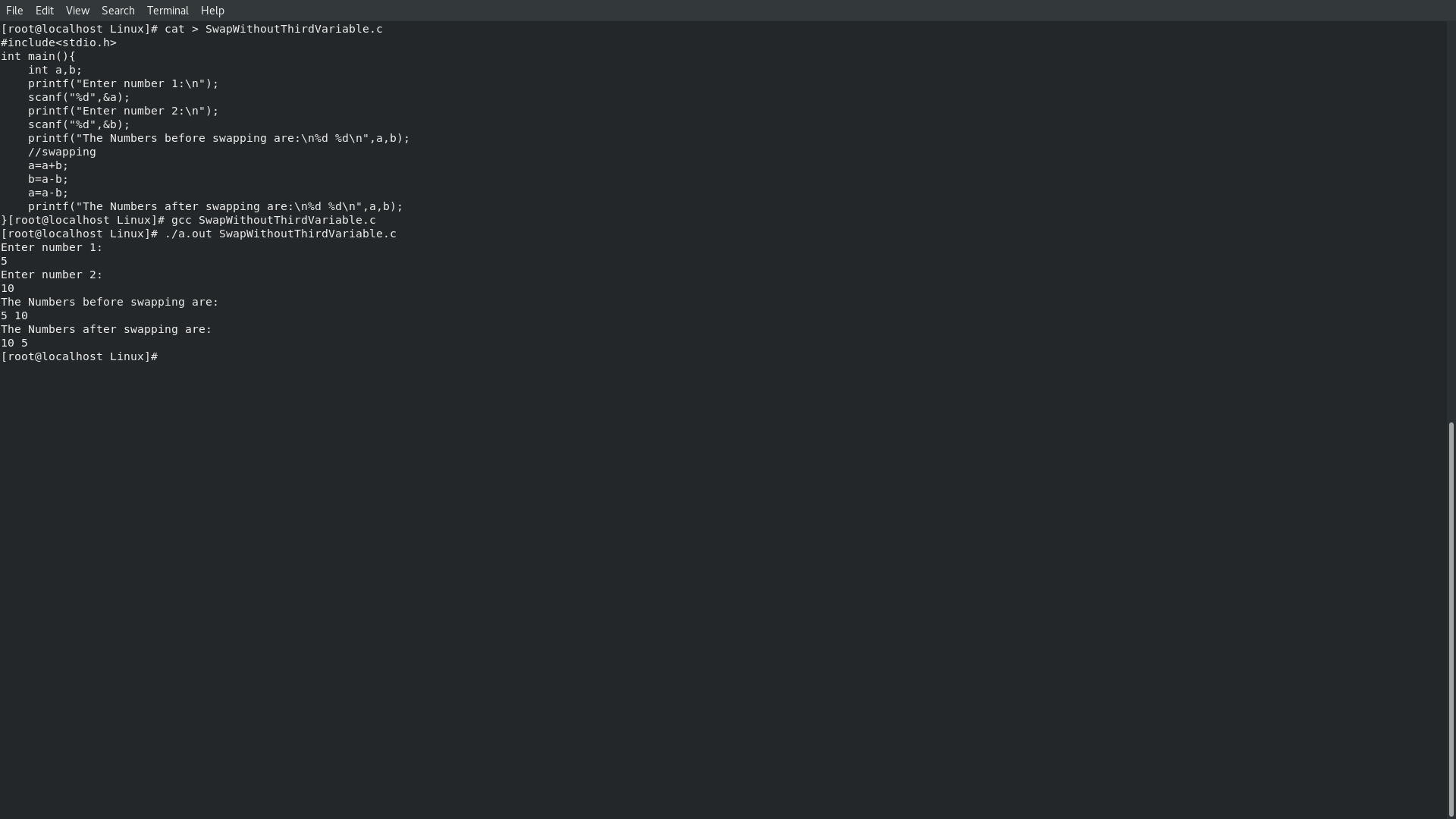
printf(“The factorial of 5 is %d”,fact);

}

CTRL + D to save

gcc Factorial5.c

./a.out Factorial5.c

* 1. Write a Linux program to swap two numbers without using third variable.
  + **Step 1:** Create a .c file using cat command:
  + **Step 2:** Enter the line of code for the respective program. (In this case, it is for swapping two numbers.
  + **Step 3:** Compile the code using the command: gcc [filename].
  + **Step 4:** Display the output of the code using the ./a.out [filename] command.

SYNTAX:

cat > SwapWithoutThirdVariable.c

#include<stdio.h>

int main(){

int a,b;

printf(“Enter number 1:\n”);

scanf(“%d”,&a);

printf(“Enter number 2:\n”);

scanf(“%d”,&b);

printf(“The numbers before swapping are: %d %d\n”,a,b);

a = a+b;

b = a-b;

a = a-b;

printf(“The numbers after swapping are: %d %d\n”,a,b);

}

CTRL + D to save

gcc SwappingWithoutThirdVariable.c

./a.out SwappingWithoutThirdVariable.c