## Experiment No. 4

**Title:** Pipes and filters in Linux

**Batch: B2** **Roll No: 16010420117** **Experiment No: 4**

**Aim:** To implement pipes and filters in Linux.

---

**Resources needed:** Any open source OS/CoCalc online editor

---

**Theory:**

**Pre lab/Prior concepts:**

**I/O Redirection**

In this expriement, we will explore a powerful feature used by many command line programs called input/output redirection. As we have seen, many commands such as ls print their output on the display. This does not have to be the case, however. By using some special notation we can redirect the output of many commands to files, devices, and even to the input of other commands.

Standard Output

Most command line programs that display their results do so by sending their results to a facility called standard output. By default, standard output directs its contents to the display. To redirect standard output to a file, the ">" character is used like this: [me@linuxbox me]$ ls > file_list.txt

In this example, the ls command is executed and the results are written in a file named file_list.txt. Since the output of ls was redirected to the file, no results appear on the display.

Each time the command above is repeated, file_list.txt is overwritten (from the beginning) with the output of the command ls. If you want the new results to be appended to the file instead, use ">>" like this:

[me@linuxbox me]$ ls >> file_list.txt

When the results are appended, the new results are added to the end of the file, thus making the file longer each time the command is repeated. If the file does not exist when you attempt to append the redirected output, the file will be created.

Standard Input

Many commands can accept input from a facility called standard input. By default,

standard input gets its contents from the keyboard, but like standard output, it can be redirected. To redirect standard input from a file instead of the keyboard, the "<" character is used like this:

[me@linuxbox me]$ sort < file_list.txt

In the above example we used the sort command to process the contents of file_list.txt. The results are output on the display since the standard output is not redirected in this example. We could redirect standard output to another file like this:

[me@linuxbox me]$ sort < file_list.txt > sorted_file_list.txt

As you can see, a command can have both its input and output redirected. Be aware that the order of the redirection does not matter. The only requirement is that the redirection operators (the "<" and ">") must appear after the other options and arguments in the command.

**Pipe**

A pipe is a way to connect the output of one program to the input of another program without any temporary file.

**Pipe Defined as:**

"A pipe is nothing but a temporary storage place where the output of one command is stored and then passed as the input for second command. Pipes are used to run more than two commands ( Multiplecommands) from same command line."

**Syntax:**

command1 | command2

example :

me@linuxbox me]$ ls -l | less

In this example, the output of the ls command is fed into less. By using this "| less" trick, you can make any command have scrolling output. I use this technique all the time.By connecting commands together, you can acomplish amazing feats. Here are some examples you'll want to try:

**Command What it does**

ls -lt | head Displays the 10 newest files in the current directory.

du | sort -nr Displays a list of directories and how much space they consume, sorted from the largest to the smallest.

find . -type f -print | wc -l Displays the total number of files in the current working directory and all of its subdirectories.

**Filters**

One class of programs you can use with pipes is called filters. Filters take standard input and perform an operation upon it and send the results to standard output. In this way, they can be used to process information in powerful ways. Here are some of the common programs that can act as filters:

**Command What it does**

sort Sorts standard input then outputs the sorted result on standard output.

uniq Given a sorted stream of data from standard input, it removes duplicate lines of data (i.e., it makes sure that every line is unique).

grep Examines each line of data it receives from standard input and outputs every line that contains a specified pattern of characters.

fmt Reads text from standard input, then outputs formatted text on standard output.

pr Takes text input from standard input and splits the data into pages with page breaks, headers and footers in preparation for printing.

head Outputs the first few lines of its input. Useful for getting the header of a file.

tail Outputs the last few lines of its input. Useful for things like getting the most recent entries from a log file.

tr Translates characters. Can be used to perform tasks such as upper/lowercase conversions or changing line termination characters from one type to another (for example, converting DOS text files into Unix style text files).

sed Stream editor. Can perform more sophisticated text translations than tr.

awk An entire programming language designed for constructing filters. Extremely

powerful.

cut Cuts specific characters or fields from a file with options

paste Creates either rows or columns of data that are combined from two separate files.

**Performing tasks with pipes**

1. Printing from the command line. Linux provides a program called lpr that accepts standard input and sends it to the printer. It is often used with pipes and filters. Here are a couple of examples:

```
cat poorly_formatted_report.txt | fmt | pr | lpr
cat unsorted_list_with_dupes.txt | sort | uniq | pr | lpr
```

In the first example, we use cat to read the file and output it to standard output, which is piped into the standard input of fmt. fmt formats the text into neat paragraphs and outputs it to standard output, which is piped into the standard input of pr. pr splits the text neatly into pages and outputs it to standard output, which is piped into the standard input of lpr. lpr takes its standard input and sends it to the printer.

The second example starts with an unsorted list of data with duplicate entries. First, **cat sends the list into sort which sorts it and feeds it into uniq which removes any** duplicates. Next pr and lpr are used to paginate and print the list.

2. Viewing the contents of tar files Often you will see software distributed as a gzipped tar file. This is a traditional Unix style tape archive file (created with tar) that has been compressed with gzip. You can recognize these files by their traditional file extensions, ".tar.gz" or ".tgz". You can use the following command to view the directory of such a file on a Linux system:

```
tar tzvf name_of_file.tar.gz | less
```

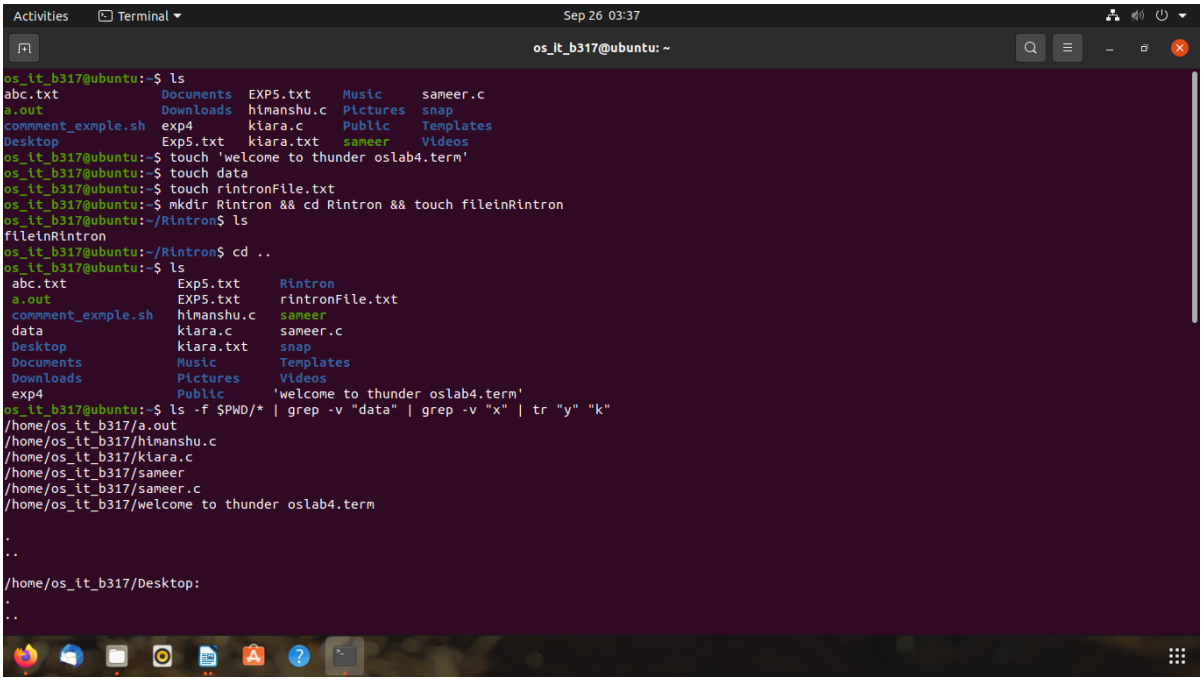
**Activities:**

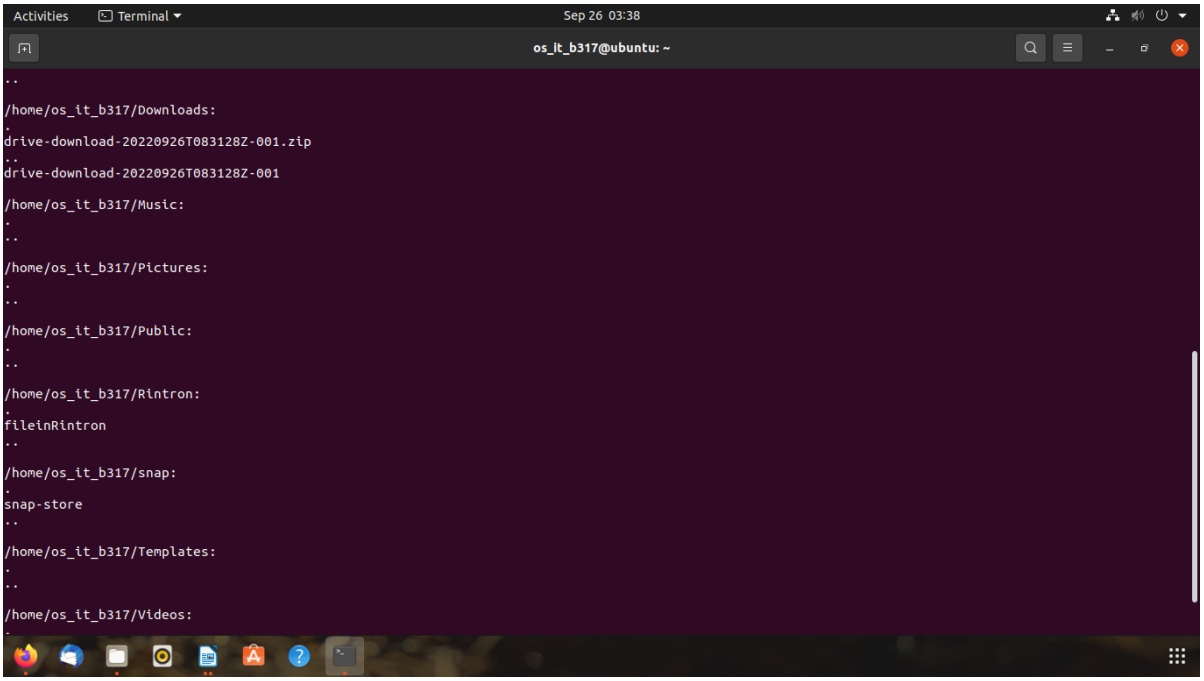
**Question 1:** Write command which produces a list of all the files on the system, such that:

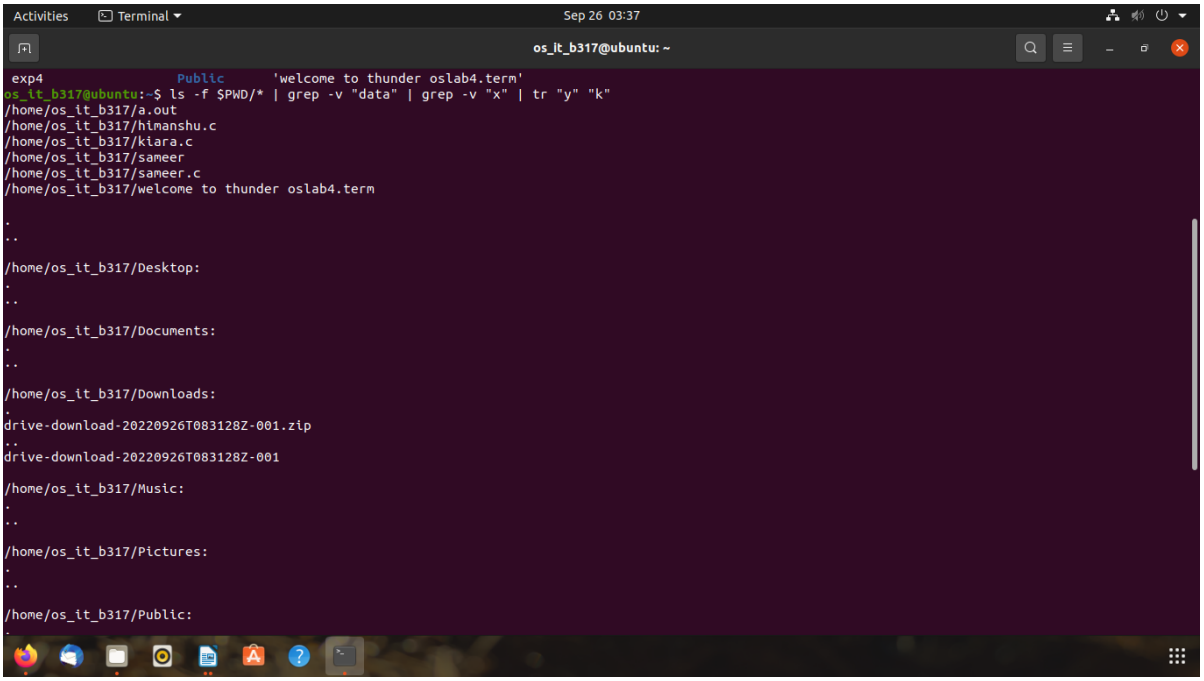
0. their full pathname does not contain the word data
1. their filename does not contain the letter x

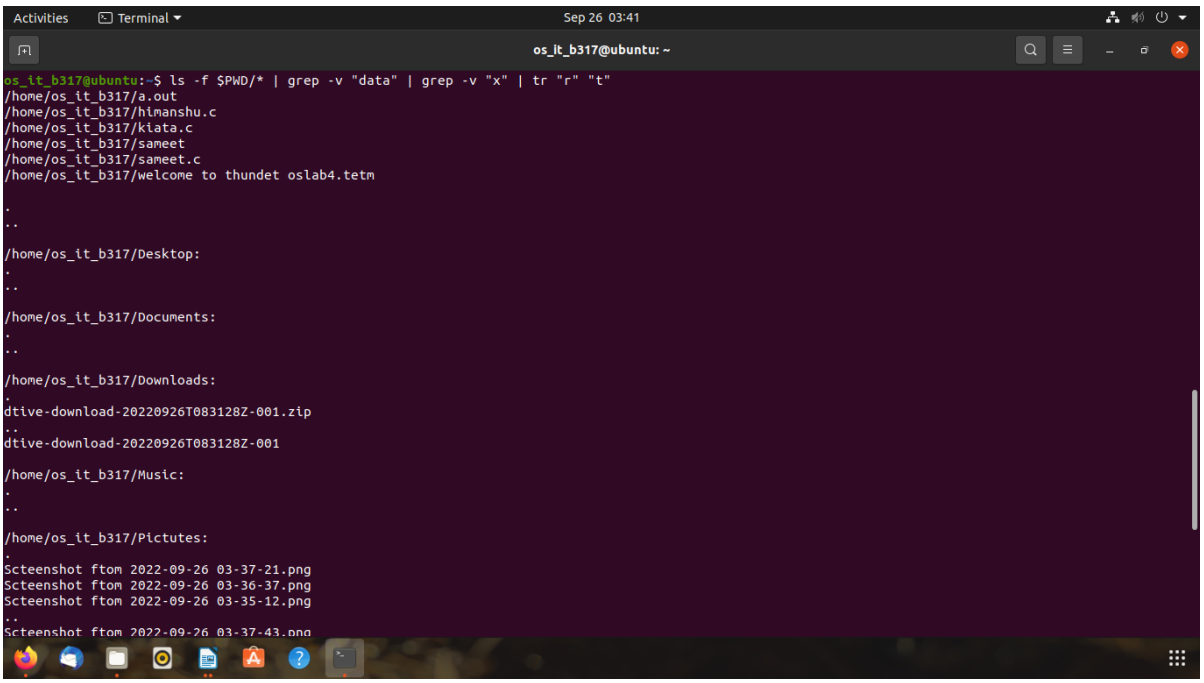
2. the script is Y2K compliant, so all `y's have been replaced by `k's.

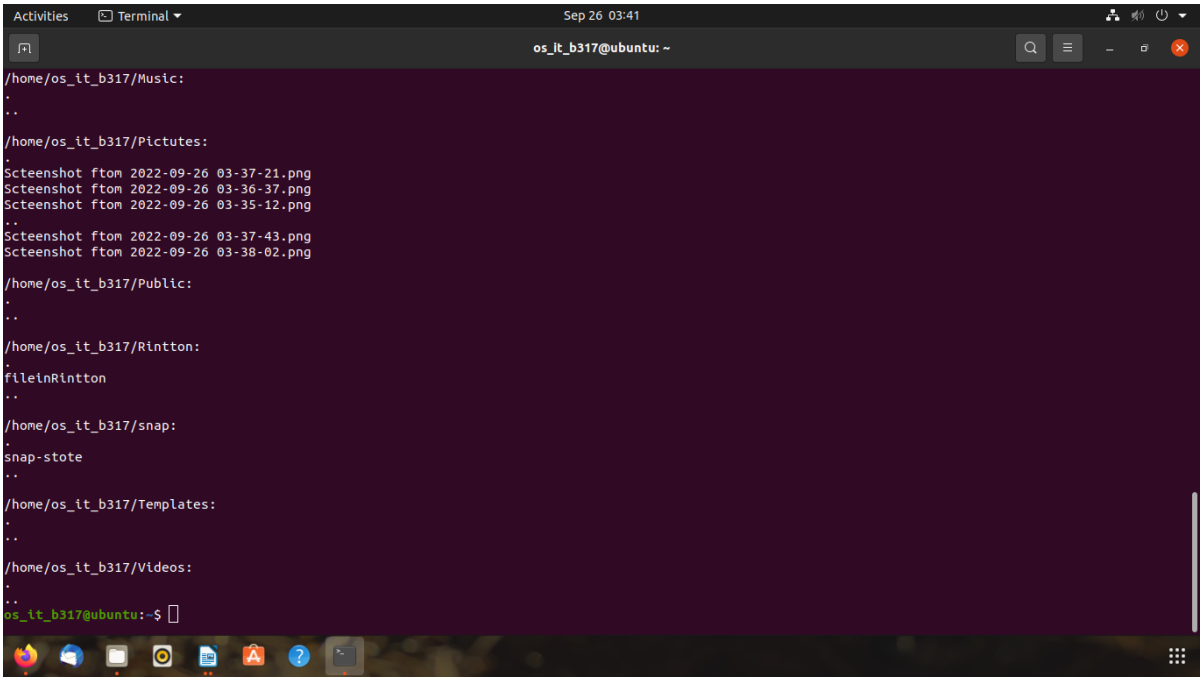
At the end, print out the number of files that were found.




Activities
Terminal
Sep 26 03:37
os_it_b317@ubuntu: ~
exp4 Public 'welcome to thunder oslab4.term'
os_it_b317@ubuntu:~$ ls -f $PWD/* | grep -v "data" | grep -v "x" | tr "y" "k"
/home/os_it_b317/a.out
/home/os_it_b317/himanshu.c
/home/os_it_b317/kiara.c
/home/os_it_b317/sameer
/home/os_it_b317/sameer.c
/home/os_it_b317/welcome to thunder oslab4.term
.
..
/home/os_it_b317/Desktop:
.
..
/home/os_it_b317/Documents:
.
..
/home/os_it_b317/Downloads:
.
drive-download-20220926T083128Z-001.zip
..
drive-download-20220926T083128Z-001
/home/os_it_b317/Music:
.
..
/home/os_it_b317/Pictures:
.
..
/home/os_it_b317/Public:


Activities
Terminal
Sep 26 03:41
os_it_b317@ubuntu: ~
os_it_b317@ubuntu:~$ ls -f $PWD/* | grep -v "data" | grep -v "x" | tr "r" "t"
/home/os_it_b317/a.out
/home/os_it_b317/himanshu.c
/home/os_it_b317/kiata.c
/home/os_it_b317/sameet
/home/os_it_b317/sameet.c
/home/os_it_b317/welcome to thundet oslab4.tetm
.
..
/home/os_it_b317/Desktop:
.
..
/home/os_it_b317/Documents:
.
..
/home/os_it_b317/Downloads:
.
dtive-download-20220926T083128Z-001.zip
..
dtive-download-20220926T083128Z-001
/home/os_it_b317/Music:
.
..
/home/os_it_b317/Pictutes:
.
Scteenshot ftom 2022-09-26 03-37-21.png
Scteenshot ftom 2022-09-26 03-36-37.png
Scteenshot ftom 2022-09-26 03-35-12.png
..

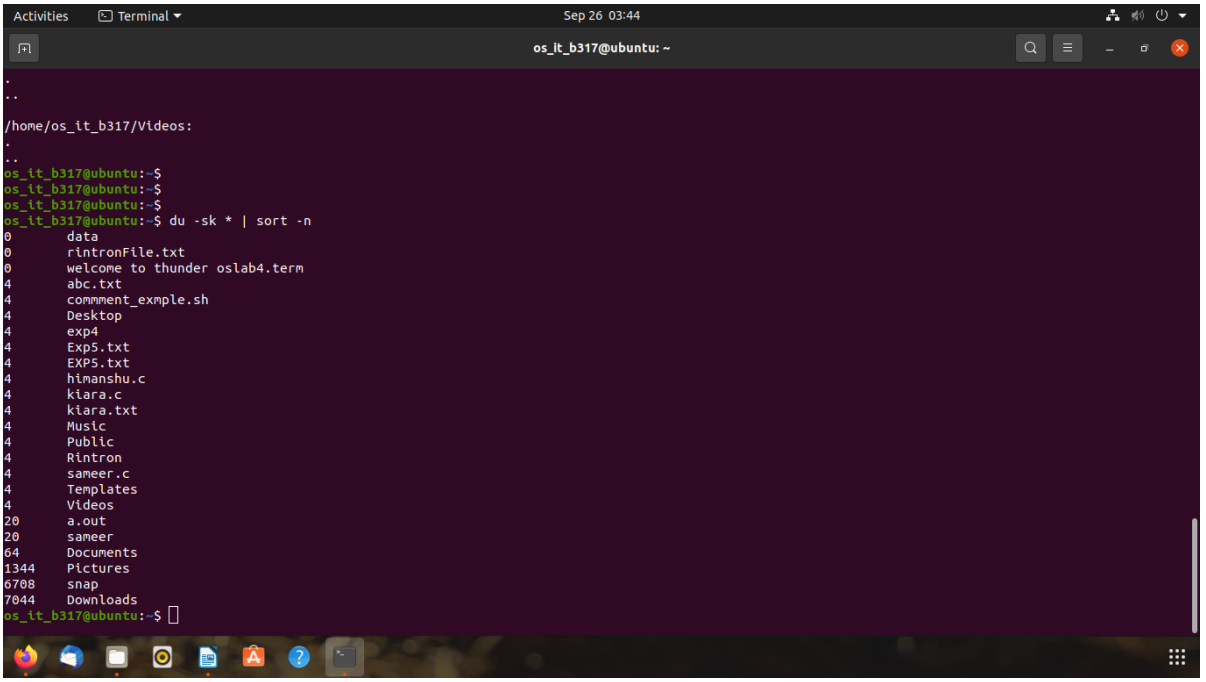



**# Here in this whole List we will not find the files/Folders whose name contain the word “data” and the letter “x” (“data” and “rintronFile.txt” are not displayed)**

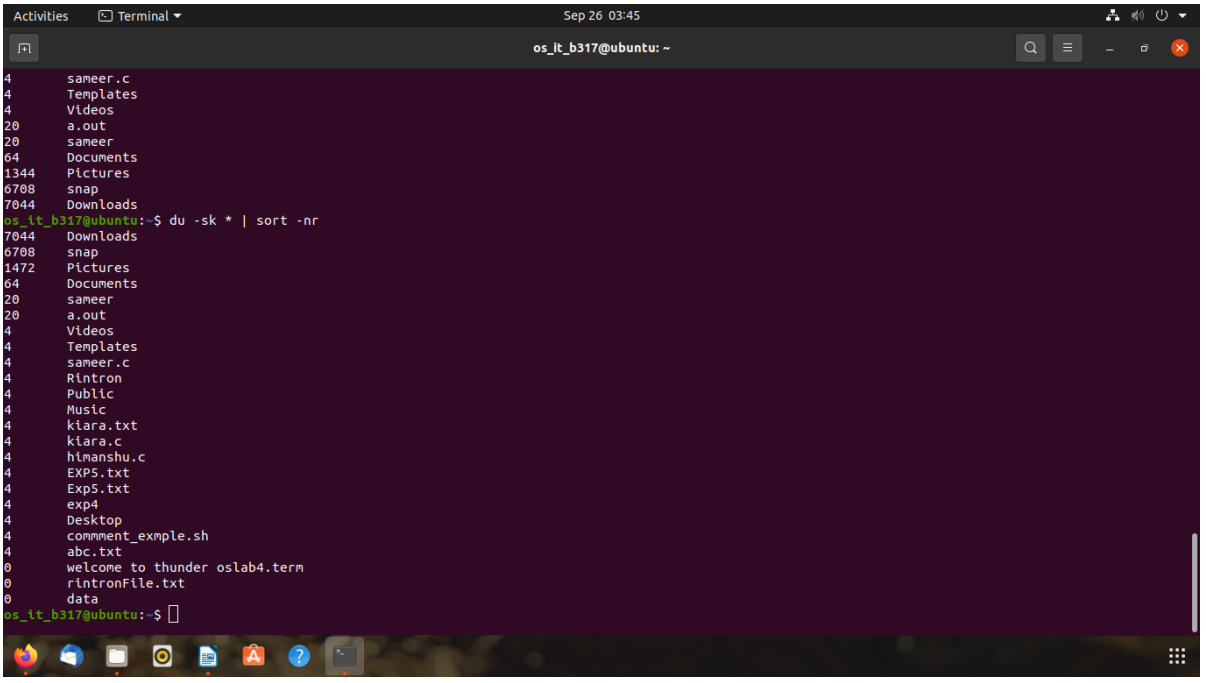
**Question 2.** Write commands that will list the size of each directory given on the command line, sorted by size. The size includes disk space used by the directory and all thefiles and subdirectories inside it. The script should take options to sort with smallest first, and with largest first.

Ans:

**#Sorting in Ascending order**



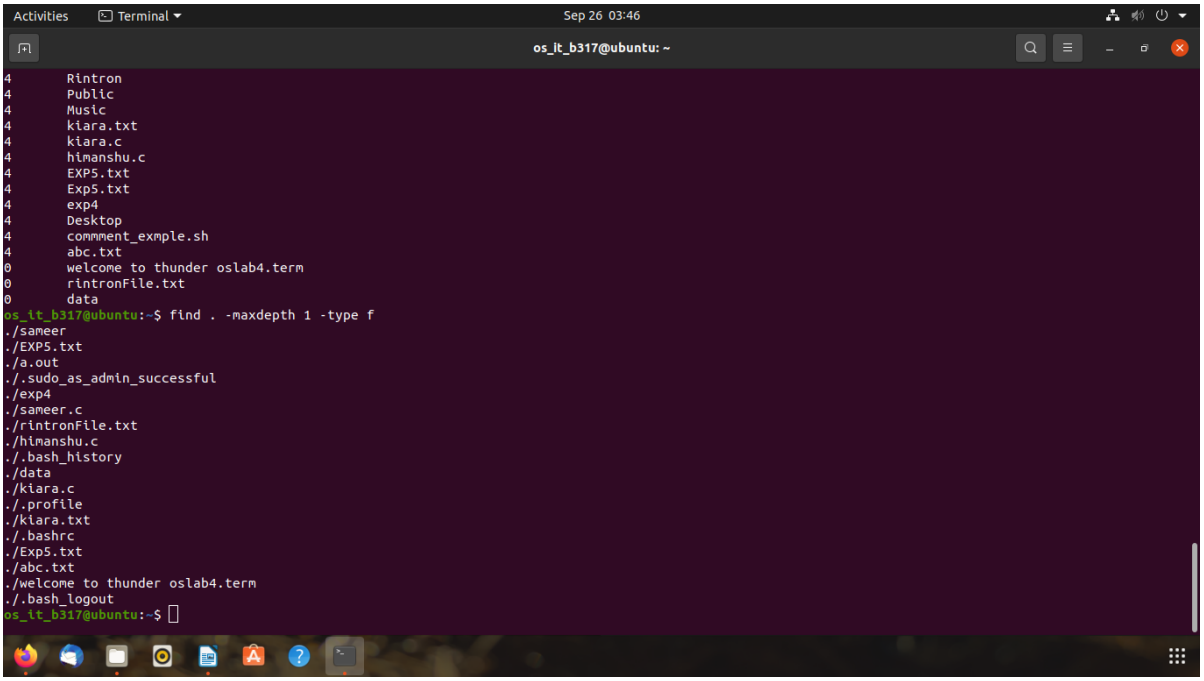

**#Sorting in Descending order**



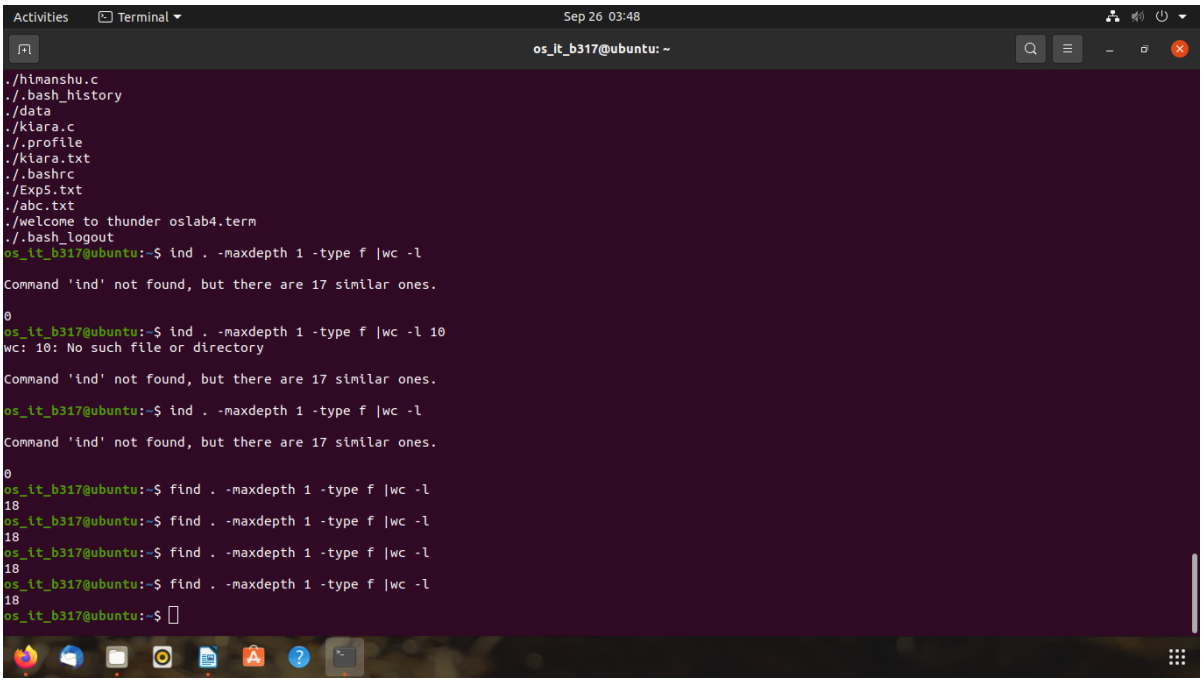
**Question 3.** Write a command to count total number of the files in present working directory.

Ans:

**#printing the total number of files in current directory**

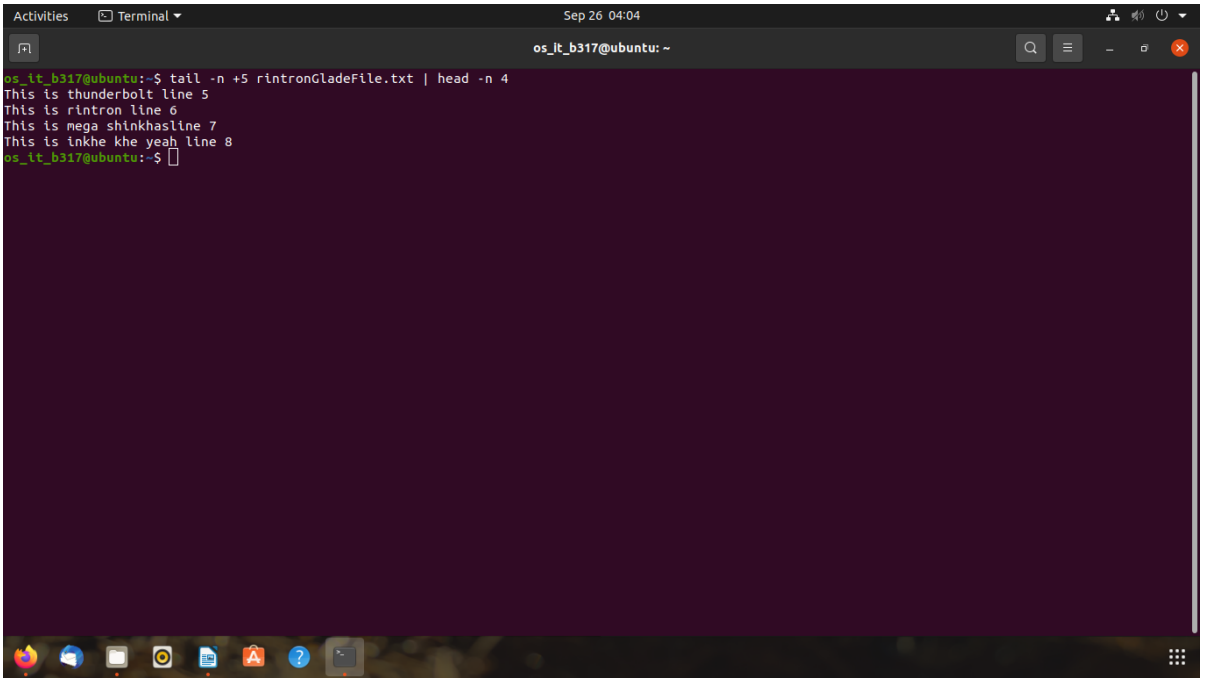



**#printing the total number of files in the directory**



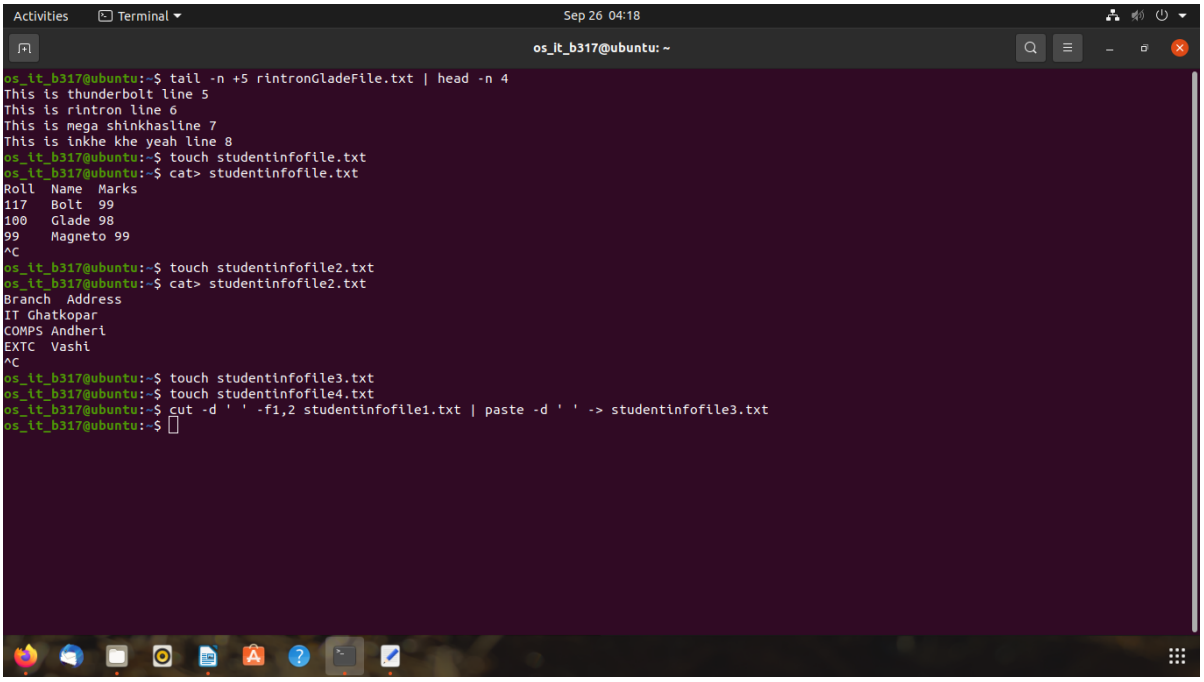
**Question 4.** Write command to extract 4 line starting from line number 5 to line number 8 from a file which contains 10 lines in it.

Ans:



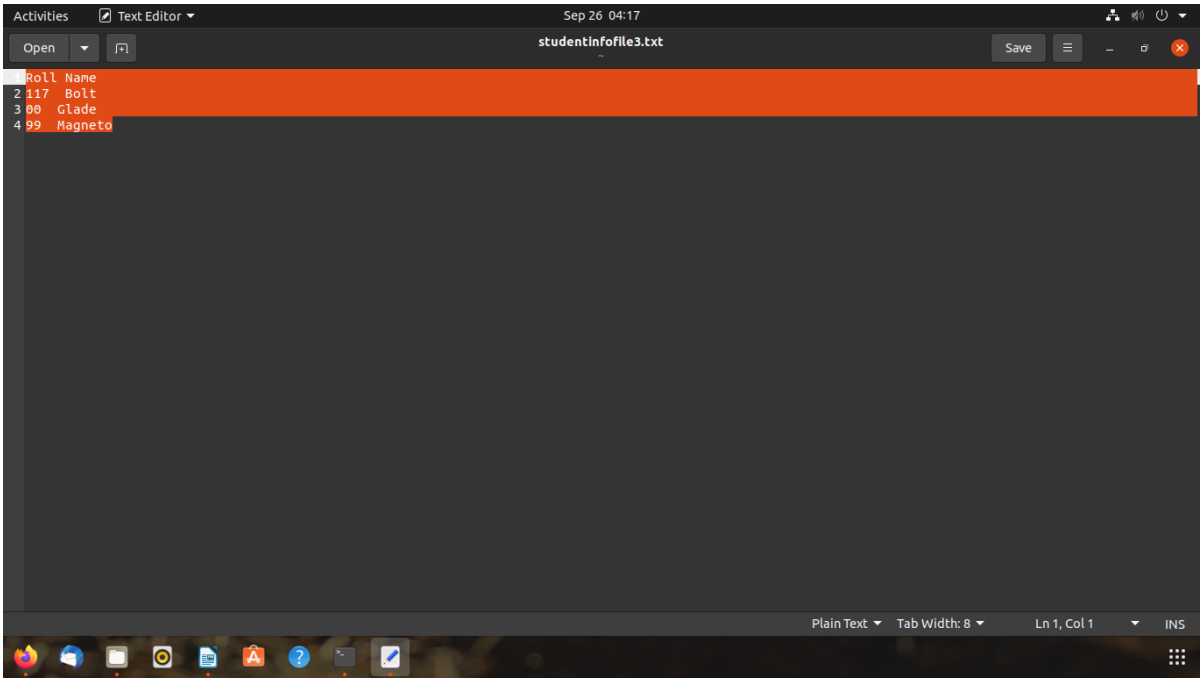
**Question 5.** Create a file containing rollno, name and marks of 3 students and another file containing branch and address of same 3 students. Use space as delimiter in both files. Write commands to cut rollno and name files first file and address field from second file and paste result in new file and display it.

Ans:

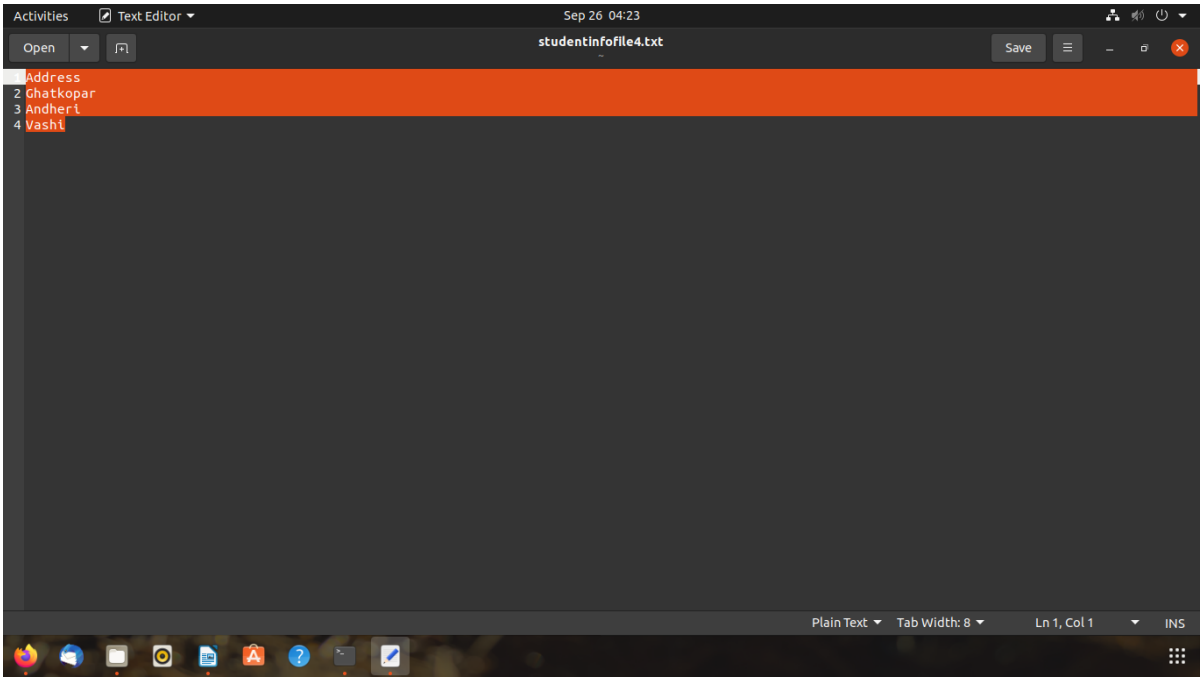



**#Here the columns 1 and 2( Roll No and Name ) are cut and pasted from file "studentinfofile1.txt" to a new file "studentinfofile3.txt"**

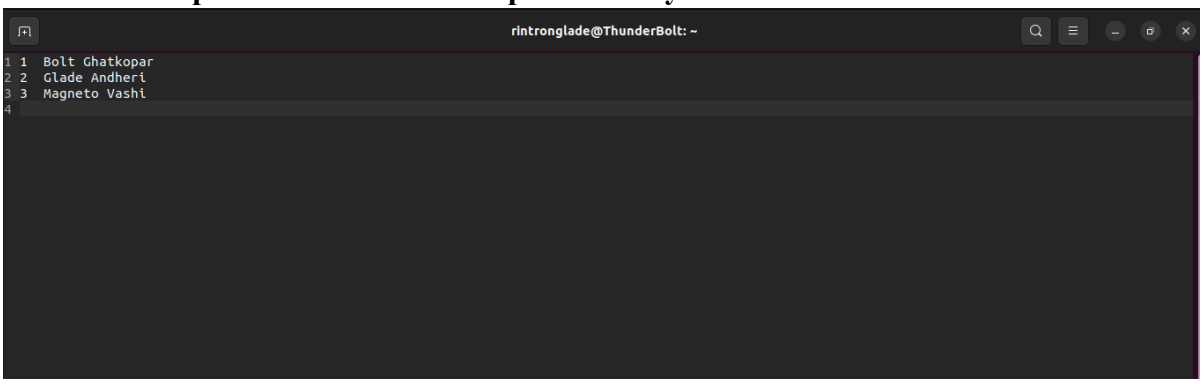
**#Contents of file studentinfofile3.txt**



**#Here the column 2( Address ) is cut and pasted from file "studentinfofile2.txt" to a new file "studentinfofile4.txt"**




**# We can also paste the combined output directely to a new file "studentinfofile5.txt".**



**Results: Perform the activity. No snapshots to be taken.**
**The assignment submitted should be e- media saved as <Roll No_Batch No_Date>**

**This file must contain on the top:**
**Name:**
**Roll No.**
**Exp No.**
**Batch:**
**Date:**

And Students have to upload this document electronically.

**Outcomes: CO4:** Demonstrate open source standards usage

**Conclusion: We learnt the implementation of pipes and filters in Linux and understood the use of various commands and functionalities related to them.**

<Students should write about the concluding remarks of the experiment themselves>.

---

**Grade: AA/AB/BB/BC/CC/CD/DD**

**Signature of faculty in-charge with date**

---

**References:**

**Books/ Journals/ Websites:**

1. Ri$_n$c$_d$hard Blum and Christine Bresnahan, “Linux Command Line & Shell Scripting”, II Edition edition, Wiley, 2012.