Experiment 14

Aim: Regular expressions, redirections: I/P, O/P, Error,> , >>.
Filters in Linux: Simple filters viz. more, less, wc, diff, sort, uniq, paste, cut, nl, tee, head, tail, tr

What is Redirection?

Redirection is a feature in Linux such that when executing a command, you can change the standard input/output devices. The basic workflow of any Linux command is that it takes an input and give an output.

- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.

Standard File Descriptors

Everything is a file in Linux and that includes input and output.

Each process can have 9 file descriptors opened at the same time. The file descriptors 0, 1, 2 are kept for the bash shell usage.

0	STDIN.
1	STDOUT.
2	STDERR.

You can use the above file descriptors to control input and output.

STDIN

STDIN stands for standard input which is the keyboard by default.

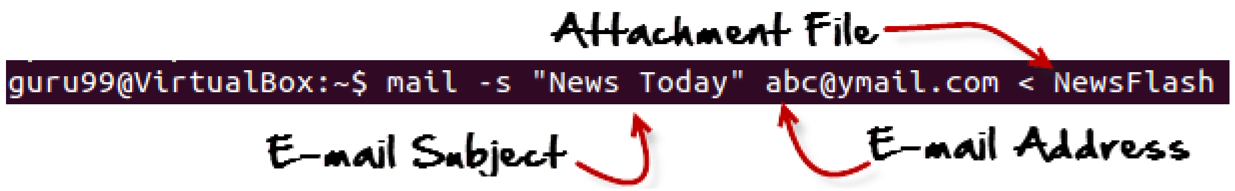
Input redirection

The '<' symbol is used for input(STDIN) redirection

Example: The mail program in Linux can help you send emails from the Terminal.

You can type the contents of the email using the standard device keyboard. But if you want to attach a File to email you can use the input re-direction operator in the following format.

```
Mail -s "Subject" to-address < Filename
```



```
guru99@VirtualBox:~$ mail -s "News Today" abc@ymail.com < NewsFlash
```

This would attach the file with the email, and it would be sent to the recipient.

The above examples were simple. Let's look at some advance re-direction techniques which make use of File Descriptors.

You can replace the STDIN which is the keyboard and replace it with a file by using the input redirect symbol (<), it sends the data as keyboard typing. No magic!!

When you type the cat command without anything, it accepts input from STDIN. Any line you type, the cat command prints that line to the screen.

STDOUT

This stands for the standard output which is the screen by default.

Output Redirection

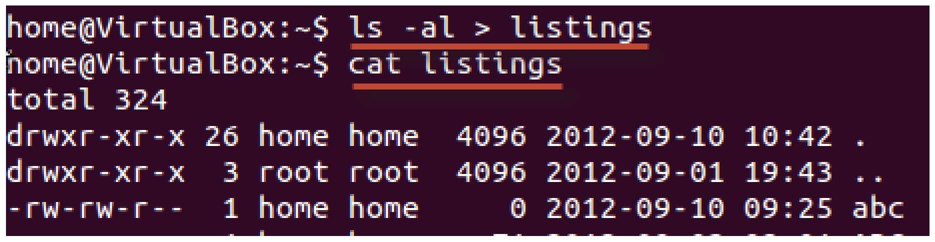
The '>' symbol is used for output (STDOUT) redirection.

> Output Redirection

Example:

```
ls -al > listings
```

Here the output of command ls -al is re-directed to file "listings" instead of your screen.



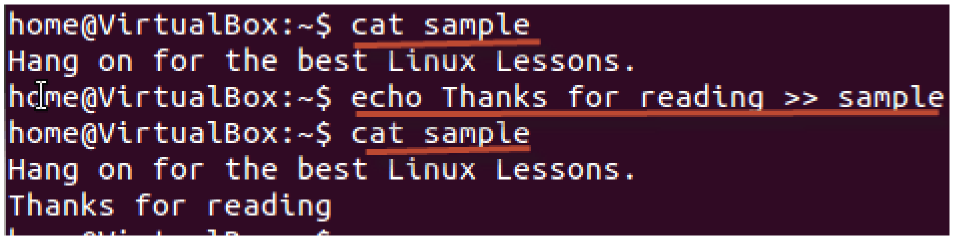
```
home@VirtualBox:~$ ls -al > listings
home@VirtualBox:~$ cat listings
total 324
drwxr-xr-x 26 home home   4096 2012-09-10 10:42 .
drwxr-xr-x  3 root root   4096 2012-09-01 19:43 ..
-rw-rw-r--  1 home home      0 2012-09-10 09:25 abc
```

Note: Use the correct file name while redirecting command output to a file. If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten."

If you do not want a file to be overwritten but want to add more content to an existing file, then you should use '>>' operator. You can redirect output to a file using the >> symbol. If we have a file contains data, you can append data to it using this symbol like this:

```
pwd >> myfile
```

The output generated by pwd is appended to myfile without deleting the existed content.



```
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
home@VirtualBox:~$ echo Thanks for reading >> sample
home@VirtualBox:~$ cat sample
Hang on for the best Linux Lessons.
Thanks for reading
```

You can redirect standard output, to not just files, but also devices!

```
$ cat music.mp3 > /dev/audio
```

The cat command reads the file music.mp3 and sends the output to /dev/audio which is the audio device. If the sound configurations in your PC are correct, this command will play the file music.mp3

The following command tries to redirect the output to a file using > symbol.

```
ls –l xfile > myfile
```

I have no file called xfile on my PC, and that generates an error which is sent to STDERR.

STDERR

This file descriptor is the standard error output of the shell which is sent to the screen by default.

If you need to redirect the errors to a log file instead of sending it to the screen, you can redirect errors using the redirection symbol.

Redirecting Errors

We can redirect the errors by placing the file descriptor which is 2 before the redirection symbol like this:

```
ls -l xfile 2>myfile
cat ./myfile
```

As you can see, the error now is in the file and nothing on the screen.

Redirecting Errors and Normal Output
To redirect errors and the normal output, you have to precede each with the proper file descriptor like this:

```
ls –l myfile xfile anotherfile 2> errorcontent 1> correctcontent
```

The ls command result is sent to the correctcontent file using the 1> symbol. And error messages were sent to the errorcontent file using the 2> symbol.

Method 1: You can redirect normal output and errors to the same file using &> symbol like this:

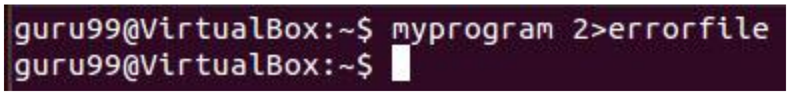
```
ls –l myfile xfile anotherfile &> content
```

All errors and normal output are redirected to file named content.
Method 2 : Syntax to redirect both output (stdout) and errors (stderr) to same file
The syntax is:

```
command1 > everything.txt 2>&1 command1 -arg > everything.txt 2>&1
```

Example 1

```
$ myprogram 2>errorsfile
```



```
guru99@VirtualBox:~$ myprogram 2>errorfile
guru99@VirtualBox:~$ 
```

Above we are executing a program names myprogram.

The file descriptor for standard error is 2.

Using "2>" we re-direct the error output to a file named "errorfile"

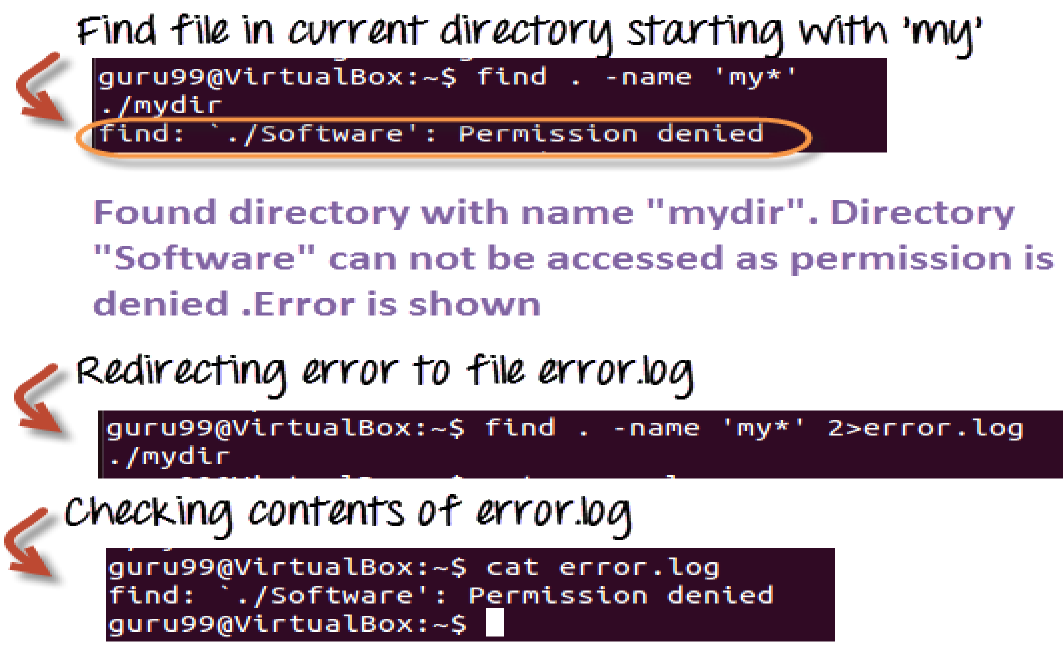
Thus, program output is not cluttered with errors.

Example 2

Here is another example which uses find statement -

```
find . -name 'my*' 2>error.log
```

Using the "find" command, we are searching the "." current directory for a file with "name" starting with "my"


Find file in current directory starting with 'my'
Found directory with name "mydir". Directory "Software" can not be accessed as permission is denied .Error is shown
Redirecting error to file error.log
Checking contents of error.log

```
guru99@VirtualBox:~$ find . -name 'my*'
./mydir
find: `./Software': Permission denied
```

```
guru99@VirtualBox:~$ find . -name 'my*' 2>error.log
./mydir
```

```
guru99@VirtualBox:~$ cat error.log
find: `./Software': Permission denied
guru99@VirtualBox:~$
```

Let's see a more complex example,

Server Administrators frequently, list directories and store both error and standard output into a file, which can be processed later. Here is the command.

```
ls Documents ABC> dirlist 2>&1
```

Here,

- which writes the output from one file to the input of another file. 2>&1 means that STDERR redirects to the target of STDOUT (which is the file dirlist).
- We are redirecting error output to standard output which in turn is being redirected to file dirlist. Hence, both the output is written to file dirlist.
- Each file in Linux has a corresponding File Descriptor associated with it The keyboard is the standard input device while your screen is the standard output device

- ">" is the output redirection operator. ">>" appends output to an existing file
- "<" is the input redirection operator
- ">&"re-directs output of one file to another.
- You can re-direct error using its corresponding File Descriptor 2.

How to Use I/O Redirection Using Pipes

To redirect the output of one command as input of another, you can use pipes, this is a powerful means of building useful command lines for complex operations.

For example, the command below will list the top five recently modified files.

```
$ ls -lt | head -n 5
```

What is a Pipe in Linux?

The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next. In short, the output of each process directly as input to the next one likes a pipeline. The symbol '|' denotes a pipe.

Pipes help you mash-up two or more commands at the same time and run them consecutively. You can use powerful commands which can perform complex tasks in a jiffy.

Let us understand this with an example.

When you use 'cat' command to view a file which spans multiple pages, the prompt quickly jumps to the last page of the file, and you do not see the content in the middle.

To avoid this, you can pipe the output of the 'cat' command to 'less' which will show you only one scroll length of content at a time.

```
cat filename | less
```

Use sort and uniq command to sort a file and print unique values.

```
$ sort record.txt | uniq
```

This will sort the given file and print the unique values only.

Use head and tail to print lines in a particular range in a file.

```
$ cat sample2.txt | head -7 | tail -5
```

This command selects first 7 lines and last 5 lines from the file and print those lines which are common to both of them.

Use ls and find to list and print all lines matching a particular pattern in matching files.

```
$ ls -l | find ./ -type f -name "*.txt" -exec grep "program" {} \;
```

This command selects files with **.txt** extension in the given directory and search for pattern like "program" in the above example and prints those ine which have program in them.

Use cat, grep, tee and wc command to read the particular entry from user and store in a file and print line count.

```
$ cat result.txt | grep "Rajat Dua" | tee file2.txt | wc -l
```

This command select **Rajat Dua** and store them in file2.txt and print total number of lines matching **Rajat Dua**

More command in Linux

As 'cat' command displays the file content. Same way 'more' command also displays the content of a file. Only difference is that, in case of larger files, 'cat' command output will scroll off your screen while 'more' command displays output one screenful at a time.

Following keys are used in 'more' command to scroll the page:

- Enter key: To scroll down page line by line.
- Space bar: To go to next page.
- b key: To go to the backward page.
- / key: Lets you search the string.

Syntax:

```
more <file name>
```

How to open a file at a line number

To open a file at a line number pass the + option along with a line number

```
more +2654 /usr/share/dict/british
```

The more viewer will open at line 2654.

```
more +5 doc_30.txt
```

How to search within more

To search within `more` press the `/` key followed by the phrase to be searched for. The search pattern accepts regular expressions. The following searches for the phrase 'eat'. This will search lines for instances of the phrases and scroll the page to the first occurrence.

```
more /usr/share/dict/british
/eat
```

In this file the first occurrence is the word 'aleatory'. To search for words starting with 'eat' a regular expression may be used.

```
more /usr/share/dict/british
/^eat
```

Now the word 'eat' is found.

To search for the next occurrence of a regular expression press the `n` key.

```
more /usr/share/dict/british
/eat
# first match is aleatory
# press 'n'
# second match is amphitheatre
```

Search string (+/string) in file

```
more -10 -d +/string filename
```

-s : This option squeezes multiple blank lines into one single blank line.
Example:

```
more -s sample.txt
```

Using more to Read Long Outputs: We use more command after a pipe to see long outputs. For example, seeing log files, etc.

```
cat a.txt | more
```

What is the less command in Linux?

With less, you can read large text files without cluttering your terminal screen. You can also search for text and monitor files in real time with it.

Some people prefer using Vim for reading large text files. But less is faster than Vim or other such text editors because it doesn't read the entire file before starting. Since less is 'read only', you don't have the risk of accidentally editing the files you are viewing.

The syntax for the less command is extremely simple:

```
less filename
```

There are numerous options with less command but it's better to focus on the practical usage that will be more useful for you.

Practical examples of less command in Linux

It is better to work with a big file to understand the usage of the less command.

1. View a text file with less

As showed in the syntax, you can use the less command to view a file in the following fashion:

```
less [option]<filename>
```

2. Exit from less

If you are not used to of less command, you might struggle to find how to exit less. Trust me it's not at all complicated. Simply ***press 'q' at any given point to exit from less***.

3. Moving around in less

- Up arrow – Move one line up
- Down arrow – Move one line down
- Space or PgDn – Move one page down
- b or PgUp – Move one page up
- g – Move to the beginning of the file

- G – Move to the end of the file
- ng – Move to the nth line

4. Display line numbers with less

If you want to see the line numbers in the less command output, you can use the option N in the following manner:

```
less -N <filename>
```

5. Ignore case with less

By default, search in less is case sensitive. To ignore case, you can use less with -I option

```
less -I <filename>
```

If you forgot to use this option, don't worry. You can also press '-I' key combination before performing a search inside less.

6. View multiple files with less command

I'll be honest with you. This is not my favorite less command example but you can totally do this.

To open multiple files with less, simply input the file names one by one:

```
less <filename1> <filename2> <filename3>
```

You'll see that it lists the file name along with its position in the list of the files.

VIEWING MULTIPLE FILES WITH LESS
You can view other files in the list using these keys:

- :n – view the next file in the list
- :p – view the previous file in the list

7. To Starts up the file from the given number(less +number file_path):

```
less +4 sample.txt
```

8. Using less command with pipes
The less command can be used in conjugation with other commands using pipes. It is particularly useful when you know that the output of a certain command is going to be huge.

For example, the output of dmesg command is usually in thousands of lines. You don't want it to flood your screen and you won't be able to analyze the output as well. Pipe it with less and you'll have a more friendly way of reading this output.

```
dmesg | less
```

wc command in Linux with examples

wc stands for **word count**. As the name implies, it is mainly used for counting purpose.

- It is used to find out **number of lines**, **word count**, **byte and characters count** in the files specified in the file arguments.
- By default it displays **four-columnar output.**
- First column shows number of lines present in a file specified, second column shows number of words present in the file, third column shows number of characters present in file and fourth column itself is the file name which are given as argument.

A Basic Example of WC Command

The '**wc**' command without passing any parameter will display a basic result of "**tecmint.txt**' file. The three numbers shown below are **12** (**number of lines**), **16** (**number of words**) and **112** (**number of bytes**) of the file.

```
[root@tecmint ~]# wc tecmint.txt

12  16 112 tecmint.txt
```

The options below allow you to select which counts are printed.

- -l, --lines - Print the number of lines.
- -w, --words - Print the number of words.
- -m, --chars - Print the number of characters.
- -c, --bytes - Print the number of bytes.
- -L, --max-line-length - Print the length of the longest line.

Count the Number of Lines

The `wc` command is mostly used with the `-l` option to count only the number of lines in a text file. For example, to count the number of lines in the `/etc/passwd` file you would type:

```
wc -l /etc/passwd
```

The first column is the number of lines and the second one is the name of the file:

```
44 /etc/passwd
```

Count the Number of Words

To to count only the number of words in a text file use `wc -w` followed by the file name. The following example counts the number of words in the `~/Documents/file.txt` file:

```
wc -w /etc/passwd
```

The number of words is shown in the first column:

```
513 /home/linuxize/Documents/file.txt
```

Counting Files in the Current Directory

The `find` command passes a list of all files in the current directory with each file name on a single line to the `wc` command, which counts the number of lines and prints the result:

```
find . -type f | wc -l
```

Count Number of Bytes and Characters

When using options ‘-c‘ and ‘-m‘ with ‘wc‘ command will print the total number of bytes and characters respectively in a file.

```
[root@tecmint ~]# wc -c tecmint.txt
112 tecmint.txt
[root@tecmint ~]# wc -m tecmint.txt
112 tecmint.txt
```

Display Length of Longest Line

The ‘wc‘ command allow an argument ‘-L‘, it can be used to print out the length of longest (number of characters) line in a file. So, we have the longest character line (‘Scientific Linux‘) in a file.

```
[root@tecmint ~]# wc -L tecmint.txt
16 tecmint.txt
```

SORT command in Linux/Unix with examples

SORT command is used to sort a file, arranging the records in a particular order. By default, the sort command sorts file assuming the contents are ASCII. Using options in sort command, it can also be used to sort numerically.

- SORT command sorts the contents of a text file, line by line.
- sort is a standard command line program that prints the lines of its input or concatenation of all files listed in its argument list in sorted order.
- The sort command is a command line utility for sorting lines of text files. It supports sorting alphabetically, in reverse order, by number, by month and can also remove duplicates.
- The sort command can also sort by items not at the beginning of the line, ignore case sensitivity and return whether a file is sorted or not. Sorting is done based on one or more sort keys extracted from each line of input.
- By default, the entire input is taken as sort key. Blank space is the default field separator.

The sort command follows these features as stated below:

1. Lines starting with a number will appear before lines starting with a letter.
2. Lines starting with a letter that appears earlier in the alphabet will appear before lines starting with a letter that appears later in the alphabet.
3. Lines starting with a lowercase letter will appear before lines starting with the same letter in uppercase.

Examples

Suppose you create a data file with name file.txt

```
Command :
$ cat > file.txt
abhishek
chitransh
satish
rajan
naveen
divyam
harsh
```

Sorting a file: Now use the sort command

Syntax :

```
$ sort filename.txt
Command:
$ sort file.txt
Output :
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
```

Note: This command does not actually change the input file, i.e. file.txt.

Sort function with mix file i.e. uppercase and lower case: When we have a mix file with both uppercase and lowercase letters then first the lower case letters would be sorted following with the upper case letters

.

Example:
Create a file mix.txt

```
Command :
$ cat > mix.txt
abc
```

```
apple
BALL
Abc
bat
```

Now use the sort command

```
Command :
$ sort mix.txt
Output :
abc
Abc
apple
bat
BALL
```

Options with sort function

1. **-o Option :** Unix also provides us with special facilities like if you want to write the **output to a new file**, output.txt, redirects the output like this or you can also use the built-in sort option -o, which allows you to specify an output file. Using the -o option is functionally the same as redirecting the output to a file. Note: Neither one has an advantage over the other.

   Example: The input file is the same as mentioned above.

   Syntax :

```
$ sort inputfile.txt > filename.txt
```

```
$ sort -o filename.txt inputfile.txt

Command:

$ sort file.txt > output.txt

$ sort -o output.txt file.txt

$ cat output.txt
Output :
```

```
abhishek
chitransh
divyam
harsh
naveen
rajan
satish
```

2. **-r Option: Sorting In Reverse Order** : You can perform a reverse-order sort using the -r flag. the -r flag is an option of the sort command which sorts the input file in reverse order i.e. descending order by default.

   Example: The input file is the same as mentioned above.

   Syntax :

```
$ sort -r inputfile.txt
Command :
$ sort -r file.txt
Output :
satish
rajan
naveen
harsh
divyam
chitransh
abhishek
```

3. **-n Option** : To sort a file **numerically** used –n option. -n option is also predefined in unix as the above options are. This option is used to sort the file with numeric data present inside.

   Let us consider a file with numbers:

```
Command :
$ cat > file1.txt
50
39
15
```

```
89
200
```

Syntax :

```
$ sort -n filename.txt
Command :
$ sort -n file1.txt
Output :
15
39
50
89
200
```

4. **-nr option** : To sort a file with **numeric data in reverse order** we can use the combination of two options as stated below.

   Example :The numeric file is the same as above.

   Syntax :

```
$ sort -nr filename.txt
Command :
$ sort -nr file1.txt
Output :
200
89
50
39
15
```

5. **-k Option** : Unix provides the feature of sorting a table on the **basis of any column number by using -k option.**

   Use the -k option to sort on a certain column. For example, use “-k 2” to sort on the second column.

   Let us create a table with 2 columns

```
$ cat > employee.txt
```

```
manager  5000
clerk    4000
employee  6000
peon     4500
director 9000
guard     3000
```

Syntax :

```
$ sort -k filename.txt
```

Command :

```
$ sort -k 2n employee.txt
guard    3000
clerk    4000
peon     4500
manager  5000
employee 6000
director 9000
```

6. **-c option :** This option is used to check if the **file given is already sorted or not** & checks if a file is already sorted pass the -c option to sort. This will write to standard output if there are lines that are out of order.The sort tool can be used to understand if this file is sorted and which lines are out of order

Suppose a file exists with a list of cars called cars.txt.

```
Audi
Cadillac
BMW
Dodge
```

Syntax :

```
$ sort -c filename.txt
```

Command :

```
$ sort -c cars.txt
```

Output :

```
sort: cars.txt:3: disorder: BMW
```

Note : If there is no output then the file is considered to be already sorted

7. **-u option :** To **sort and remove duplicates** pass the -u option to sort. This will write a sorted list to standard output and remove duplicates. This option is helpful as the duplicates being removed give us a redundant file.

   Example: Suppose a file exists with a list of cars called cars.txt.

```
Audi
BMW
Cadillac
BMW
Dodge
```

   Syntax :

```
$ sort -u filename.txt
Command :
$ sort -u cars.txt
$ cat cars.txt
Output :
Audi
BMW
Cadillac
Dodge
```

8. **-M Option :** To **sort by month** pass the -M option to sort. This will write a sorted list to standard output ordered by month name.

   Suppose the following file exists and is saved as months.txt

```
$ cat > months.txt
February
January
March
August
September
```

   Syntax :

```
$ sort -M filename.txt
```

   Using The -M option with sort allows us to order this file.

```
Command :
$ sort -M months.txt
$ cat months.txt
Output :
January
February
March
August
September
```

Application and uses of sort command

1. It can sort any type of file be it table file text file numeric file and so on.
2. Sorting can be directly implemented from one file to another without the present work being hampered.
3. Sorting of table files on the basis of column has been made way simpler and easier.
4. So many option are available for sorting in all possible ways.
5. The most beneficial use is that a particular data file can be used many times as no change is made in the input file provided.
6. Original data is always safe and not hampered.

uniq Command Examples in Unix and Linux Tutorials

Uniq command in unix or linux system is used to suppress the duplicate lines from a file. It discards all the successive identical lines except one from the input and writes the output.

The syntax of uniq command is

```
uniq [option] filename
```

The options of uniq command are:

- c : Count of occurrence of each line.
- d : Prints only duplicate lines.
- D : Print all duplicate lines
- f : Avoid comparing first N fields.
- i : Ignore case when comparing.
- s : Avoid comparing first N characters.
- u : Prints only unique lines.
- w : Compare no more than N characters in lines

1. Using -c option : It tells the number of times a line was repeated.

```
$uniq -c kt.txt
3 I love music.
1
2 I love music of Kartik.
1
1 Thanks.
```

```
/*at the starting of each line its repeated number isdisplayed*/
```

2. Using -d option : It only prints the repeated lines.

```
$uniq -d kt.txt
I love music.
I love music of Kartik.
/*it only displayed one duplicate line per group*
```

3. Using -D option : It also prints only duplicate lines but not one per group.

```
uniq -D kt.txt

I love music.
I love music.
I love music.
I love music of Kartik.
I love music of Kartik.

/* all the duplicate lines are displayed*/
```

4. Using -u option : It prints only the unique lines.

```
$uniq -u kt.txt
Thanks.

/*only unique lines aredisplayed*/
```

5. Using -i option : It is used to make the comparison case-insensitive.

```
$cat f4.txt
I LOVE MUSIC
i love music
THANKS

//using uniq command//
$uniq f4.txt
I LOVE MUSIC
i love music
THANKS

/*the lines aren't treated as duplicates with simple  use of uniq*/

//now using -i option//

$uniq -i f4.txt
I LOVE MUSIC
```

```
THANKS

/*now second line is removed when -i option is used*/
```

Paste Command Examples in Unix / Linux Tutorials

Paste command is one of the useful commands in unix or linux operating system. The paste command merges the lines from multiple files. The paste command sequentially writes the corresponding lines from each file separated by a TAB delimiter on the unix terminal.

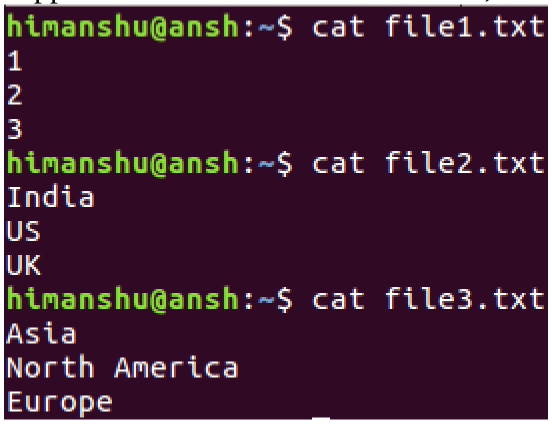
The syntax of the paste command is

```
paste [options] files-list
```

The options of paste command are:
-d : Specify of a list of delimiters.
-s : Paste one file at a time instead of in paralle

. How to join lines of multiple files using paste command?

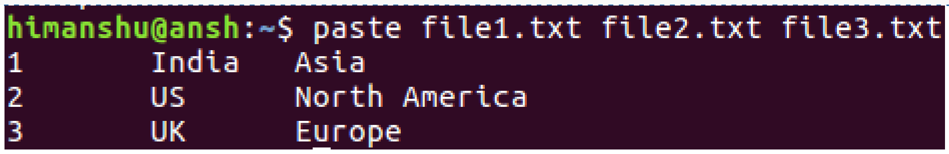
Suppose we have three files - file1.txt, file2.txt, and file3.txt - with following contents:



```
himanshu@ansh:~$ cat file1.txt
1
2
3
himanshu@ansh:~$ cat file2.txt
India
US
UK
himanshu@ansh:~$ cat file3.txt
Asia
North America
Europe
```

And the task is to merge lines of these files in a way that each row of the final output contains index, country, and continent, then you can do that using paste in the following way:

```
paste file1.txt file2.txt file3.txt
```



```
himanshu@ansh:~$ paste file1.txt file2.txt file3.txt
1       India   Asia
2       US      North America
3       UK      Europe
```

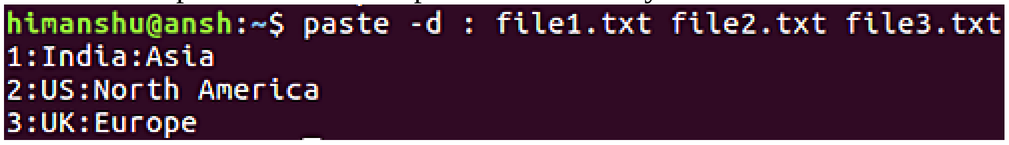
Q2. How to apply delimiters when using paste?

Sometimes, there can be a requirement to add a delimiting character between entries of each resulting row. This can be done using the **-d** command line option, which requires you to provide the delimiting character you want to use.

For example, to apply a colon (:) as a delimiting character, use the paste command in the following way:

paste -d : file1.txt file2.txt file3.txt

Here's the output this command produced on our system:



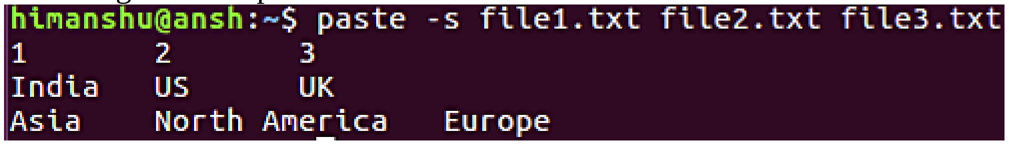
```
himanshu@ansh:~$ paste -d : file1.txt file2.txt file3.txt
1:India:Asia
2:US:North America
3:UK:Europe
```

Q3. How to change the way in which lines are merged?

By default, the paste command merges lines in a way that entries in the first column belongs to the first file, those in the second column are for the second file, and so on and so forth. However, if you want, you can change this so that the merge operation happens row-wise.
This you can do using the **-s** command line option.

paste -s file1.txt file2.txt file3.txt

Following is the output:



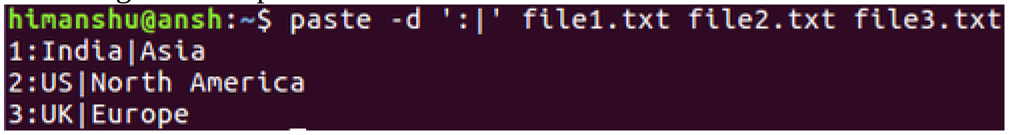
```
himanshu@ansh:~$ paste -s file1.txt file2.txt file3.txt
1       2       3
India   US      UK
Asia    North America   Europe
```

Q4. How to use multiple delimiters?

Yes, you can use multiple delimiters as well. For example, if you want to use both : and |, you can do that in the following way:

paste -d ':|' file1.txt file2.txt file3.txt

Following is the output:



```
himanshu@ansh:~$ paste -d ':|' file1.txt file2.txt file3.txt
1:India|Asia
2:US|North America
3:UK|Europe
```

<u>Paste command with a single file</u>:

1. paste command without any options is as good as the cat command when operated on a single file.

```
$ paste file1
```

```
Linux
Unix
Solaris
HPUX
AIX
```

2. Join all lines in a file:

```
$ paste -s file1
Linux   Unix    Solaris HPUX    AIX
```

-s option of paste joins all the lines in a file. Since no delimiter is specified, default delimiter tab is used to separate the columns.

3. Join all lines using the comma delimiter:

```
$ paste -d, -s file1
Linux,Unix,Solaris,HPUX,AIX
```

-d option is used to specify the delimiter. Using this -d and -s combination, all the lines in the file get merged into a single line.

Cut command in Linux

`The syntax for the cut command is as follows:`

```
cut OPTION... [FILE]...
```

Cut is a command line utility that allows you to cut parts of lines from specified files or piped data and print the result to standard output. It can be used to cut parts of a line by delimiter, byte position, and character.

When using the cut command you must use one and only one of the following options:
- -f (--fields=LIST) - Select by specifying a field, a set of fields, or a range of fields. This is the most commonly used option.
- -b (--bytes=LIST) - Select by specifying a byte, a set of bytes, or a range of bytes.
- -c (--characters=LIST) - Select by specifying a character, a set of characters, or a range of characters.

Other options are:

- -d (--delimiter) - Specify a delimiter that will be used instead of the default “TAB” delimiter.
- --complement - complement the selection. When using this option cut will display all bytes, characters or fields except the selected.

- -s (--only-delimited) - By default cut will print any line that contains no delimiter character. When using this option cut will not print lines not containing delimiters.
- --output-delimiter - The default is to use the input delimiter as the output delimiter. This option allows you to specify a different output delimiter string.

The cut command can accept zero or more input FILE names. If no FILE is specified, or when FILE is -, cut will read the standard input.

The LIST argument passed to the -f, -b, and -c options can be an integer, multiple integers separated by commas, a range of integer or multiple integer ranges separated by commas. Each range can be one of the following:

- N the Nth field, byte or character, starting from 1.
- N- from the Nth field, byte or character, to the end of the line.
- N-M from the Nth to the Mth field, byte, or character.
- -M from the first to the Mth field, byte or character.

Unix Cut Command Example

-b(byte): Use to cut the character

To extract the specific bytes,
$ cut -b 1,2,3 state.txt

In this, 1- indicate from 1st byte to end byte of a line
$ cut -b 1- state.txt

List with ranges
$ cut -b 1-3,5-7 state.txt

In this, -3 indicate from 1st byte to 3rd byte of a line
$ cut -b -3 state.txt

To cut by character use the -c option.

We will see the usage of cut command by considering the below text file as an example

```
cat> file.txt
unix or linux os
is unix good os
is linux good os
```

1. Write a unix/linux cut command to print characters by position?

The cut command can be used to print characters in a line by specifying the position of the characters. To print the characters in a line, use the -c option in cut command

```
cut –c 4 file.txt
x
u
l
```

The above cut command prints the fourth character in each line of the file. You can print more than one character at a time by specifying the character positions in a comma separated list as shown in the below example

```
cut –c 4,6 file.txt
xo
ui
ln
```

This command prints the fourth and sixth character in each line.

2. Write a unix/linux cut command to print characters by range?

You can print a range of characters in a line by specifying the start and end position of the characters.

```
cut –c 4-7 file.txt
x or
unix
linu
```

The above cut command prints the characters from fourth position to the seventh position in each line. To print the first six characters in a line, omit the start position and specify only the end position.

```
cut –c -6 file.txt
unix o
is uni
is lin
```

To print the characters from tenth position to the end, specify only the start position and omit the end position.

```
cut –c 10- file.txt
inux os
ood os
good os
```

If you omit the start and end positions, then the cut command prints the entire line.

```
cut -c- file.txt
```

3. Write a unix/linux cut command to print the fields using the delimiter?

You can use the cut command just as awk command to extract the fields in a file using a delimiter. The -d option in cut command can be used to specify the delimiter and -f option is used to specify the field position.

```
cut –d ' ' –f 2 file.txt
or
unix
linux
```

This command prints the second field in each line by treating the space as delimiter. You can print more than one field by specifying the position of the fields in a comma delimited list.

```
cut –d ' ' –f 2,3 file.txt
or linux
unix good
linux good
```

The above command prints the second and third field in each line.

Note: If the delimiter you specified is not exists in the line, then the cut command prints the entire line. To suppress these lines use the -s option in cut command.

4. Write a unix/linux cut command to display range of fields?

You can print a range of fields by specifying the start and end position.

```
cut –d ' ' –f 1-3 file.txt
```

The above command prints the first, second and third fields. To print the first three fields, you can ignore the start position and specify only the end position.

```
cut –d ' ' –f -3 file.txt
```

To print the fields from second fields to last field, you can omit the last field position.

```
cut –d ' ' –f 2- file.txt
```

5. Write a unix/linux cut command to display the first field from /etc/passwd file?

The /etc/passwd is a delimited file and the delimiter is a colon (:). The cut command to display the first field in /etc/passwd file is

```
cut –d ':' –f 1 /etc/passwd
```

Use the -b (--bytes) option to cut out a section of a line by specifying a byte position. Select the 5th byte:

```
 echo 'drüberspringen' | cut -b 5
```

Select the 5th, 9th and 13th bytes:

```
echo 'drüberspringen' | cut -b 5,9,13
bpg
```

Select the range from 1st to 5th byte:

```
echo 'drüberspringen' | cut -b 1-5
drüb
```

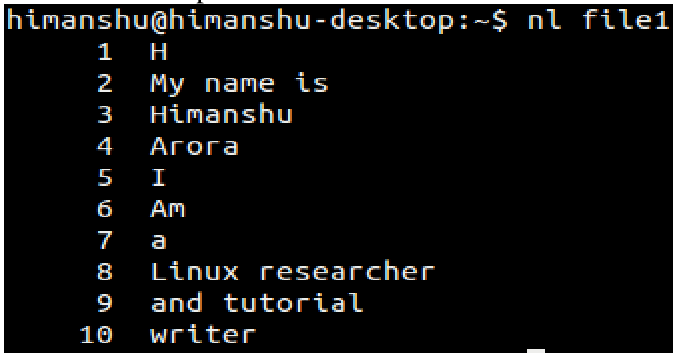
nl command in Linux

How to use nl command?

Basic usage of nl is very easy - all you have to do is to pass as argument the name of the file whose lines you want to number.

nl [filename]

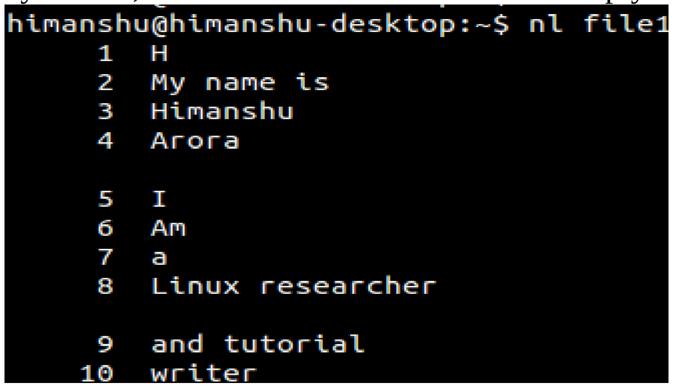
Here's an example:



```
himanshu@himanshu-desktop:~$ nl file1
     1	H
     2	My name is
     3	Himanshu
     4	Arora
     5	I
     6	Am
     7	a
     8	Linux researcher
     9	and tutorial
    10	writer
```

Q2. How to number empty lines?

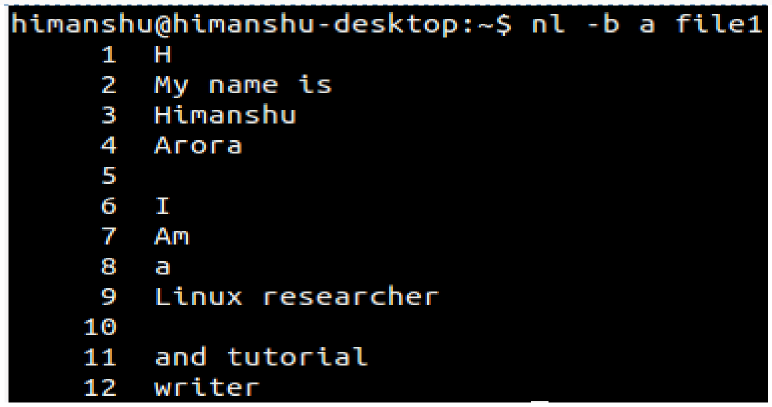
By default, the nl command doesn't number empty lines:



```
himanshu@himanshu-desktop:~$ nl file1
     1	H
     2	My name is
     3	Himanshu
     4	Arora

     5	I
     6	Am
     7	a
     8	Linux researcher

     9	and tutorial
    10	writer
```

However, if you want, you can change this behavior by passing value 'a' to the -b command line option.

```
nl -b a [filename]
```



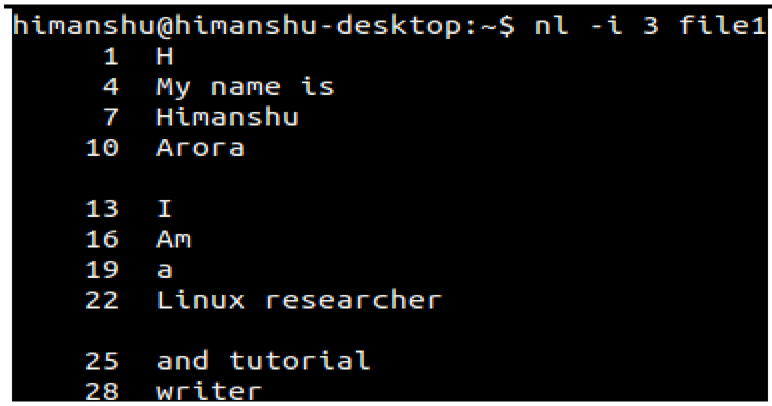
```
himanshu@himanshu-desktop:~$ nl -b a file1
     1	H
     2	My name is
     3	Himanshu
     4	Arora
     5	
     6	I
     7	Am
     8	a
     9	Linux researcher
    10	
    11	and tutorial
    12	writer
```

Q3. How to customize the number increment value?

By default, the number increment value is 1. However, you can customize this using the -i command line option.

```
nl -i [new-inc-val] [filename]
```

For example:



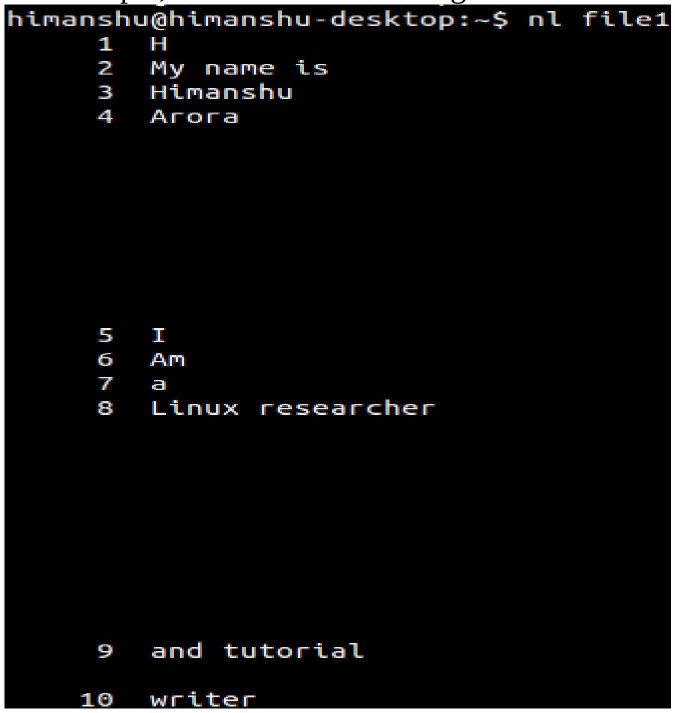
```
himanshu@himanshu-desktop:~$ nl -i 3 file1
     1  H
     4  My name is
     7  Himanshu
    10  Arora

    13  I
    16  Am
    19  a
    22  Linux researcher

    25  and tutorial
    28  writer
```

Q4. How to make nl consider multiple empty lines as one?

For this, use the -l command line option. Here's how the man page explains it:

> Consider NUMBER (default 1) consecutive empty lines to be one logical line for numbering, and only number the last one. Where fewer than NUMBER consecutive empty lines occur, do not number them. An empty line is one that contains no characters, not even spaces or tabs.

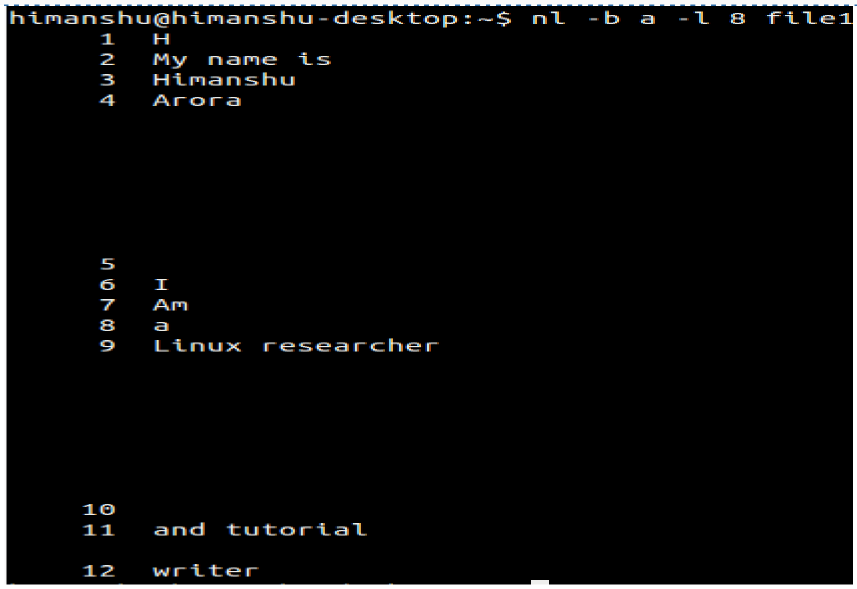
For example, let's take the following file:



```
himanshu@himanshu-desktop:~$ nl file1
     1  H
     2  My name is
     3  Himanshu
     4  Arora









     5  I
     6  Am
     7  a
     8  Linux researcher










     9  and tutorial

    10  writer
```

And suppose, we want nl to consider 8 consecutive empty lines to be one logical line for numbering. Then here's the command we'd run:

nl -b a -l 8 [filename]

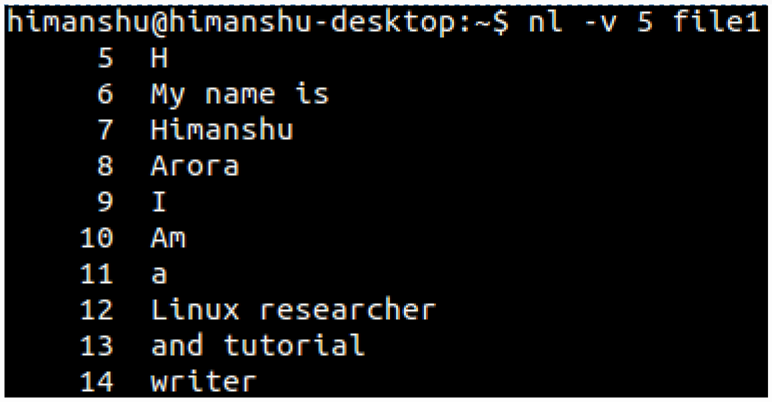


```
himanshu@himanshu-desktop:~$ nl -b a -l 8 file1
     1  H
     2  My name is
     3  Himanshu
     4  Arora







     5
     6  I
     7  Am
     8  a
     9  Linux researcher







    10
    11  and tutorial

    12  writer
```

Q5. How to make nl use a different starting line number?

By default, numbering starts with 1. However, this you can change using the -v command line option. Here's how:

nl -v [new-start-number] [filename]



```
himanshu@himanshu-desktop:~$ nl -v 5 file1
     5  H
     6  My name is
     7  Himanshu
     8  Arora
     9  I
    10  Am
    11  a
    12  Linux researcher
    13  and tutorial
    14  writer
```

STYLE is one of:

a number all lines

t number only nonempty lines

n number no lines

Examples :

1. A Basic Example

```
$ cat list.txt
apples
oranges
potatoes
lemons
garlic
$ nl list.txt
     1 apples
     2 oranges
     3 potatoes
     4 lemons
     5 garlic
```

2. Save output of nl to a file

```
$ cat list.txt
apples
oranges
potatoes
lemons
garlic
$ nl list.txt > nltext.txt
$ cat nltext.txt
     1 apples
     2 oranges
     3 potatoes
     4 lemons
     5 garlic
```

Consider the following text file named text.txt for examples :

advertisement

```
$ cat text.txt
UK
Australia
Newzealand
Brazil
America
```

3. Increment line numbers with any value using -i option

The option -i can be used to override the default increment of 1 in line numbers.

Here is an example where we have used -i to increase the line number increment to 5 :

```
$ nl -i5 text.txt
     1 UK
     6 Australia
    11 Newzealand
    16 Brazil
    21 America
```

4. Add string after line numbers using -s option

By default, the nl command adds only line numbers. But, through -s option, any string can be added that can act as a separator between line numbers and the line text.

```
$ nl -s. text.txt
     1.UK
     2.Australia
     3.Newzealand
     4.Brazil
     5.America
```

tee command in Linux with examples

tee command reads the standard input and writes it to both the standard output and one or more files. The command is named after the T-splitter used in plumbing. It basically breaks the output of a program so that it can be both displayed and saved in a file. It does both the tasks simultaneously, copies the result into the specified files or variables and also display the result.

SYNTAX:

```
tee [OPTION]... [FILE]...
```

Options :

1.-a Option : It basically do not overwrite the file but append to the given file. Suppose we have **file1.txt**

```
Input: geek for geeks
```

and **file2.txt**

```
Input:geeks for geeks
```

SYNTAX :

```
geek@HP:~$ wc -l file1.txt|tee -a file2.txt
```

OUTPUT :

```
3 file1.txt

geek@HP:~$cat file2.txt
```

```
OUTPUT:
geeks for geeks
3 file1.txt
```

Q1. How to make sure tee appends information in files?

By default, the tee command overwrites information in a file when used again. However, if you want, you can change this behavior by using the -a command line option.

```
[command] | tee -a [file]
```

So basically, the -a option forces tee to append information to the file.

Q3. How to make tee write to multiple files?

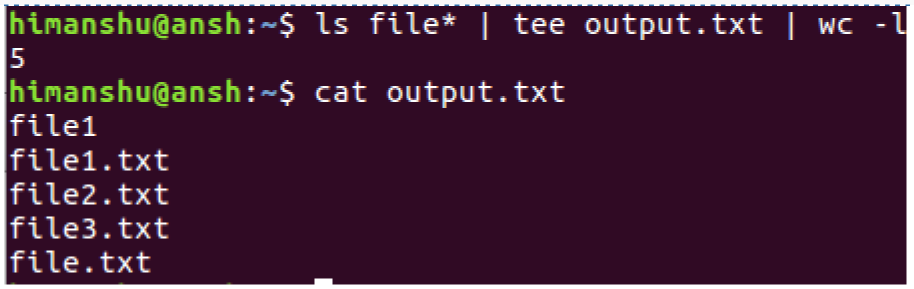
That's pretty easy. You just have to mention their names.

```
[command] | tee [file1] [file2] [file3]
```

Q3. How to make tee redirect output of one command to another?

You can not only use tee to simultaneously write output to files, but also to pass on the output as input to other commands. For example, the following command will not only store the filenames in 'output.txt' but also let you know - through wc - the number of entries in the output.txt file.

```
ls file* | tee output.txt | wc -l
```



```
himanshu@ansh:~$ ls file* | tee output.txt | wc -l
5
himanshu@ansh:~$ cat output.txt
file1
file1.txt
file2.txt
file3.txt
file.txt
```

<u>tr command in Unix/Linux with examples</u>

The tr command in UNIX is a command line utility for translating or deleting characters. It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace. It can be used with UNIX pipes to support more complex translation. **tr stands for translate.**

Syntax :

```
$ tr [OPTION] SET1 [SET2]
```

Options

-c : complements the set of characters in string i.e., operations apply to characters not in the given set

-d : delete characters in the first set from the output.

-s : replaces repeated characters listed in the set1 with single occurrence

-t : truncates set1

Sample Commands

1. How to convert lower case to upper case

To convert from lower case to upper case the predefined sets in tr can be used.

```
$cat greekfile
```

Output:

```
WELCOME TO
GeeksforGeeks
$cat greekfile | tr “[a-z]” “[A-Z]”
```

Output:

```
WELCOME TO
GEEKSFORGEEKS
```

or

```
$cat geekfile | tr “[:lower:]” “[:upper:]”
```

Output:

```
WELCOME TO
GEEKSFORGEEKS
```

2. How to translate white-space to tabs

The following command will translate all the white-space to tabs

```
$ echo "Welcome To GeeksforGeeks" | tr [:space:] '\t'
```

Output:

```
Welcome    To    GeeksforGeeks
```

3. How to translate braces into parenthesis

You can also translate from and to a file. In this example we will translate braces in a file with parenthesis.

```
$cat greekfile
```

Output:

```
 {WELCOME TO}
GeeksforGeeks
$ tr '{}' '()'   newfile.txt
```

Output:

```
(WELCOME TO)
GeeksforGeeks
```

The above command will read each character from “geekfile.txt”, translate if it is a brace, and write the output in “newfile.txt”.

4. How to use squeeze repetition of characters using -s

To squeeze repeat occurrences of characters specified in a set use the -s option. This removes repeated instances of a character.

OR we can say that,you can convert multiple continuous spaces with a single space

```
$ echo "Welcome    To    GeeksforGeeks" | tr -s [:space:] ' '
```

Output:

```
Welcome To GeeksforGeeks
```

5. How to delete specified characters using -d option

To delete specific characters use the -d option. This option deletes characters in the first set specified.

```
$ echo "Welcome To GeeksforGeeks" | tr -d 'w'
```

Output:

```
elcome To GeeksforGeeks
```

6. To remove all the digits from the string, use

```
$ echo "my ID is 73535" | tr -d [:digit:]
```

Output:

```
my ID is
```

7. How to complement the sets using -c option

You can complement the SET1 using -c option. For example, to remove all characters except digits, you can use the following.

```
$ echo "my ID is 73535" | tr -cd [:digit:]
```
Output:

```
73535
```

8. To save the results written to stdout in a file for later processing, use the shell's output redirection feature (>) as shown.

```
$ cat linux.txt | tr [a-z] [A-Z] >output.txt
$ cat output.txt

LINUX IS MY LIFE
LINUX HAS CHANGED MY LIFE
LINUX IS BEST AND EVERTHING TO ME..:)
```

9. in regards to the redirection, you can send input to TR using the input redirection and redirect the output to a file using the same command, as shown.

```
$ tr [a-z] [A-Z] < linux.txt >output.txt
```

Head command in Linux with examples

It is the complementary of Tail command. The head command, as the name implies, print the top N number of data of the given input. By default, it prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.

Syntax:

```
head [OPTION]... [FILE]...
```

Let us consider two files having name **state.txt** and **capital.txt** contains all the names of the Indian states and capitals respectively.

```
$ cat state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Jharkhand
Karnataka
```

```
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
Nagaland
Odisha
Punjab
Rajasthan
Sikkim
Tamil Nadu
Telangana
Tripura
Uttar Pradesh
Uttarakhand
West Bengal
```

Without any option, it displays only the first 10 lines of the file specified. Example:

```
$ head state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
```

Options

Short Options	Long Options
-n	--lines
-c	--bytes
-q	--quiet
-v	--verbose

1. -n num: Prints the first 'num' lines instead of first 10 lines. **num** is mandatory to be specified in command otherwise it displays an error.

```
$ head -n 5 state.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
```

If it is desired to obtain some number of lines other than the default ten, the *-n* option can be used followed by an integer indicating the number of lines desired. For example, the above example could be modified to display the first 15 lines from each file:

```
head -n15 aardvark armadillo
```

-n is a very tolerant option. For example, it is not necessary for the integer to directly follow it without a space in between. Thus, the following command would produce the same result:

```
head -n   15 aardvark armadillo
```

In fact, the letter *n* does not even need to be used at all. Just the hyphen and the integer (with no intervening space) are sufficient to tell head how many lines to return. Thus, the following would produce the same result as the above commands:

```
head -15 aardvark armadillo
```

2. -c num: Prints the first 'num' bytes from the file specified. Newline count as a single character, so if head prints out a newline, it will count it as a byte. **num** is mandatory to be specified in command otherwise displays an error.

```
$ head -c 6 state.txt
Andhra
```

3. -q: It is used if more than 1 file is given. Because of this command, data from each file is not precedes by its file name.

Without using -q option

```
==> state.txt  capital.txt <==
Hyderabad
Itanagar
Dispur
Patna
Raipur
Panaji
Gandhinagar
Chandigarh
Shimla
Srinagar
```

With using -q option

```
$ head -q  state.txt capital.txt
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
Hyderabad
Itanagar
Dispur
Patna
Raipur
Panaji
Gandhinagar
Chandigarh
Shimla
Srinagar
```

4. -v: By using this option, data from the specified file is always preceded by its file name.

```
$ head -v state.txt
```

```
==> state.txt <==
Andhra Pradesh
Arunachal Pradesh
Assam
Bihar
Chhattisgarh
Goa
Gujarat
Haryana
Himachal Pradesh
Jammu and Kashmir
```

Tail command in Linux

What is the tail command?

The tail command is a command-line utility for outputting the last part of files given to it via standard input. It writes results to standard output. By default tail returns the last ten lines of each file that it is given. It may also be used to follow a file in real-time and watch as new lines are written to it.

How to view the last ten lines of a file

To view the last ten lines of a file pass the name of a file to the tail command. The last ten lines of the file will be printed to standard output.

```
tail /usr/share/dict/words
zygote's
zygotes
zygotic
zymurgy
zymurgy's
Zyrtec
Zyrtec's
Zyuganov
Zyuganov's
Zzz
```

How to limit the number of lines to show

To set the numbers of lines to show with tail pass the -n option followed by the number of lines to show.

```
tail -n 1 /usr/share/dict/words
Zzz
```

How to limit the number of bytes to show

To limit the number of bytes shown with tail pass the -c option. Instead of limiting by number of lines this will limit by the number of bytes passed to the -c option. In the following example the output is limited to 16 bytes.

```
tail -c 24 /usr/share/dict/words
Zyuganov
Zyuganov's
Zzz
```

How to show multiple files

To show the last ten lines of multiple files pass more than one filename to the tail command. This will output the last ten lines of each file to standard output with a header indicating which file is being shown.

```
tail /usr/share/dict/words /usr/share/dict/french
==> /usr/share/dict/words <==
zygote's
zygotes
zygotic
zymurgy
zymurgy's
Zyrtec
Zyrtec's
Zyganov
Zyuganov's
Zzz

==> /usr/share/dict/french <==
zoos
zouave
zouaves
zozoter
zéro
zéros
```

```
zyeuter
zézaiement
zézaiements
zézayer
```

To suppress the header line pass the -q option. This can be useful to combine files.

```
tail -q /usr/share/dict/words /usr/share/dict/french
zygote's
zygotes
zygotic
zymurgy
zymurgy's
Zyrtec
Zyrtec's
Zyuganov
Zyuganov's
Zzz
zoos
zouave
zouaves
zozoter
zéro
zéros
zyeuter
zézaiement
zézaiements
zézayer
```

How to watch a file for changes

To watch a file for changes with the tail command pass the -f option. This will show the last ten lines of a file and will update when new lines are added. This is commonly used to watch log files in real-time. As new lines are written to the log the console will update will new lines.

```
tail -f /var/log/nginx/access.log
173.169.79.32 - - [03/Oct/2016:21:20:09 +0100] "GET / HTTP/1.1" 200 2213 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_6) AppleWebKit/601.7.7 (KHTML, like Gecko)"
...
```

Newer versions of tail also support watching multiple files. As the file updates a header will show which line the update is from.

```
tail -f /var/log/nginx/access.log /var/log/nginx/access.log

==> /var/log/nginx/access.log <==
173.169.79.32 - - [03/Oct/2016:21:23:09 +0100] "GET /apple-touch-icon-precomposed.png
HTTP/1.1" 404 162 "-" "Safari/11601.7.7 CFNetwork/760.6.3 Darwin/15.6.0 (x86_64)"

==> /var/log/nginx/error.log <==
2016/10/03 21:23:53 [error] 30632#30632: *1737 access forbidden by rule, client:
216.137.60.86, server: shapeshed.com, request: "GET /wp-login.php HTTP/1.1", host:
"shapeshed.com"
```

How to use tail with pipes

The tail command can be piped to from other commands. In the following example the output of the ls command is piped to tail to only show the five files or folders modified the longest time ago.

```
ls -t /etc | tail -n 5
login.defs
request-key.conf
libao.conf
mime.types
pcmcia
```

Applications of head Command

1. **Print line between M and N lines:** For this purpose we use head, tail and pipeline(|) commands. Command is: **head -M file_name | tail -(M-N)**, since the first line takes first M lines and tail command cuts (M-N)Lines starting from the end. Let say from state.txt file we have to print lines between 10 and 20.

```
$ head -n 20 state.txt | tail -10
Jharkhand
Karnataka
Kerala
Madhya Pradesh
Maharashtra
Manipur
Meghalaya
Mizoram
```

```
Nagaland
Odisha
```

2. **How to use the head with pipeline(|):** The head command can be piped with other commands. In the following example, the output of the ls command is piped to head to show only the three most recently modified files or folders.

```
Display all recently modified or recently used files.

$ ls -t
e.txt
d.txt
c.txt
b.txt
a.txt

Cut three most recently used file.
$ ls -t | head -n 3
e.txt
d.txt
c.txt
```

It can also be piped with one or more filters for additional processing. For example, the sort filter could be used to sort the three most recently used files or folders in the alphabetic order.

```
$ ls -t | head -n 3 | sort
c.txt
d.txt
e.txt
```

There are number of other filters or commands along which we use head command. Mainly, it can be used for viewing huge log files in Unix.