Pipe, Grep and Sort Command in Linux/Unix

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 like 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.

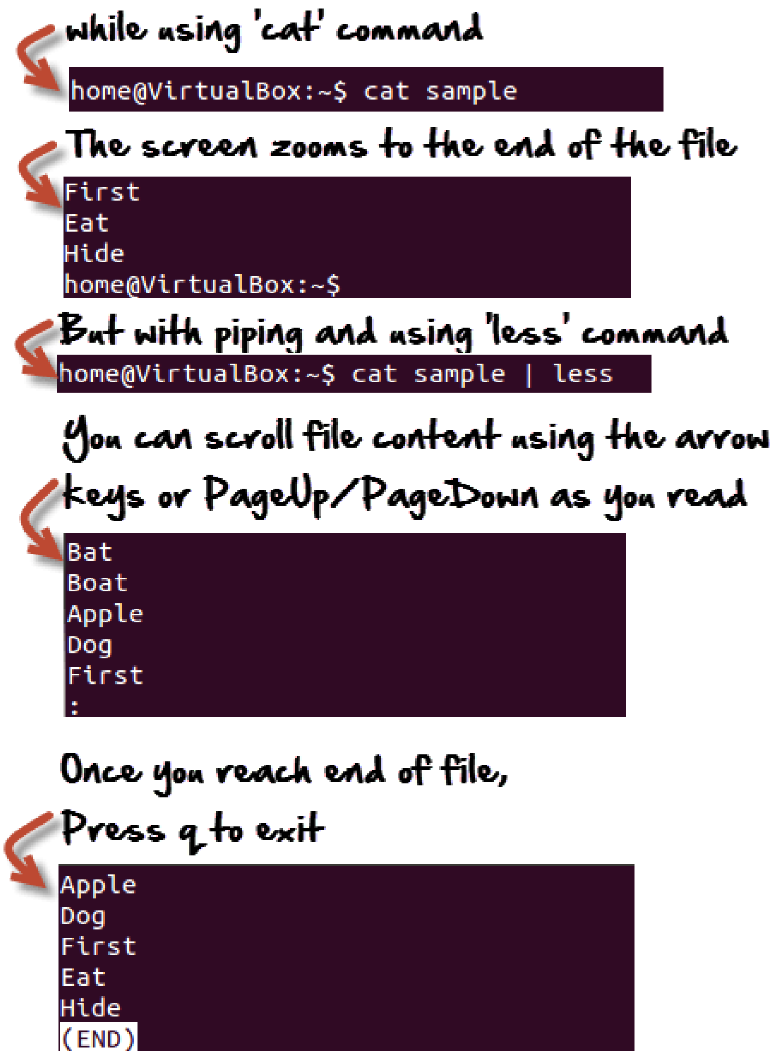
Let us understand this with an example.

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
```

An illustration would make it clear.


while using 'cat' command
The screen zooms to the end of the file
But with piping and using 'less' command
You can scroll file content using the arrow keys or PageUp/PageDown as you read
Once you reach end of file, Press q to exit

```
home@VirtualBox:~$ cat sample
```

```
First
Eat
Hide
home@VirtualBox:~$
```

```
home@VirtualBox:~$ cat sample | less
```

```
Bat
Boat
Apple
Dog
First
:
```

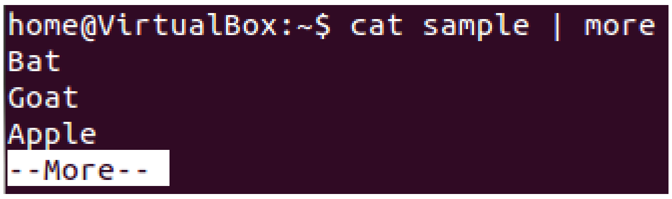
```
Apple
Dog
First
Eat
Hide
(END)
```

The 'more' command

Instead of 'less', you can also use.

```
cat Filename | more
```

And, you can view the file in digestible bits and scroll down by simply hitting the enter key.



```
home@VirtualBox:~$ cat sample | more
Bat
Goat
Apple
--More--
```

The 'grep' command

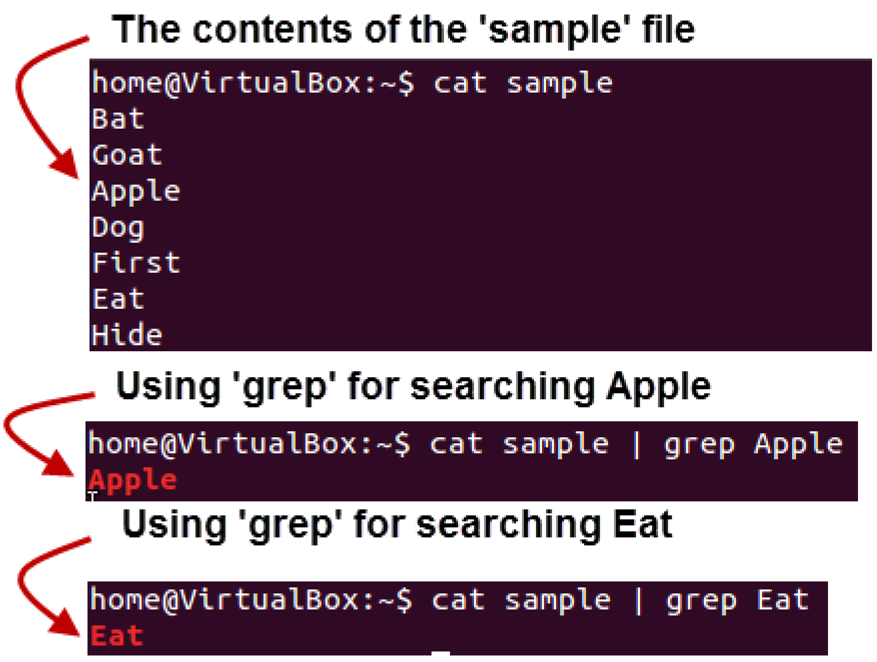
Suppose you want to search a particular information the postal code from a text file.

You may manually skim the content yourself to trace the information. A better option is to use the grep command. It will scan the document for the desired information and present the result in a format you want.

Syntax:

```
grep search_string
```

Let's see it in action -



The contents of the 'sample' file
Using 'grep' for searching Apple
Using 'grep' for searching Eat

```
home@VirtualBox:~$ cat sample
Bat
Goat
Apple
Dog
First
Eat
Hide
```

```
home@VirtualBox:~$ cat sample | grep Apple
Apple
```

```
home@VirtualBox:~$ cat sample | grep Eat
Eat
```

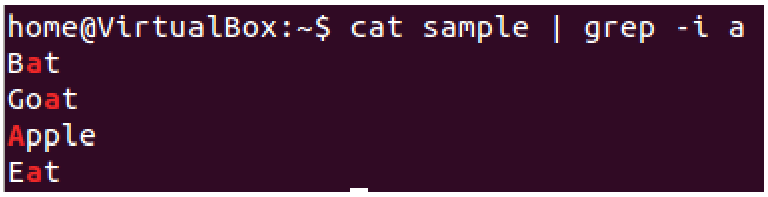
Here, **grep** command has searched the file 'sample', for the string 'Apple' and 'Eat'.

Following options can be used with this command.

Option	Function
-v	Shows all the lines that do not match the searched string
-c	Displays only the count of matching lines
-n	Shows the matching line and its number
-i	Match both (upper and lower) case

Let us try the first option **'-i'** on the same file use above -

Using the 'i' option grep has filtered the string 'a' (case-insensitive) from the all the lines.



```
home@VirtualBox:~$ cat sample | grep -i a
Bat
Goat
Apple
Eat
```

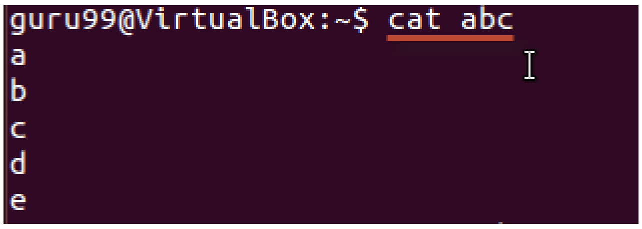
The 'sort' command

This command helps in **sorting out the contents of a file alphabetically.**

The syntax for this command is:

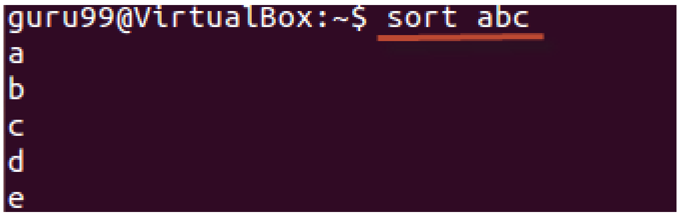
```
sort Filename
```

Consider the contents of a file.



```
guru99@VirtualBox:~$ cat abc
a
b
c
d
e
```

Using the sort command

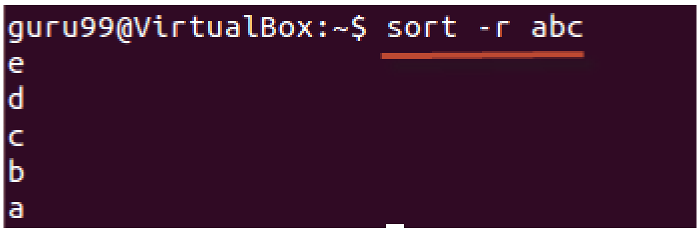


```
guru99@VirtualBox:~$ sort abc
a
b
c
d
e
```

There are **extensions** to this command as well, and they are listed below.

Option	Function
-r	Reverses sorting
-n	Sorts numerically
-f	Case insensitive sorting

The example below shows reverse sorting of the contents in file 'abc'.

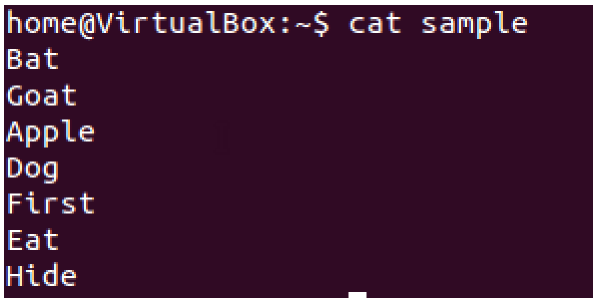


```
guru99@VirtualBox:~$ sort -r abc
e
d
c
b
a
```

What is a Filter?

A filter takes input from one command, does some processing, and gives output. When you pipe two commands, the "filtered " output of the first command is given to the next. Let's understand this with the help of an example.

We have the following file 'sample'

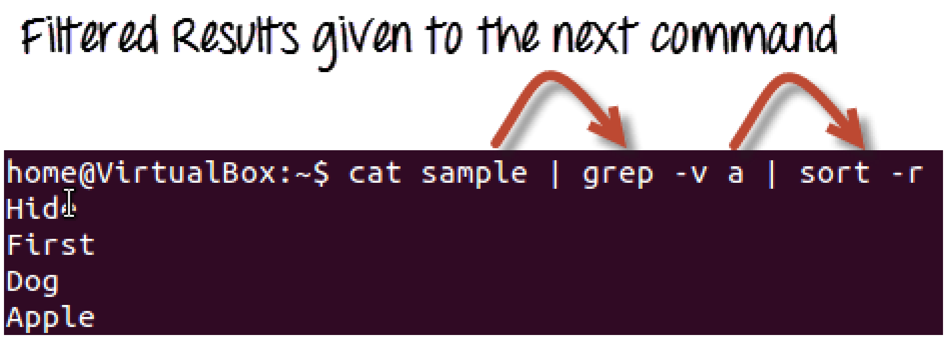


```
home@VirtualBox:~$ cat sample
Bat
Goat
Apple
Dog
First
Eat
Hide
```

We want to highlight only the lines that do not contain the character 'a', but the result should be in reverse order. For this, the following syntax can be used.

```
cat sample | grep -v a | sort - r
```

Let us look at the result.



```
home@VirtualBox:~$ cat sample | grep -v a | sort -r
Hide
First
Dog
Apple
```

The 'wc' command

The **wc** (**word count**) command in Unix/Linux operating systems is used to find out number of **newline count**, **word count**, **byte and characters** count in a files specified by the file arguments. The syntax of **wc** command as shown below.

```
# wc [options] filenames
```

The following are the options and usage provided by the command.

```
wc -l : Prints the number of lines in a file.
wc -w : prints the number of words in a file.
wc -c : Displays the count of bytes in a file.
wc -m : prints the count of characters from a file.
wc -L : prints only the length of the longest line in a file.
```

So, let's see how we can use the '**wc**' command with their few available arguments and examples in this article. We have used the '**tecmint.txt**' file for testing the commands. Let's find out the output of the file using cat command as shown below.

```
[root@tecmint ~]# cat tecmint.txt
Red Hat
CentOS
Fedora
Debian
Scientific Linux
OpenSuse
Ubuntu
Xubuntu
Linux Mint
Pearl Linux
Slackware
Mandriva
```

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
```

To count number of newlines in a file use the option ‘**-l**‘, which prints the number of lines from a given file. Say, the following command will display the count of newlines in a file. In the output the first filed assigned as count and second field is the name of file.

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

12 tecmint.txt
```

Using ‘**-w**‘ argument with ‘**wc**‘ command prints the number of words in a file. Type the following command to count the words in a file.

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

16 tecmint.txt
```

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
```

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
```

The “head” Command

The head command reads the first ten lines of a any given file name. The basic syntax of head command is:

```
head [options] [file(s)]
```

For example, the following command will display the first 10 lines of the file named ‘**/etc/passwd**‘.

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
```

If it is desired to retrieve more number of lines than the default 10, then ‘**-n**‘ option is used along with an integer telling the number of lines to be retrieved. For example, the following command will display first **5** lines from the file ‘**/var/log/yum.log**‘ file.

```
#  head -n5 /var/log/apache2/access.log

Jan 10 00:06:49 Updated: openssl-1.0.1e-16.el6_5.4.i686
Jan 10 00:06:56 Updated: openssl-devel-1.0.1e-16.el6_5.4.i686
Jan 10 00:11:42 Installed: perl-Net-SSLeay-1.35-9.el6.i686
Jan 13 22:13:31 Installed: python-configobj-4.6.0-3.el6.noarch
Jan 13 22:13:36 Installed: terminator-0.95-3.el6.rf.noarch
```

In fact, there is no need to use ‘**-n**‘ option. Just the hyphen and specify the integer without spaces to get the same result as the above command.

```
#  head  -5 /var/log/apache2/access.log

Jan 10 00:06:49 Updated: openssl-1.0.1e-16.el6_5.4.i686
Jan 10 00:06:56 Updated: openssl-devel-1.0.1e-16.el6_5.4.i686
Jan 10 00:11:42 Installed: perl-Net-SSLeay-1.35-9.el6.i686
Jan 13 22:13:31 Installed: python-configobj-4.6.0-3.el6.noarch
Jan 13 22:13:36 Installed: terminator-0.95-3.el6.rf.noarch
```

The head command can also display any desired number of bytes using ‘**-c**‘ option followed by the number of bytes to be displayed. For example, the following command will display the first **45** bytes of given file.

```
#  head -c45 /var/log/apache2/access.log

Jan 10 00:06:49 Updated: openssl-1.0.1e-16.el
```

The “tail” Command

The tail command allows you to display last 10 lines of any text file. Similar to the head command above, tail command also support options **‘n‘** number of lines and **‘n‘** number of characters. The basic syntax of tail command is:

```
#  tail [options] [filenames]
```

For example, the following command will print the last 10 lines of a file called ‘**access.log**‘.

```
#  tail /var/log/apache2/access.log

1390288226.042      0 172.16.18.71 TCP_DENIED/407 1771 GET http://download.newnext.me/spark.bin?
- NONE/- text/html
1390288226.198      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
1390288226.210   1182 172.16.20.44 TCP_MISS/200 70872 GET
http://mahavat.gov.in/Mahavat/index.jsp pg DIRECT/61.16.223.197 text/html
1390288226.284     70 172.16.20.44 TCP_MISS/304 269 GET http://mahavat.gov.in/Mahavat/i/i-19.gif
pg DIRECT/61.16.223.197 -
1390288226.362    570 172.16.176.139 TCP_MISS/200 694 GET http://p4-gayr4vyqxh7oa-
3ekrqzjikvrczq44-if-v6exp3-v4.metric.gstatic.com/v6exp3/redir.html pg
1390288226.402      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
1390288226.437    145 172.16.18.53 TCP_DENIED/407 1723 OPTIONS http://172.16.25.252/ - NONE/-
text/html
1390288226.445      0 172.16.18.53 TCP_DENIED/407 1723 OPTIONS http://172.16.25.252/ - NONE/-
text/html
1390288226.605      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
1390288226.808      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
```

Similarly, you can also print the last few lines using the ‘**-n**‘ option as shown below.

```
#  tail -5 /var/log/apache2/access.log

1390288226.402      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
1390288226.437    145 172.16.18.53 TCP_DENIED/407 1723 OPTIONS http://172.16.25.252/ - NONE/-
text/html
1390288226.445      0 172.16.18.53 TCP_DENIED/407 1723 OPTIONS http://172.16.25.252/ - NONE/-
text/html
1390288226.605      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
1390288226.808      0 172.16.16.55 TCP_DENIED/407 1753 CONNECT ent-shasta-rrs.symantec.com:443 -
NONE/- text/html
```

You can also print the number of characters using ‘**-c’** argument as shown below.

```
#  tail -c5 /var/log/apache2/access.log

ymantec.com:443 - NONE/- text/html
```

tail: 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/apache2/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)"
...
```