Linux Commands

short line

# Introduction

* 1. Fmt
  2. more
  3. Less
  4. Head
  5. Tail
  6. Sed
  7. Find
  8. Grep, Egrep, Fgrep, Rgrep
  9. Pr
  10. Tr
  11. Sort
  12. Uniq
  13. AWK

# 

# 

## **Introduction to Linux Filter Commands**

As per the user requirement the linux filter commands will read the standard inputs, performs the necessary actions or operations on top of it and writes the end result to the standard output format. A filter is a small and specialized program in the Linux operating system to get meaningful input from the user / client and with the help of a few other filters and pipes to perform a series of operations to get the highly unique or specified end result. It will help to process the information in a very dominant way such as shell jobs, modify the live data, reporting, etc.

Syntax of filter command:

[filter method] [filter option] [data | path | location of data]

### 

### **Filter Methods in Linux Operating System**

Given below are different filter methods:

* Fmt
* more
* Less
* Head
* Tail
* Sed
* Find
* Grep, Egrep, Fgrep, Rgrep
* Pr
* Tr
* Sort
* Uniq
* AWK

#### 

#### **1. Fmt**

The fmt is a simple and optimal text formatter. It is helpful to reformat the input data and print in end result in the standard output format.

Code:

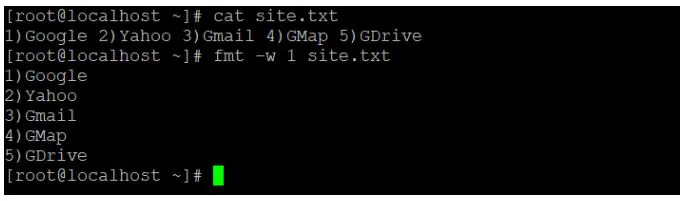
**cat site.txt**

**fmt -w 1 site.txt**

Explanation :

* In site.txt file, we are having some names. But it is in one single row and separated by space.
* The fmt command is useful to display in a single line multiple words into individual records separated by space.

Output:



#### 

#### 

#### 

#### **2. More**

The more commands are useful for file analysis. It will read the big size file. It will display the large file data in page format. The page down and page up key will not work. To display the new record, we need to press the **“enter”** key.

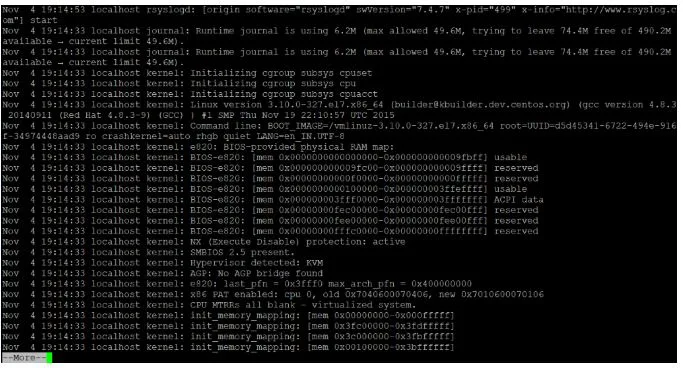
Code:

**cat /var/log/messages | more**

Explanation:

* We are reading the large file “/var/log/messages” of Linux via more commands.

Output:



#### **3. Less**

The less command is like more command but it is faster with large files. It will display the large file data in page format. The page down and page up key will work. To display the new record, we need to press “enter” key.

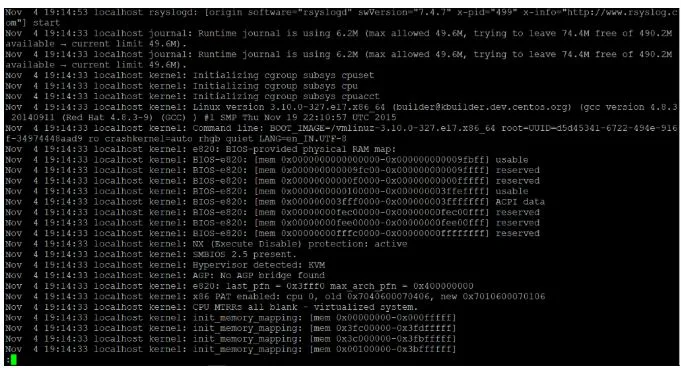
Code:

**cat /var/log/messages | less**

Explanation:

* We are reading the large file “/var/log/messages” of Linux via less command.

Output:



#### **4. Head**

As the name suggested, we are able to filter / read the initial or top lines or row of data. By default, it will read the first 10 lines or records of the given data.

If we need to read more lines, then we need to specify the number of lines that we need to read with the help of **“-n”** keyword.

Popular Course in this category



Note: The record calculation will start from the top of the file or data.

Code:

**head -n 7 file.txt**

Explanation :

* We have the sample file data having 10 records in it.
* Now we are using the default “head” command to read the data file.
* If we need to read or get the data except for the default value then we will use the “-n” keyword to read the number of lines of records.

**cat file.txt**

Output:



**head file.txt**

Output:



**head -n 7 file.txt**

Output:



#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **5. Tail**

If we need to get the data from the bottom of the file then we will use the tail command. By default, it will read the last 10 lines or records of the given data. If we need to read more lines, then we need to specify the number of lines that we need to read with the help of the **“-n”** keyword.

Code:

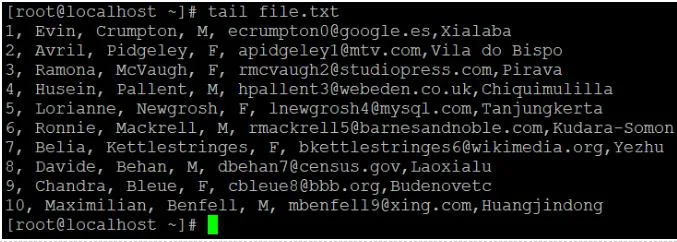
**tail -n 3 file.txt**

Explanation:

* We are using the default “tail” command to read the data file.
* If we need to read or get the data except for default value then we will use the “-n” keyword to read the number of lines of records.

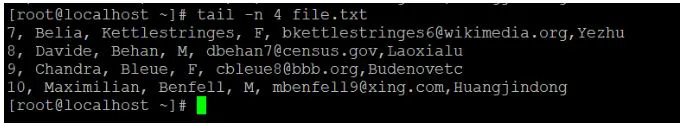
**tail file.txt**

Output:



**tail -n 4 file.txt**

Output:



#### 

#### 

#### **6. Sed**

For filtering and transforming text data, sed is a very powerful stream editor utility. It is most useful in shell or development jobs to filter out the complex data.

Code:

**sed -n '5,10p' file.txt**

Explanation:

* In head and tail command, we are able to filter the number of records from top or bottom.
* But if there is need to display the record form starting point to ending point.
* Then we need to use the sed command. We have 10 records in file.txt. But we need to record from line/record from 5 to 7 only.

**cat file.txt**

Output :



#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **7. Find**

The find filter command is useful to find the files from the Linux operating system.

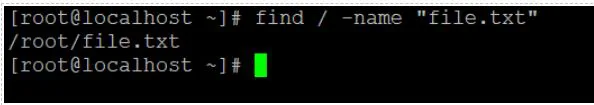
Code:

find / -name "file.txt"

Explanation:

* With the help of find command, we will filter out necessary files or directory in the Linux operating system.
* We need to add two main parameters in the find command i.e.
* Search path: “/” (I have provided the root directory path.)
* filename: “file.txt” (filename or directory name to filter or search)

Output:



#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **8. Grep, Egrep, Fgrep, Rgrep**

The grep, egrep, fgrep, rgrep are similar commands. It will be useful to filter or extract the matching pattern string form the input data or file.

Code:

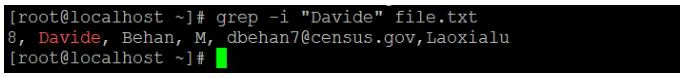
**grep -i "Davide" file.txt**

Explanation:

* In grep command, we need to specify the matching string in the command.
* The utility will filter out the same matching string (Davide) form the input data or file (file.txt) and display the matching record in “red” colour format.

Note: We can use the “-i” keyword in the grep command. It will help to ignore the upper case or lower case letter while filtering the data.

Output:



#### **9. Pr**

The pr command is useful to convert the input data into the printable format with proper column structure.

Code:

**yum list installed | pr --columns 2 -l 40 filename**

Explanation:

* In the above pr command, we are printing the output data in two columns (using ‘–columns’ keyword) and 40 lines in individual page (using ‘-l’ keyword).

Output:



#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### 

#### **10. Tr**

The tr command will translate or delete characters from the input string or data. We can transform the input data into upper or lower case.

Code:

echo "www.xyz.com" | tr [:lower:] [:upper:]

Explanation:

* In the above tr command, we need to define the input text or data in set 1 “[:lower:]” and output data in set 2 “[:upper:]”.

Output:



#### 

#### 

#### **11. Sort**

As the name suggested, we can sort or filter the records in ascending order.

Code:

sort char.txt

Explanation :

* We are having the text file (char.txt).
* It contains the number of character in a scattered way.
* Now we are using the sort command on the same file.
* After using the sorting filter on the char.txt file.
* The character will sort and display in acceding pattern.

Output:

**cat char.txt**



sort char.txt



#### 

#### **12. Uniq**

The uniq command is useful to omit repeated records or lines from the standard input.

If you want to display the number of occurrences of a line or record in the input file or data. We can use the “-c” keyword in the uniq command.

Code:

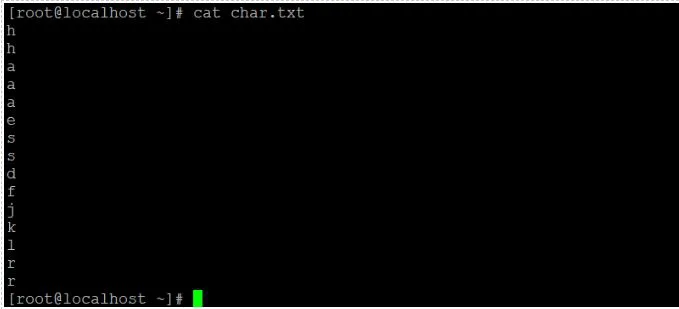
**uniq -c char.txt**

Explanation:

* In char.txt file, we have the number of duplicate characters in it.
* After using the uniq command, it will remove the duplicate character and count the number of duplicate character.

Output:

**cat char.txt**



**uniq –c char.txt**



#### 

#### **13. AWK**

In awk, we have the functionality to read the file. But here, we are using integer variables to read the file.

Below is the integer information and meaning if they are used in **awk** commands.

* $0: read the complete file or input text.
* $1: read the first field.
* $2: read the second field.
* $n: read the nth field.

Note: If we are not using any separator keyword “-F” in the awk command. Then the awk command considers the space in the text file or input data and separates the result as per the $variable provided in the same awk command.

Code:

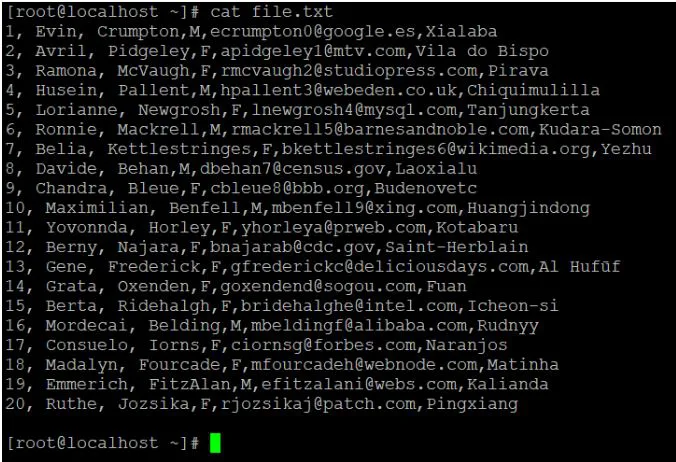
**cat file.txt | awk '{print $1}'**

Explanation:

* In file.txt, we are having the sample data.
* If we use **$0** then the entire data will be read.
* If we will use **$1** then only first column data will filter out.

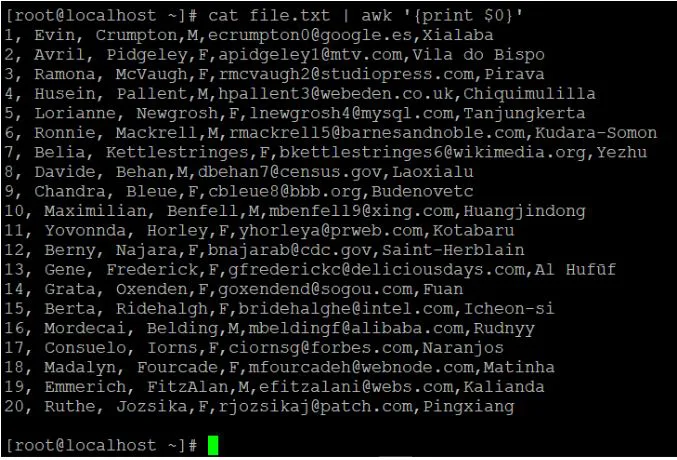
**cat file.txt**

Output: Sample file view.



**cat file.txt | awk ‘{print $0}’**

Output: Output with **“$0”**.



**cat file.txt | awk ‘{print $1}’**

Output: Output with **“$1”**

