Filters in Linux

cat : Displays the text of the file line by line.

Syntax:

cat -option filename

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

root@krosum:~# cat test.txt

Hi this is sample

text file in linux

os

root@krosum:~# cat >newfile.log # creating new FILE

test log

sample log

root@krosum:~# cat newfile.log

test log

sample log

root@krosum:~# cat -n test.txt

1 Hi this is sample

2 text file in linux

3 os

head : Displays the first n lines of the specified text files.

If the number of lines is not specified then by default prints first 10 lines.

Syntax:

```
head [-number_of_lines_to_print] [path]
```

root@krosumlabs:~/SH# head s1.log *# first 10 lines*

```
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
```

root@krosumlabs:~/SH# **head -n 3 s1.log** *# first 3 lines*
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000

root@krosumlabs:~/SH# **head -n 13 s1.log** *# first 13 lines*
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000

tail : It works the same way as head, just in reverse order.

The only difference in tail is, it returns the lines from bottom to up.

Syntax:

```
tail [-number_of_lines_to_print] [path]
```

root@krosumlabs:~/SH# **tail s1.log** *# last 10 lines*
afsda
fsdaf
sad
fasd
fs
adfa
sf
as
fasd
fsad
root@krosumlabs:~/SH# tail -n 3 s1.log ***# last 3 lines***
as
fasd
fsad

root@krosumlabs:~/SH# tail -n 3 /var/log/boot.log
```
 * Starting network connection manager                    [ OK ]
Skipping profile in /etc/apparmor.d/disable: usr.bin.firefox
 * Starting AppArmor profiles                             [ OK ]
```

root@krosumlabs:~/SH# ps -e|head
```
  PID TTY          TIME CMD
    1 ?        00:00:01 init
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 ksoftirqd/0
    6 ?        00:00:00 migration/0
    7 ?        00:00:00 cpuset
    8 ?        00:00:00 khelper
    9 ?        00:00:00 netns
   10 ?        00:00:00 sync_supers
   11 ?        00:00:00 bdi-default
```

root@krosumlabs:~/SH# ps -e|tail
```
 2522 ?        00:00:01 applet.py
 2547 ?        00:00:00 deja-dup-monito
 3282 ?        00:00:00 dhclient
 3458 pts/1    00:00:00 bash
 4271 pts/2    00:00:00 bash
 7178 ?        00:00:00 kworker/0:2
 7251 pts/0    00:00:00 ps
 7252 pts/0    00:00:00 tail
```

cut: The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output.

Syntax:

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

Sample input file

root@krosumlabs:~/SH# cat -n emp.csv

```
1   101,ram,sales,pune,10000
2   202,kumar,prod,bglore,5098
3   905,arun,sales,chennai,5050
4   307,arun,HR,chennai,5000
5   104,vijay,prod,mumbai,20060
6   505,anu,hr,hyd,2000
7   806,xerox,sales,pune,6000
```

root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort

```
10000
2000
20060
5000
5050
5098
6000
```

root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort -n
2000
5000
5050
5098
6000
10000
20060
root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort -nr
20060
10000
6000
5098
5050
5000
2000

sort : Sorts the lines alphabetically by default but there are many options available to modify the sorting mechanism.

Syntax:

```
sort [-options] [path]
```

root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort

```
1000
2000
2060
5000
5050
5098
6000
```

root@krosumlabs:~/SH# cat -n emp.csv

```
    1	101,ram,sales,pune,10000
    2	202,kumar,prod,bglore,5098
    3	905,arun,sales,chennai,5050
    4	307,arun,HR,chennai,5000
    5	104,vijay,prod,mumbai,20060
    6	505,anu,hr,hyd,2000
    7	806,xerox,sales,pune,6000
```

root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort
10000
2000
20060
5000
5050
5098
6000
root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort -n
2000
5000
5050
5098
6000
10000
20060
root@krosumlabs:~/SH# cut -d, -f 5 emp.csv |sort -nr
20060
10000
6000
5098
5050
5000
2000

uniq : Removes duplicate lines.

uniq has a limitation that it can only remove continuous duplicate lines(although this can be fixed by the use of piping).

Syntax:

```
uniq [options] [path]
```

root@krosumlabs:~/SH# whatis uniq
uniq (1) - report or omit repeated lines

root@krosumlabs:~/SH# cat >ip # creating new file
unix shell
unixshell
unix
unix
unix
java
html
java
HTML
root@krosumlabs:~/SH# uniq ip
unix shell
unixshell
unix
java
html
java
HTML

root@krosumlabs:~/SH# sort ip
html
HTML
java
java
unix
unix
unix
unixshell
unix shell

root@krosumlabs:~/SH# sort ip|uniq
html
HTML
java
unix
unixshell
unix shell

root@krosumlabs:~/SH# sort ip|uniq -i
html
java
unix
unixshell
unix shell

root@krosumlabs:~/SH# cut -d, -f 3 emp.csv
sales
prod
sales
HR
prod
hr
sales

root@krosumlabs:~/SH# cut -d, -f 3 emp.csv |sort
hr
HR
prod
prod
sales
sales
sales

root@krosumlabs:~/SH# cut -d, -f 3 emp.csv |sort|uniq
hr
HR
prod
sales

root@krosumlabs:~/SH# cut -d, -f 3 emp.csv |sort|uniq -i
hr
prod
sales

wc : wc command gives the number of lines, words and characters in the data.

Syntax:

```
wc [-options] [path]
```

The wc gives 4 outputs as:

- number of lines
- number of words
- number of characters
- path

root@krosum:~# wc /etc/passwd
36 59 1754 /etc/passwd

root@krosum:~# wc -l /etc/passwd *# no.of lines*
36 /etc/passwd

root@krosum:~# wc -w /etc/passwd *# no.of words*
59 /etc/passwd

root@krosum:~# wc -c /etc/passwd *# no.of chars*
1754 /etc/passwd

root@krosum:~# ps -e|wc -l
144
root@krosumlabs:~/SH# cut -d, -f 3 emp.csv |sort|uniq -i|wc -l
3
root@krosumlabs:~/SH# c=`cut -d, -f 3 emp.csv |sort|uniq -i|wc -l`
root@krosumlabs:~/SH# echo $c
3

grep : grep is used to search a particular information from a text file.

Syntax:

```
grep [options] pattern [path]
```

root@krosumlabs:~/SH# grep sales emp.csv
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000

root@krosumlabs:~/SH# grep -n sales emp.csv
1:101,ram,sales,pune,10000
3:905,arun,sales,chennai,5050
7:806,xerox,sales,pune,6000

root@krosumlabs:~/SH# ps -e|grep init
1 ? 00:00:01 init

root@krosumlabs:~/SH# ps -e|grep -n init
2: 1 ? 00:00:01 init

root@krosumlabs:~/SH# ps -e|grep -n bash
122: 2511 pts/0 00:00:01 bash
126: 3458 pts/1 00:00:00 bash
127: 4271 pts/2 00:00:00 bash

root@krosumlabs:~/SH# lsmod|grep -n bluetooth
21:bluetooth 148839 11 bnep,rfcomm,btusb

root@krosumlabs:~/SH# grep -n hr emp.csv
6:505,anu,hr,hyd,2000

root@krosumlabs:~/SH# grep -in hr emp.csv
4:307,arun,HR,chennai,5000
6:505,anu,hr,hyd,2000

root@krosumlabs:~/SH# grep -v sales emp.csv
202,kumar,prod,bglore,5098
307,arun,HR,chennai,5000
104,vijay,prod,mumbai,20060
505,anu,hr,hyd,2000

root@krosumlabs:~/SH# grep sales emp.csv
101,ram,sales,pune,10000
905,arun,sales,chennai,5050
806,xerox,sales,pune,6000
root@krosumlabs:~/SH#

root@krosumlabs:~/SH# grep -o sales emp.csv
sales
sales
sales

root@krosumlabs:~/SH# grep -c sales emp.csv
3

root@krosumlabs:~/SH# ps -e|grep -vc bash
129

root@krosumlabs:~/SH# ps -e|grep -c bash
3

root@krosumlabs:~/SH# grep -e sales -e prod -e chennai emp.csv
101,ram,sales,pune,10000
202,kumar,prod,bglore,5098
905,arun,sales,chennai,5050
307,arun,HR,chennai,5000
104,vijay,prod,mumbai,20060
806,xerox,sales,pune,6000

root@krosumlabs:~/SH# ps -e|grep -e mysql -e init -e java -e python
    1 ?   00:00:01 init
 1133 ?   00:00:12 mysqld

root@krosumlabs:~/SH# grep sales *
emp.csv:101,ram,sales,pune,10000
emp.csv:905,arun,sales,chennai,5050
emp.csv:806,xerox,sales,pune,6000

root@krosumlabs:~/SH# grep sales emp.csv p1.log p1.txt
emp.csv:101,ram,sales,pune,10000
emp.csv:905,arun,sales,chennai,5050
emp.csv:806,xerox,sales,pune,6000

root@krosumlabs:~/SH# grep -l sales emp.csv p1.log p1.txt
emp.csv
root@krosumlabs:~/SH# grep -l choice *
p10
p9

root@krosumlabs:~/SH# grep -n choice p9
16:	*)	echo "Sorry invalid choice"
root@krosumlabs:~/SH# grep -n choice p10
1:PS3="Enter your choice:"
16:	*)	echo "Sorry invalid choice"

root@krosumlabs:~/SH# ps -e|grep mysql
 1133 ?	00:00:12 mysqld

root@krosumlabs:~/SH# ps -e|grep -w mysql
root@krosumlabs:~/SH#
root@krosumlabs:~/SH# grep -e sales -e prod emp.csv
101,ram,sales,pune,10000
202,kumar,prod,bglore,5098
905,arun,sales,chennai,5050
104,vijay,prod,mumbai,20060
806,xerox,sales,pune,6000
root@krosumlabs:~/SH#

root@krosumlabs:~/SH# egrep 'sales|prod' emp.csv
101,ram,sales,pune,10000
202,kumar,prod,bglore,5098
905,arun,sales,chennai,5050
104,vijay,prod,mumbai,20060
806,xerox,sales,pune,6000

root@krosumlabs:~/SH# grep -E 'sales|prod' emp.csv
101,ram,sales,pune,10000
202,kumar,prod,bglore,5098
905,arun,sales,chennai,5050
104,vijay,prod,mumbai,20060
806,xerox,sales,pune,6000

tac : tac is just the reverse of cat and it works the same way, i.e., instead of printing from lines 1 through n, it prints lines n through 1.

It is just reverse of cat command.

Syntax:

```
tac [path]
```

root@krosum:~# cat test.txt *# original input file*

Hi this is sample

text file in linux

os

root@krosum:~# tac test.txt

os

text file in linux

Hi this is sample

comm: comm compare two sorted files line by line and write to standard output; the lines that are common and the lines that are unique.

Syntax :

comm [OPTION]... FILE1 FILE2

root@krosumlabs:~/SH# whatis comm
comm (1) - compare two sorted files line by line

root@krosumlabs:~/SH# cat -n file1.txt
```
    1   p1.java
    2   p2.java
    3   p3.java
    4   p4.java
```

root@krosumlabs:~/SH# cat -n file2.txt
```
    1   p3.java
    2   p4.java
    3   p5.java
    4   p6.java
```

root@krosumlabs:~/SH# comm file1.txt file2.txt
```
p1.java
p2.java
                    p3.java
                    p4.java
          p5.java
          p6.java
```

root@krosumlabs:~/SH# comm -23 file1.txt file2.txt
p1.java
p2.java

root@krosumlabs:~/SH# comm -13 file1.txt file2.txt
p5.java
p6.java

root@krosumlabs:~/SH# comm -12 file1.txt file2.txt
p3.java
p4.java

root@krosumlabs:~/SH# comm -2 file1.txt file2.txt
p1.java
p2.java
	p3.java
	p4.java

tr: 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.

Syntax :

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

root@krosum:~# cat test.txt

Hi this is sample

text file in linux

os

root@krosum:~# cat test.txt | tr 'a-z' 'A-Z'

HI THIS IS SAMPLE

TEXT FILE IN LINUX

OS

root@krosum:~# ps|tr 'a-z' 'A-Z'

```
  PID TTY          TIME CMD
 2969 PTS/1    00:00:00 SU
 2977 PTS/1    00:00:00 BASH
 3103 PTS/1    00:00:00 PS
 3104 PTS/1    00:00:00 TR
```

find: Find command used to search and locate list of files and directories

Syntax:-
find <searching from path> -name search file

Find all the files whose name is emp.csv in a current working directory.

root@krosumlabs Day3]# pwd
/root/ShellScript/Day3

root@krosumlabs Day3]# find -name emp.csv
./emp.csv
./L1/emp.csv
./L1/L2/emp.csv
./L1/L2/L3/emp.csv

find command search the input files recursively

root@krosumlabs Day3]# find ~ -name emp.csv *# Find all the files whose name is emp.csv in a login directory .*

/root/emp.csv
/root/Demo/emp.csv
/root/ShellScript/Day3/emp.csv
/root/ShellScript/Day3/L1/emp.csv
/root/ShellScript/Day3/L1/L2/emp.csv
/root/ShellScript/Day3/L1/L2/L3/emp.csv
/root/Temp/emp.csv

root@krosumlabs Day3]# find ~ -iname emp.csv *# Find Files Using Name and Ignoring Case*

./EMP.csv
./emp.csv
./L1/emp.csv
./L1/Emp.csv
./L1/L2/emp.csv
./L1/L2/L3/emp.csv

root@krosumlabs Day3]# find -name "*.log" *# Search a file with pattern*

./r1.log

./r2.log

./L1/temp.log

root@krosumlabs Day3]# find -type f *# Find list of regular files in a current directory.*

./ab.txt

./EMP.csv

./emp.csv

./L1/emp.csv

./L1/Emp.csv

./L1/L2/emp.csv

./L1/L2/L3/emp.csv

./r1.log

./r2.log

./L1/temp.log

./p1.sh

./temp.log

root@krosumlabs Day3]# find -type d *# Find list of directory files in a current directory.*

.

./L1

./L1/L2

./L1/L2/L3

root@krosumlabs Day3]# find /dev -type c *# Find list of character type device files in a /dev directory*

- /dev/hidraw0
- /dev/rfkill
- /dev/vcsa5
- /dev/tty1
- ...
- /dev/mem
- /dev/vga_arbiter

Find Files Based on their Permissions

root@krosumlabs Day4]# find -perm 0777 *# Find all the files whose permissions are 777*

./p1.sh

./p2.sh

./p3.sh

root@krosumlabs Day4]# find -perm -u=rwx

./p1.sh

./p2.sh

./p3.sh

Find the passwd file under all sub-directories starting from root directory.

root@krosumlabs ~]# find / -name passwd

/usr/bin/passwd

/sys/fs/selinux/class/passwd

/usr/share/bash-completion/passwd

/etc/pam.d/passwd

/etc/passwd

mindepth and maxdepth

- Using mindepth and maxdepth limiting search to a specific directory.
- maxdepth levels : Descend at most levels (a non-negative integer) levels of directories below the starting points.
- -maxdepth 0 means only apply the tests and actions to the starting-points themselves.
- mindepth levels : Do not apply any tests or actions at levels less than levels (a non-negative integer).
- -mindepth 1 means process all files except the starting-points.

Find the passwd file under / directory and one level down

(i.e root — level 1, and one sub-directory — level 2)

root@krosumlabs ~]# find / -maxdepth 2 -name passwd

/etc/passwd

Find the passwd file under / directory (search from level 3)

root@krosumlabs ~]# find / -mindepth 3 -name passwd

/usr/bin/passwd

/sys/fs/selinux/class/passwd

/usr/share/bash-completion/passwd

/etc/pam.d/passwd

Find the passwd file under / directory (search from level 4)

root@krosumlabs ~]# find / -mindepth 4 -name passwd

/sys/fs/selinux/class/passwd

/usr/share/bash-completion/passwd

Find Files and Directories Based on Date and Time

- As units you can use:
- b – for 512-byte blocks (this is the default if no suffix is used)
- c – for bytes
- w – for two-byte words
- k – for Kilobytes (units of 1024 bytes)
- M – for Megabytes (units of 1048576 bytes)
- G – for Gigabytes (units of 1073741824 bytes)
- we can search for exact file size, or just for bigger (+) or smaller (–) files.

For example all bigger than 512k files

root@krosumlabs ~]# find / -size +512k

search only reg.files

root@krosumlabs ~]# find / -type f -size +512k

To find all 50MB files.

root@krosumlabs ~]# find / -size 50M

To find all the files which are greater than 50MB and less than 100MB.

root@krosumlabs ~]# find / -size +50M -size -100M

To find all the files which are modified 30 days back.

root@krosumlabs ~]# find / -mtime 30

To find all the files which are accessed 30 days back.

root@krosumlabs ~]# find / -atime 30

To find all the files which are modified more than 50 days back and less than 100 days.

root@krosumlabs ~]# find / -mtime +50 –mtime -100

To find all the files which are changed in last 1 hour.

root@krosumlabs ~]# find / -cmin -60

To find all the files which are modified in last 1 hour.

root@krosumlabs ~]# find / -mmin -60

xargs

xargs converts input from standard input into arguments to a command

root@krosumlabs ~]# echo "one
two
three
four"

one
two
three
four

By default xargs displays whatever comes to its stdin as shown below.

root@krosumlabs ~]# echo "one

two

three

four"|xargs

one two three four

root@krosumlabs~]# find -name "*.txt"

./ab.txt

./sab.txt

./temp.txt

Delete all the .txt files

root@krosumlabs~]# find -name "*.txt"|xargs rm

- find list of emp.csv files under /root directory
- search a sales keyword from filtered files

find /root -name "*.csv" |xargs grep -n sales

Execute command (exec)

- **find -exec command {} \;**

- **find /root -name "*.csv" -exec grep -n sales {} \;**

• search all files with size more than 100MB and delete them.

- **find / -size +100M -exec /bin/rm {} \;**