
ondia


Linux Plus for
AWS and DevOps

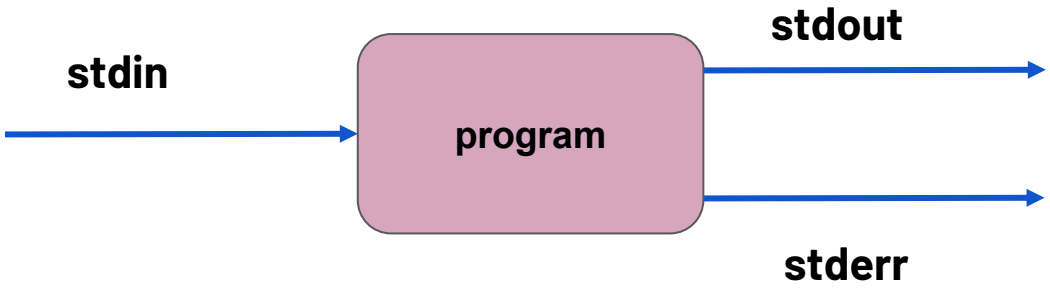
Using Filter

Table of Contents

1

stdin, stdout, stderr

stdin, stdout, stderr



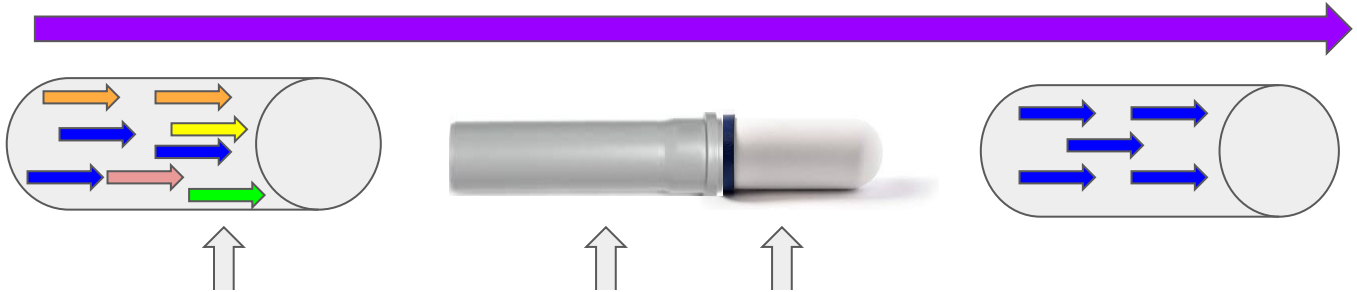
2

Filters

Filters

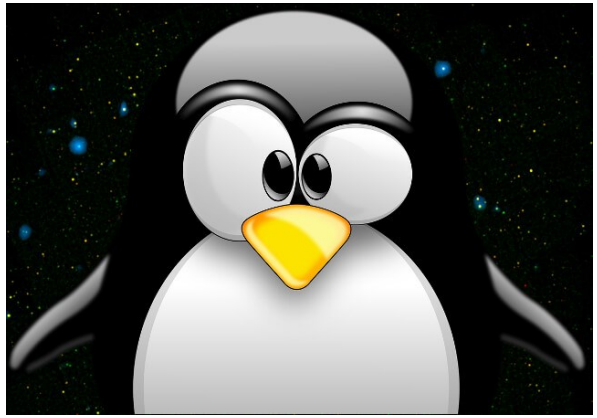
A filter is a program that takes data from one command, does some processing and gives output. Filter commands generally are used with a **pipe.**

Pipe ('|') is a mechanism that send the output of one command as input of another command.



Commands

- cat
- tee
- grep
- cut
- tr
- wc
- sort
- uniq
- comm
- sed
- awk



Filters Commands

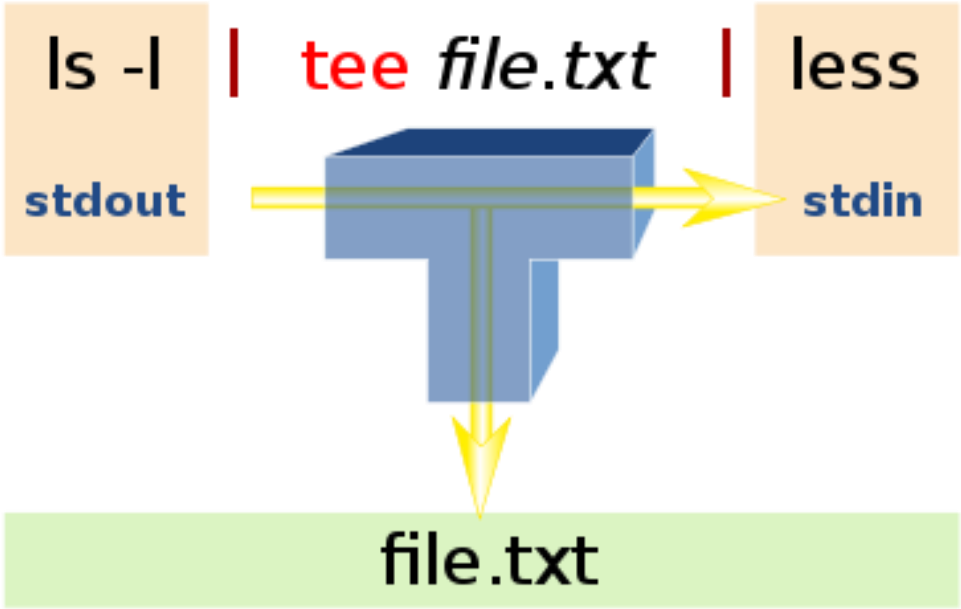
cat

When between two pipes, the cat command does nothing (except putting stdin on stdout). Displays the text of the file line by line.

```
aslan@AslanTurker:~/linuxplus$ cat days.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
aslan@AslanTurker:~/linuxplus$ cat days.txt | cat | cat | cat | cat
sunday
monday
tuesday
wednesday
thursday
friday
saturday
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

tee tee is almost the same as cat, except that it has two identical outputs.



Filters Commands

grep

The most common use of grep is to filter lines of text containing (or not containing) a certain string.

```
aslan@AslanTurker:~/linuxplus$ cat tennis.txt
Amelie Mauresmo, Fra
Justine Henin, BEL
Serena Williams, USA
Venus Williams, USA
aslan@AslanTurker:~/linuxplus$ cat tennis.txt | grep Williams
Serena Williams, USA
Venus Williams, USA
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

cut

The cut filter can select columns from files, depending on a delimiter or a count of bytes

cut -d(delimiter) -f(columnNumber) <fileName>

```
aslan@AslanTurker:~/linuxplus$ ls *.* -l
-rw-r--r-- 1 aslan aslan   0 Jan 30 12:35 Linuxplus.txt
-rw-r--r-- 1 aslan aslan  65 Jan 30 15:14 count.txt
-rw-r--r-- 1 aslan aslan  64 Jan 30 15:17 days.txt
-rw-r--r-- 1 aslan aslan   0 Jan 30 12:37 linux.txt
-rw-r--r-- 1 aslan aslan   0 Jan 30 12:31 linuxplus.txt
-rw-r--r-- 1 aslan aslan  75 Jan 30 15:17 marks.txt
-rw-r--r-- 1 aslan aslan 258 Jan 30 12:59 quotes.txt
-rw-r--r-- 1 aslan aslan  85 Jan 30 15:17 tennis.txt
-rw-r--r-- 1 aslan aslan  15 Jan 30 13:01 winter.txt
aslan@AslanTurker:~/linuxplus$ ls *.* -l | cut -d' ' -f3
aslan
aslan
aslan
aslan
aslan
aslan
aslan
aslan
aslan
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

tr

The command 'tr' stands for 'translate'.
It is used to translate, like from lowercase to uppercase and vice versa or new lines into spaces.

```
aslan@AslanTurker:~/linuxplus$ cat linuxplus.txt
Linux is only free if your time has no value
aslan@AslanTurker:~/linuxplus$ cat linuxplus.txt | tr "aer" "iou"
Linux is only fuoo if youu timo his no viluo
aslan@AslanTurker:~/linuxplus$ cat quotes.txt
1. "The only way to do great work is to love what you do."

2. "Success is not final, failure is not fatal: It is the courage to continue that counts."

3. "Believe you can and you're halfway there."

4. "The best way to predict the future is to create it."
aslan@AslanTurker:~/linuxplus$ cat quotes.txt | tr "\n" " "
1. "The only way to do great work is to love what you do."  2. "Success is not final, failure is not fatal: It is the coura
ge to continue that counts."  3. "Believe you can and you're halfway there."  4. "The best way to predict the future is to
create it." aslan@AslanTurker:~/linuxplus$
```

Filters Commands

wc

Counting words, lines and characters is easy with wc.

- wc **<fileName>** (Counts words, lines and characters)
- wc -l **<fileName>** (Counts only lines)
- wc -w **<fileName>** (Counts only words)
- wc -c **<fileName>** (Counts only characters)

```
aslan@AslanTurker:~/linuxplus$ cat count.txt
one
two
three
four
five
six
seven
eight
nine
ten
eleven

aslan@AslanTurker:~/linuxplus$ wc count.txt
12 11 69 count.txt
aslan@AslanTurker:~/linuxplus$ wc -l count.txt
12 count.txt
aslan@AslanTurker:~/linuxplus$ wc -w count.txt
11 count.txt
aslan@AslanTurker:~/linuxplus$ wc -c count.txt
69 count.txt
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

sort

The sort filter will default to an alphabetical sort.

sort -r	the flag returns the results in reverse order
sort -f	the flag does case insensitive sorting

```
aslan@AslanTurker:~/linuxplus$ cat marks.txt
victor-10
albert-9
walter-8
john-10
james-9
oliver-7
tom-7
aeron-9
aslan@AslanTurker:~/linuxplus$ sort marks.txt
aeron-9
albert-9
james-9
john-10
oliver-7
tom-7
victor-10
walter-8
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

uniq

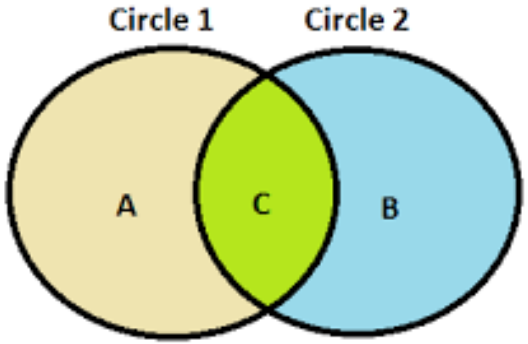
With the help of uniq command you can form a **sorted list** in which every word will occur only once.

```
aslan@AslanTurker:~/linuxplus$ cat trainees.txt
john
james
aeron
oliver
walter
albert
james
john
travis
mike
aeron
thomas
daniel
john
aeron
oliver
mike
john
aslan@AslanTurker:~/linuxplus$ sort trainees.txt | uniq
aeron
albert
daniel
james
john
mike
oliver
thomas
travis
walter
aslan@AslanTurker:~/linuxplus$
```

Filters Commands

comm

The 'comm' command compares two files or streams.
By default, 'comm' will always display three columns.
First column indicates non-matching items of first file, second column indicates non-matching items of second file, and third column indicates matching items of both the files.
Both the files has to be in sorted order for 'comm' command to be executed.




```
aslan@AslanTurker:~/linuxplus$ cat a.txt
Aeron
Bill
James
John
Oliver
Walter
aslan@AslanTurker:~/linuxplus$ cat b.txt
Guile
James
John
Raymond
aslan@AslanTurker:~/linuxplus$ comm a.txt b.txt
Aeron
Bill
        Guile
                James
                John
Oliver
        Raymond
Walter
aslan@AslanTurker:~/linuxplus$
```

Exercise

1. Create a file named countries.csv with the following content

```
Country,Capital,Continent
USA,Washington,North America
France,Paris,Europe
Canada,Ottawa,North America
Germany,Berlin,Europe
```

1. a. Cut only “Continent” column
   b. Remove header
   c. Sort the output
   d. List distinct values
   e. Save final output to “continents.txt” file
1. Display content of continents.txt file

4

Using Control Operators

Table of Contents

Control Operators

- Semicolon (;)
- Ampersand (&)
- Dollar Question Mark ($?)
- Double Ampersand (&&)
- Double Vertical Bar (||)
- Combining && and ||
- Pound Sign (#)
- Escaping Special Characters (\)
- End of line Backslash

Control Operators

We put more than one command on the command line using control operators.

Control Operator	Usage
; semicolon	More than one command can be used in a single line.
& ampersand	Command ends with & and doesn't wait for the command to finish.
$? dollar question mark	Used to store exit code of the previous command.
&& double ampersand	Used as logical AND.
|| double vertical bar	Used as logical OR.
Combining && and ||	Used to write if then else structure in the command line.
# pound sign	Anything was written after # will be ignored.

Semicolon (;)

You can put two or more commands on the same line separated by a **semicolon (;)**

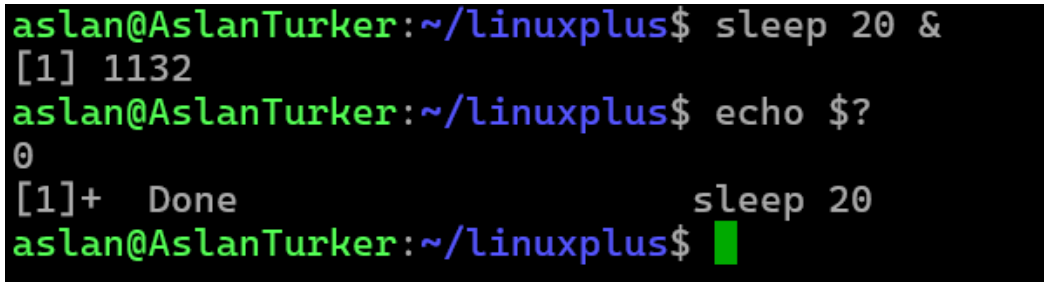
```
aslan@AslanTurker:~/linuxplus$ cat days.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
aslan@AslanTurker:~/linuxplus$ cat count.txt
one
two
three
four
five
six
seven
eight
nine
ten
eleven

aslan@AslanTurker:~/linuxplus$ cat days.txt ; cat count.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
one
two
three
four
five
six
seven
eight
nine
ten
eleven

aslan@AslanTurker:~/linuxplus$
```

Ampersand (&)

When a line ends with an ampersand &, the shell will not wait for the command to finish. You will get your shell prompt back, and the command is executed in background. You will get a message when this command has finished executing in background.



- Look at the above snapshot, command **"sleep 20 &"** has displayed a message after 15 seconds.
- Meanwhile, in the shell prompt, we can write any other command.

Dollar Question Mark ($?)

This control operator is used to check the status of last executed command. If status shows '0' then command was successfully executed and if shows '1' then command was a failure.

```
aslan@AslanTurker:~/linuxplus$ ls
a.txt  b.txt      days.txt  file2  linuxplus.txt  quotes.txt  trainees.txt
all    count.txt  file1     file3  marks.txt      tennis.txt  winter.txt
aslan@AslanTurker:~/linuxplus$
aslan@AslanTurker:~/linuxplus$ echo $?
0
aslan@AslanTurker:~/linuxplus$ rmdir *
rmdir: failed to remove 'a.txt': Not a directory
rmdir: failed to remove 'all': Not a directory
rmdir: failed to remove 'b.txt': Not a directory
rmdir: failed to remove 'count.txt': Not a directory
rmdir: failed to remove 'days.txt': Not a directory
rmdir: failed to remove 'file1': Not a directory
rmdir: failed to remove 'file2': Not a directory
rmdir: failed to remove 'file3': Not a directory
rmdir: failed to remove 'linuxplus.txt': Not a directory
rmdir: failed to remove 'marks.txt': Not a directory
rmdir: failed to remove 'quotes.txt': Not a directory
rmdir: failed to remove 'tennis.txt': Not a directory
rmdir: failed to remove 'trainees.txt': Not a directory
rmdir: failed to remove 'winter.txt': Not a directory
aslan@AslanTurker:~/linuxplus$ echo $?
1
aslan@AslanTurker:~/linuxplus$
```

Double Ampersand (&&)

The command shell interprets the && as the logical AND. When using this command, the second command will be executed only when the first one has been successfully executed.

```
aslan@AslanTurker:~/linuxplus$ cat count.txt && cat days.txt
one
two
three
four
five
six
seven
eight
nine
ten
eleven

sunday
monday
tuesday
wednesday
thursday
friday
saturday
aslan@AslanTurker:~/linuxplus$ cd .. && ls
Conditional_Statements  awk.txt  calculation2.sh  count.txt         deneme.out  linuxplus
Loops-Functions         calc.sh  case.sh          count_backup.txt  images.jpg  number++.sh
aslan@AslanTurker:~$
```

Double Vertical Bar (||)

The command shell interprets the (||) as the logical OR. This is opposite of logical AND. Means second command will execute only when first command will be a failure.

```
aslan@AslanTurker:~/linuxplus$ cat days.txt || echo "linuxplus" ; echo one
sunday
monday
tuesday
wednesday
thursday
friday
saturday
one
aslan@AslanTurker:~/linuxplus$ catz days.txt || echo "linuxplus" ; echo one
Command 'catz' not found, did you mean:
  command 'cat' from deb coreutils (9.4-2ubuntu2)
Try: sudo apt install <deb name>
linuxplus
one
aslan@AslanTurker:~/linuxplus$
```

Combining && and ||

You can use this logical AND and logical OR to write an if-then-else structure on the command line. This example uses echo to display whether the rm command was successful.

```
aslan@AslanTurker:~/linuxplus$ cat a.txt
Aeron
Bill
James
John
Oliver
Walter
aslan@AslanTurker:~/linuxplus$ rm a.txt && echo "It worked." || echo "It failed!"
It worked.
aslan@AslanTurker:~/linuxplus$ rm a.txt && echo "It worked." || echo "It failed!"
rm: cannot remove 'a.txt': No such file or directory
It failed!
aslan@AslanTurker:~/linuxplus$
```

Pound Sign (#)

Everything written after a pound sign (#) is ignored by the shell. This is useful to write a shell comment but has no influence on the command execution or shell expansion.

```
aslan@AslanTurker:~$ cd linuxplus/                    #We move to linuxplus directory
aslan@AslanTurker:~/linuxplus$ ls                     #Is it empty or not?
all    count.txt  file1  file3          marks.txt   tennis.txt    winter.txt
b.txt  days.txt   file2  linuxplus.txt  quotes.txt  trainees.txt
aslan@AslanTurker:~/linuxplus$
```

Escaping Special Characters (\)

Escaping characters are used to enable the use of control characters in the shell expansion but without interpreting it by the shell.

```
aslan@AslanTurker:~/linuxplus$ echo this is \* symbol
this is * symbol
aslan@AslanTurker:~/linuxplus$ echo this \ \ \ \is \ \ \ \Linux.
this    is    Linux.
aslan@AslanTurker:~/linuxplus$ echo Here escaping characters: \\\ \#\ \&\ \"\ \'
Here escaping characters: \ # & " '
aslan@AslanTurker:~/linuxplus$
```

End of Line Backslash (\)

Lines ending in a backslash are continued on the next line. The shell does not interpret the newline character and will wait on shell expansion and execution of the command line until a newline without backslash is encountered.

```
aslan@AslanTurker:~/linuxplus$ echo This command line \
> is split in three \
> parts
This command line is split in three parts
aslan@AslanTurker:~/linuxplus$
```

Exercise

1. a. Search for “clarusway.txt” in the current directory
   b. If it exists display its content
   c. If it does not exist print message “Too early!”
1. Create a file named “clarusway.txt” that contains “Congratulations”
2. Repeat Step 1

Homework

diff

diff (1) - compare files line by line
diff (1p) - compare two files

https://www.geeksforgeeks.org/diff-command-linux-examples/#:~:text=diff%20stands%20for%20difference.,make%20the%20two%20files%20identical.

https://www.linuxtechi.com/diff-command-examples-linux/

THANKS!

Any questions?