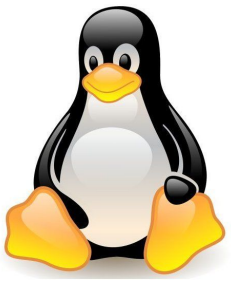
# Linux Plus
for
# AWS and DevOps



CLARUSWAY
WAY TO REINVENT YOURSELF

# Using Filter


CLARUSWAY
WAY TO REINVENT YOURSELF

# Table of Contents

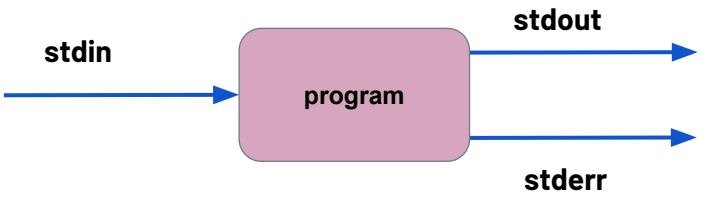




1

## stdin, stdout, stderr

# stdin, stdout, stderr



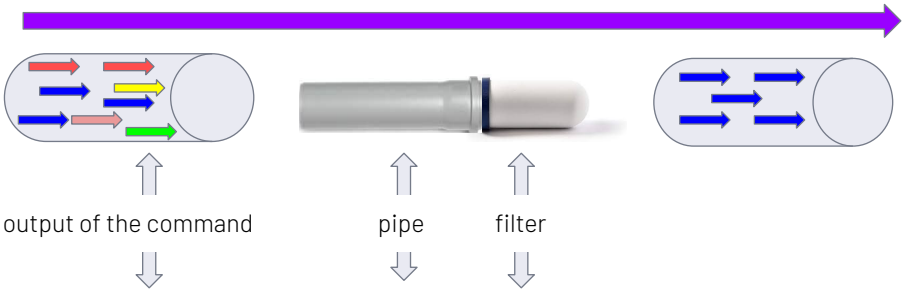



## 2 Filters

# 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.



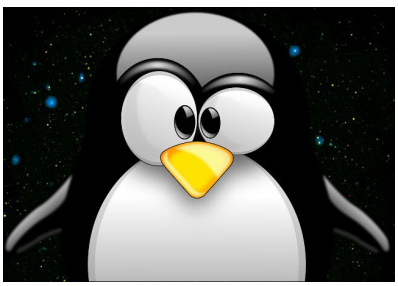

CLARUSWAY
WAY TO REINVENT YOURSELF



## Commands

3

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



CLARUSWAY
WAY TO REINVENT YOURSELF

# 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.

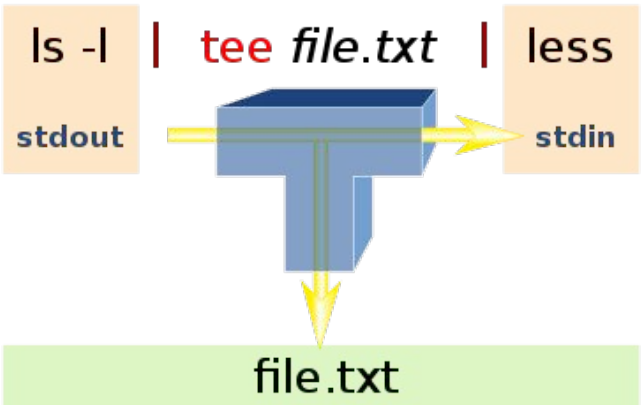
```
ubuntu@clarusway: $ cat days.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
ubuntu@clarusway: $ cat days.txt | cat | cat | cat | cat
sunday
monday
tuesday
wednesday
thursday
friday
saturday
ubuntu@clarusway: $
```



# 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.

```
ubuntu@clarusway: $ cat tennis.txt
Amelie Mauresmo, Fra
Justine Henin, BEL
Serena Williams, USA
Venus Williams, USA
ubuntu@clarusway: $ cat tennis.txt | grep Williams
Serena Williams, USA
Venus Williams, USA
ubuntu@clarusway: $
```



# 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>**

```
user@clarusway-linux:~$ ls *.* -l
-rw-r--r-- 1 user user   16 Mar  2 21:56 classes.html
-rw-r--r-- 1 user user 8980 Mar  2 21:53 examples.desktop
-rw-r--r-- 1 user user   24 Mar  2 23:22 html.txt
-rw-r--r-- 1 user user   17 Mar  2 22:42 lesson.txt
-rw-r--r-- 1 user user   13 Mar  2 23:22 linux.txt
-rw-r--r-- 1 user user    0 Mar  4 21:42 xtml.txt
user@clarusway-linux:~$ ls *.* -l | cut -d' ' -f3
user
user
user
user
user
user
user@clarusway-linux:~$
```

# 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.

```
ubuntu@clarusway: $ cat clarusway.txt
Way to Reinvent Yourself
ubuntu@clarusway: $ cat clarusway.txt | tr 'aer' 'iou'
Wiy to Roinvont Youusolf
ubuntu@clarusway: $ cat count.txt
one
two
three
four
five
ubuntu@clarusway:~$ cat count.txt | tr '\n' ' '
one two three four five ubuntu@clarusway:~$
```



# 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)

```
ubuntu@clarusway: $ cat count.txt
one
two
three
four
five
ubuntu@clarusway: $ wc count.txt
 5  5 24 count.txt
ubuntu@clarusway: $ wc -l count.txt
5 count.txt
ubuntu@clarusway: $ wc -w count.txt
5 count.txt
ubuntu@clarusway: $ wc -c count.txt
24 count.txt
ubuntu@clarusway: $
```

# 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 |

```
ubuntu@clarusway: $ cat marks.txt
John-10
James-9
Aaron-8
Oliver-7
Walter-6
ubuntu@clarusway: $ sort marks.txt
Aaron-8
James-9
John-10
Oliver-7
Walter-6
ubuntu@clarusway: $
```

CLARUSWAY
WAY TO REINVENT YOURSELF



# Filters Commands

**uniq**

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

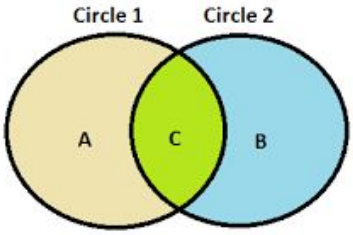
```
ubuntu@clarusway: $ cat marks.txt
John
James
Aaron
Oliver
Walter
Aaron
John
James
John
John

ubuntu@clarusway: $ sort marks.txt | uniq

Aaron
James
John
Oliver
Walter
ubuntu@clarusway: $
```

CLARUSWAY
WAY TO REINVENT YOURSELF

# 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.




```
ubuntu@clarusway: $ cat file1
Aaron
James
John
Oliver
Walter
ubuntu@clarusway: $ cat file2
Guile
James
John
Raymond
ubuntu@clarusway: $ comm file1 file2
Aaron
        Guile
                James
                John
Oliver
        Raymond
Walter
ubuntu@clarusway: $
```



# Exercise 1

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
```

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

# 4 Using Control Operators




# Table of Contents

# 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 (;)**

```
ubuntu@clarusway: $ cat days.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
ubuntu@clarusway: $ cat count.txt
one
two
three
four
five
ubuntu@clarusway: $ cat days.txt ; cat count.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
one
two
three
four
five
ubuntu@clarusway: $
```

# 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.

```
ubuntu@clarusway: $ sleep 20 &
[1] 3396
ubuntu@clarusway: $
[1]+  Done                    sleep 20
ubuntu@clarusway: $
```

- 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.

```
ubuntu@clarusway: $ ls
Repo           count.txt  days.txt  file1.txt  file2.txt      key.txt      keypc.pem  marks.txt  tennis.txt
clarusway.txt  days       file1     file2      get-docker.sh  key.txt.pub  keypc.pub  temp.txt   testdir
ubuntu@clarusway: $
ubuntu@clarusway: $ echo $?
0
ubuntu@clarusway: $ rmdir *
rmdir: failed to remove 'Repo': Directory not empty
rmdir: failed to remove 'clarusway.txt': Not a directory
rmdir: failed to remove 'count.txt': Not a directory
rmdir: failed to remove 'days': Not a directory
rmdir: failed to remove 'days.txt': Not a directory
rmdir: failed to remove 'file1': Not a directory
rmdir: failed to remove 'file1.txt': Not a directory
rmdir: failed to remove 'file2': Not a directory
rmdir: failed to remove 'file2.txt': Not a directory
rmdir: failed to remove 'get-docker.sh': Not a directory
rmdir: failed to remove 'key.txt': Not a directory
rmdir: failed to remove 'key.txt.pub': Not a directory
rmdir: failed to remove 'keypc.pem': Not a directory
rmdir: failed to remove 'keypc.pub': Not a directory
rmdir: failed to remove 'marks.txt': Not a directory
rmdir: failed to remove 'temp.txt': Not a directory
rmdir: failed to remove 'tennis.txt': Not a directory
rmdir: failed to remove 'testdir': Directory not empty
ubuntu@clarusway: $ echo $?
1
ubuntu@clarusway: $
```

# 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.

```
ubuntu@clarusway: $ cat days.txt && cat count.txt
sunday
monday
tuesday
wednesday
thursday
friday
saturday
one
two
three
four
five
ubuntu@clarusway: $ cd Repo && ls
m
ubuntu@clarusway:~/Repo$
```



# 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.

```
ubuntu@clarusway: $ cat days.txt || echo "clarusway" ; echo one
sunday
monday
tuesday
wednesday
thursday
friday
saturday
one
ubuntu@clarusway: $ zecho days.txt || echo "clarusway" ; echo one

Command 'zecho' not found, did you mean:

  command 'aecho' from deb netatalk
  command 'echo' from deb coreutils

Try: sudo apt install <deb name>

clarusway
one
ubuntu@clarusway: $
```

# 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.

```
ubuntu@clarusway: $ cat file1
Aaron
James
John
Oliver
Walter
ubuntu@clarusway: $ rm file1 && echo It worked! || echo It failed!
It worked!
ubuntu@clarusway: $ rm file1 && echo It worked! || echo It failed!
rm: cannot remove 'file1': No such file or directory
It failed!
ubuntu@clarusway: $
```



# 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.

```
ubuntu@clarusway: $ mkdir test          # We create a directory
ubuntu@clarusway: $ cd test             # We enter the directory
ubuntu@clarusway:~/test$ ls             # is it empty ?
ubuntu@clarusway:~/test$
```

# 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.

```
ubuntu@clarusway:~$ echo this is \* symbol.
this is * symbol.
ubuntu@clarusway:~$ echo this \ \ \ \is \ \ \ \clarusway.
this    is    clarusway.
ubuntu@clarusway:~$ echo escaping \\\ \#\ \&\ \"\ \'
escaping \ # & " '
ubuntu@clarusway:~$
```



# 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.

```
ubuntu@clarusway: $ echo This command line \
> > is split in three \
> > parts
ubuntu@clarusway: $ This command line is split in three parts
```




# 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!”
2. Create a file named “clarusway.txt” that contains “Congratulations”
3. Repeat Step 1

# Exercises

**ls -l | cut -d' ' -f3**
**ls -l | tee output.txt | cut -d' ' -f3**
**ls -l | cut -d' ' -f3 | tee output.txt**

**cat file.txt | tr 'a' 'W'** **cat file.txt | tr 'a' 'W' | file.txt**

**cat /etc/passwd | cut -d' ' -f1 | wc -l**

**cat tennis_players.txt | sort** **cat tennis_players.txt | sort -r**

**info ls | tee ls_exp.txt** **cat /etc/passwd | cut -d' ' -f3 | uniq | wc -l**



# 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?