Linux para big data

Alex Di Genova

25/08/2025

Visión global

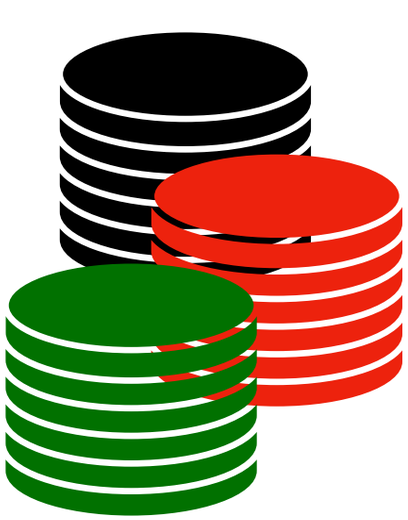
Byte	B	10^0	1
Kilobyte	KB	10^3	1,000
Megabyte	MB	10^6	1,000,000
Gigabyte	GB	10^9	1,000,000,000
Terabyte	TB	10^{12}	1,000,000,000,000
Petabyte	PB	10^{15}	1,000,000,000,000,000
Exabyte	EB	10^{18}	1,000,000,000,000,000,000

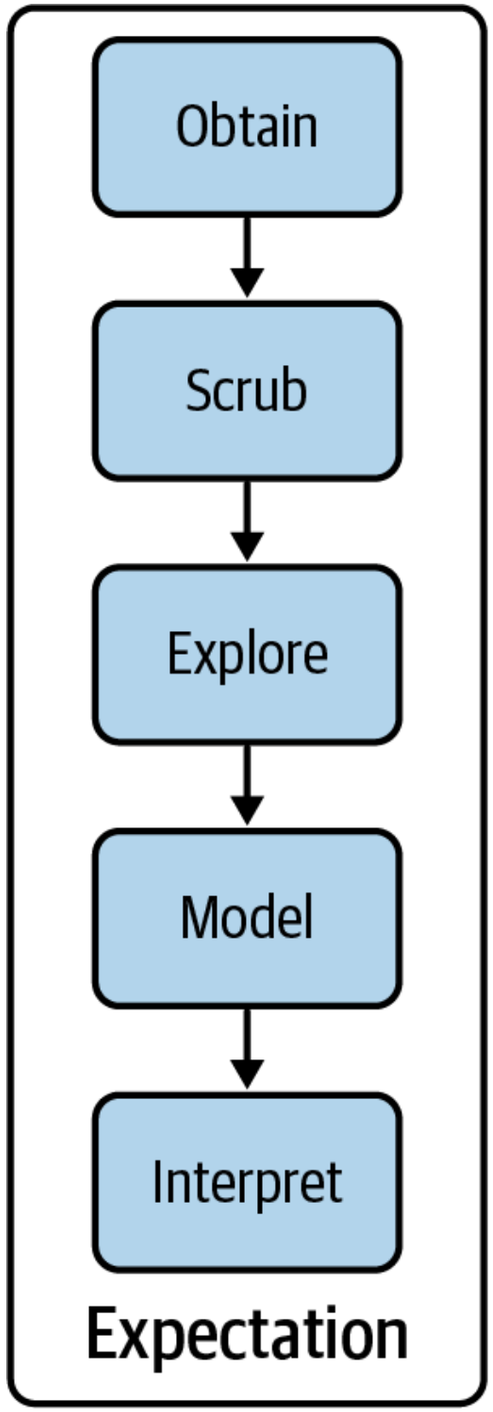
- Astronomía
- Genómica
- Redes Sociales
- Youtube

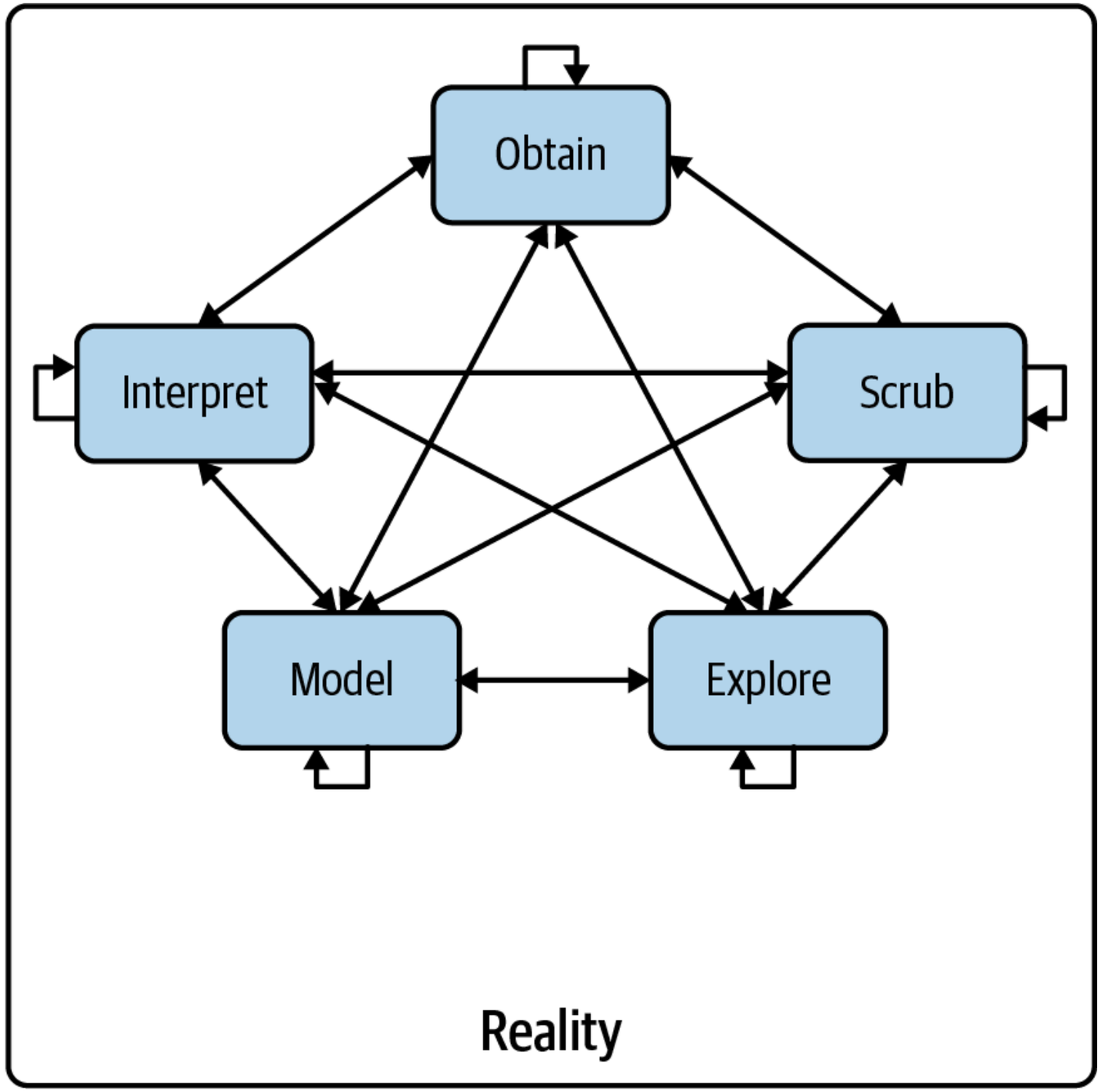
Stephens, Zachary D., et al. "**Big data: astronomical or genomical?**." *PLoS biology* 13.7 (2015): e1002195.

Data science

Expectation vs Reality





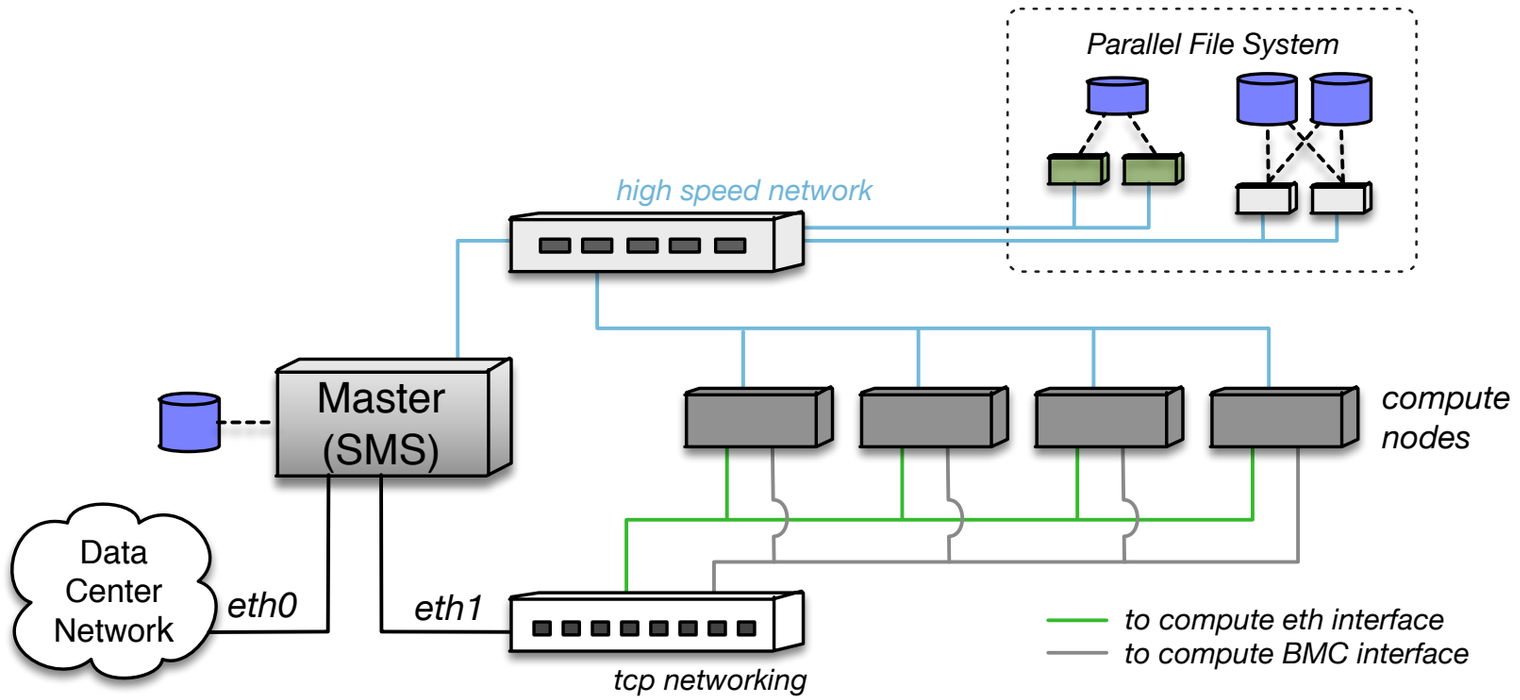


DATA -> INFORMATION

Sistemas distribuidos

Tipos

- Sistemas de cómputo distribuido
  - Clúster de cómputo
    - Hardware similar
    - Red local de alta velocidad
    - Mismo sistema operativo



The **master** handles the allocation of nodes to a particular parallel program, maintains a batch queue of submitted jobs, and provides an interface for the users of the system.

Linux

A operating system

User Processes

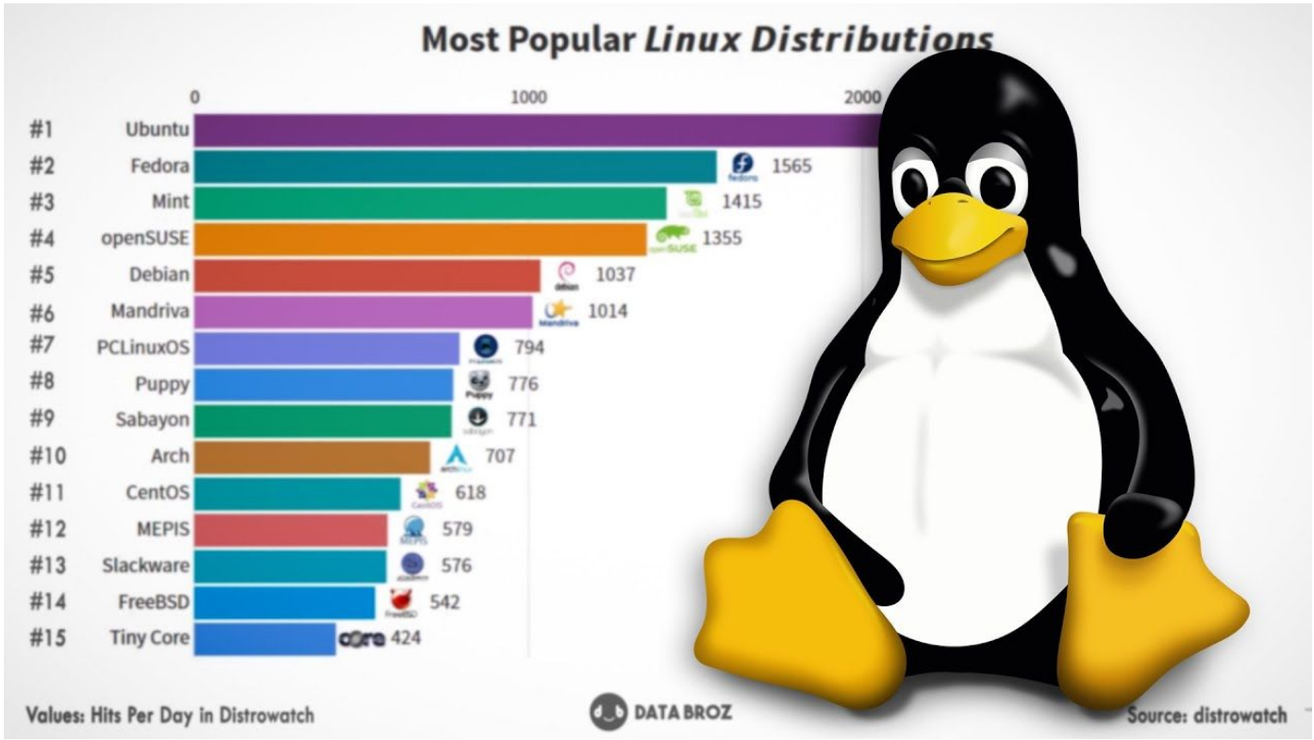
- Graphical User Interface
- Servers
- Shell

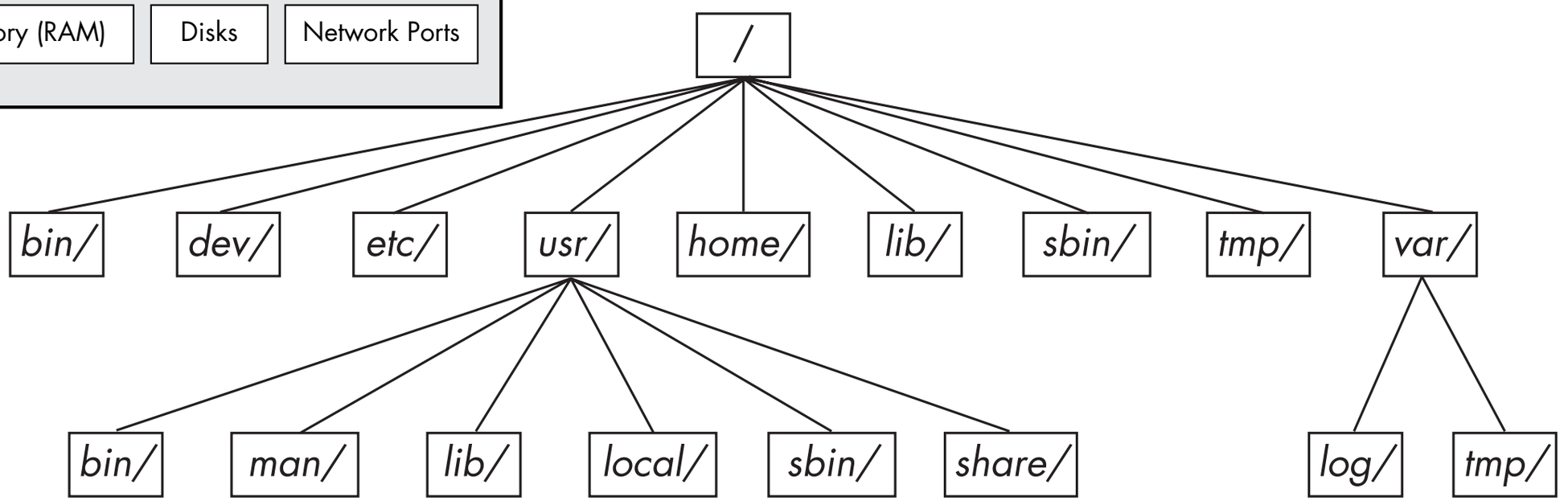
Linux Kernel

- System Calls
- Process Management
- Memory Management
- Device Drivers

Hardware

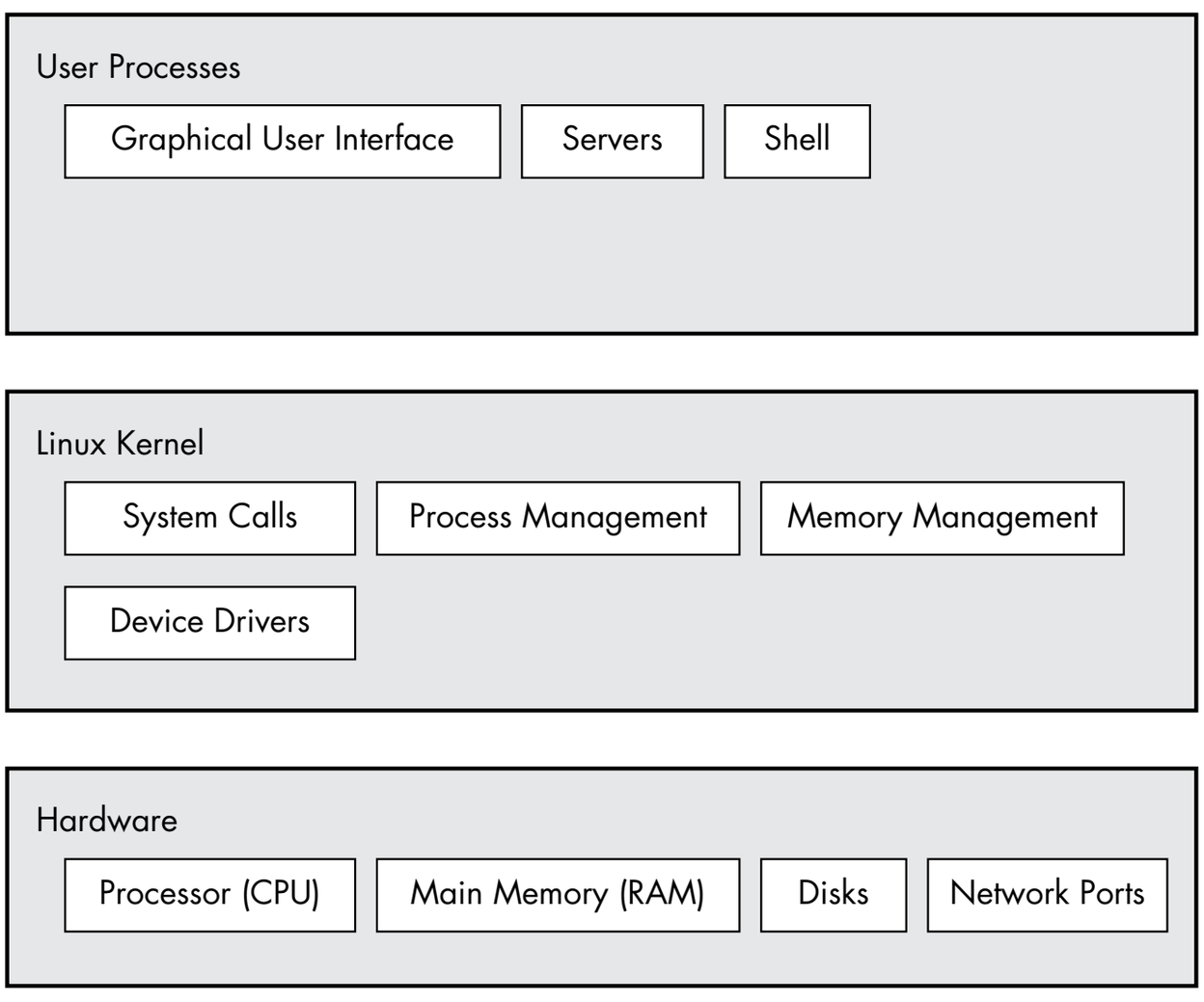
- Processor (CPU)
- Main Memory (RAM)
- Disks
- Network Ports



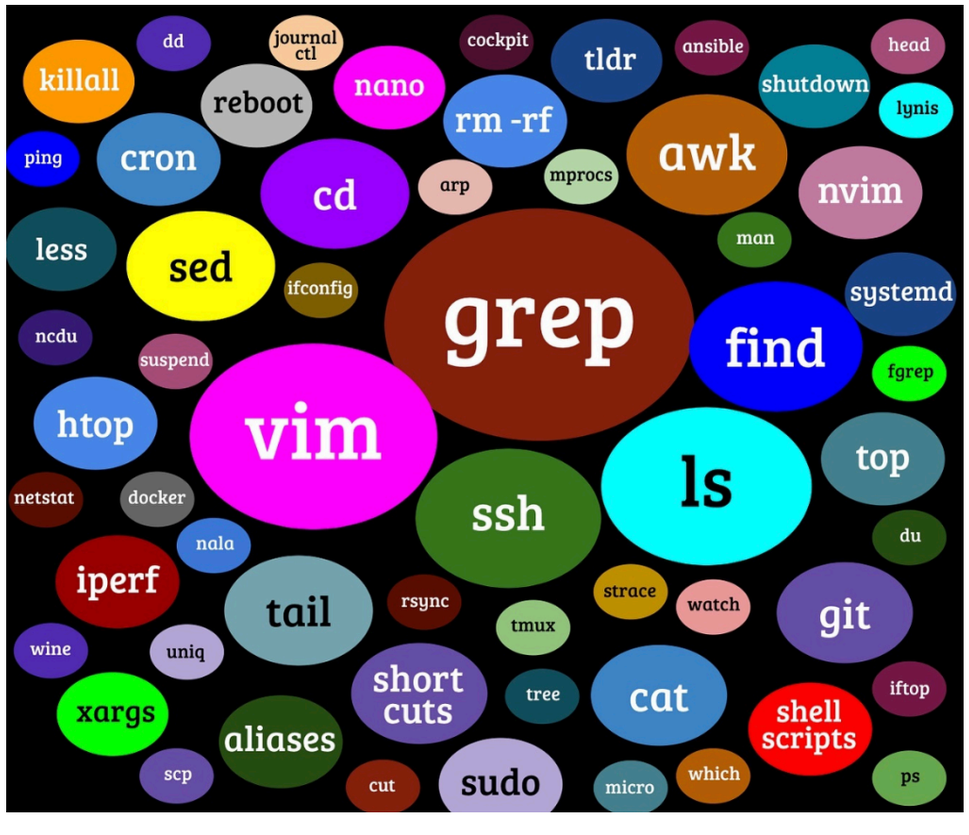


Linux

A operating system



- The *shell* is a program that runs commands.
- The *shell* also serves as a small programming environment.



- https://github.com/trinib/Linux-Bash-Commands

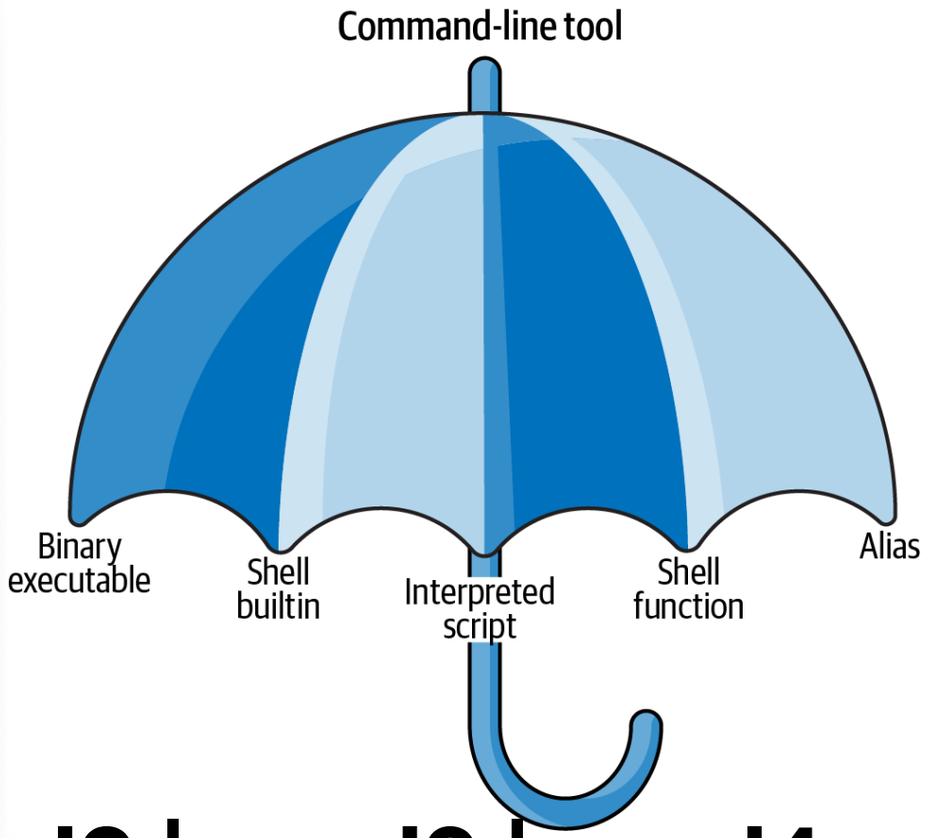
Linux

Shell command line

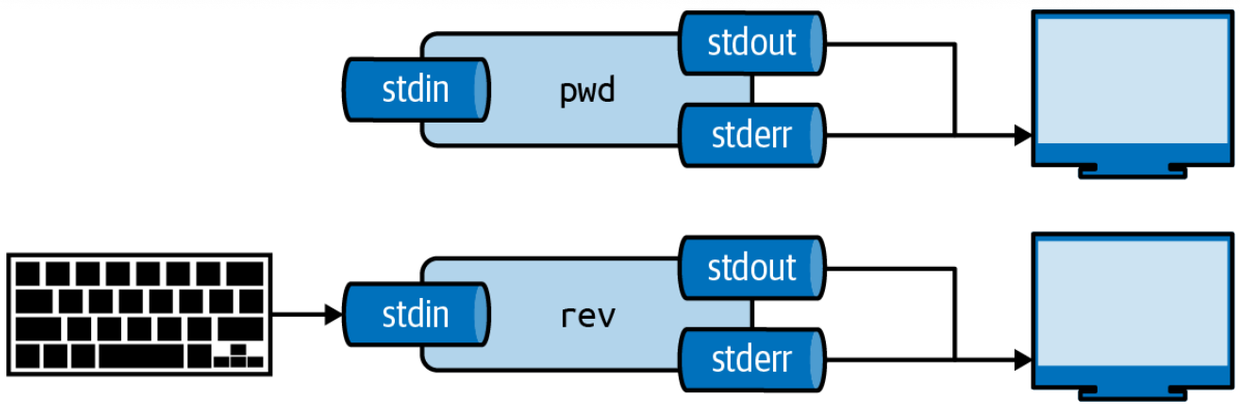
clases — adigenova@host04:~ — ssh adigenova@172.16.105.104 — 80×23

...@host04:~ — -bash ... | ...R/pipelines — -bash ... | ...ethylation — -bash ... | ...trelka_pool — -bash ... | ...ova@172.16.105.104 | ...elka_pool] — -bash ... | ...egfs/home — -bash ... | +

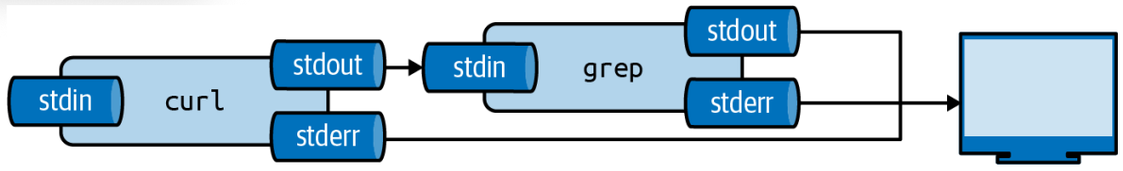
```
[adigenova@host04 ~]$ echo "The command line is the force DCBI" | cowsay -f tux
 ___________________________________
< The command line is the force DCBI >
 -----------------------------------
   \
    \
        .--.
       |o_o |
       |:_/ |
      //   \ \
     (|     | )
    /'\_   _/`\
    \___)=(___/

[adigenova@host04 ~]$
```



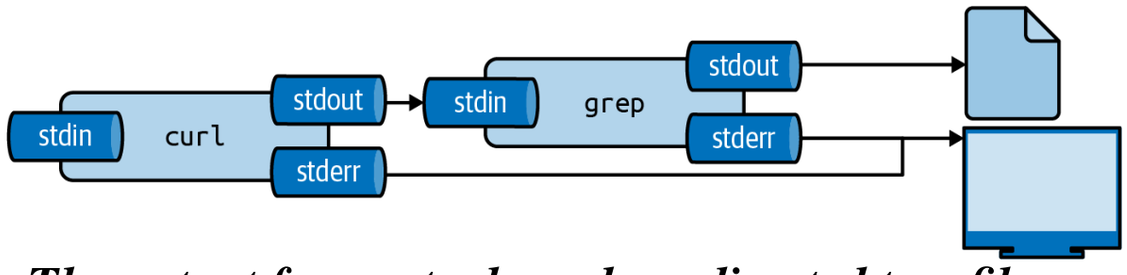
cmd1 | cmd2 | cmd3 | cmd4 ...



Every tool has three standard streams: standard input (stdin), standard output (stdout), and standard error (stderr)



The output from a tool can be piped to another tool



The output from a tool can be redirected to a file

Linux

Basic shell comands

User Management

```
sudo su                                  # Switch to root user
sudo foo                                 # Execute commands(has permission denied) as the root user
sudo nano /foo/foo.txt                   # Open directories and files(is not writable) as the root user
su username                              # Switch to a different user

passwd                                   # To change the password of a user
adduser username                         # To add a new user
userdel username                         # To remove user
userdel -rl--remove username             # To remove user with home directory and mail spool
usermod -al--append -Gl--groups GROUPNAME USERNAME # To add a user to a group
deluser USER GROUPNAME                   # To remove a user from a group

last                                     # Display information of all the users logged in
last username                            # Display information of a particular user
w                                        # Display who is online
```

Terminal Multiplexers

Start multiple terminal sessions. Active sessions persist r

```
tmux             # Start a new session (CTRL-b + d to detach)
tmux ls          # List all sessions
tmux attach -t 0 # Reattach to a session

screen           # Start a new session (CTRL-a + d to detach)
screen -S foo    # Start a new named session
screen -ls       # List all sessions
screen -R 31166  # Reattach to a session
```

System Information

```
uname -s                 # Print kernel name
uname -r                 # Print kernel release
uname -m                 # Print Architecture
uname -o                 # Print Operating System
uname -a                 # Print all Systen info

lsb_release -a           # Print distribution-specific information
dpkg --print-architecture # Print-architecture by name

cat /proc/cpuinfo        # Show cpu info
cat /proc/meminfo        # Show memory info
```

Secure Shell Protocol (SSH)

```
ssh hostname                 # Connect to hostname using your current user name (p 22)
ssh -i foo.pem hostname      # Connect to hostname using the identity file
ssh user@hostname            # Connect to hostname using the user over the default SSH port 22
ssh user@hostname -p 8765    # Connect to hostname using the user over a custom port
ssh ssh://user@hostname:8765 # Connect to hostname using the user over a custom port
```

Secure Copy

```
scp foo.txt ubuntu@hostname:/home/ubuntu          # Copy foo.txt into the specified remote directory
scp ubuntu@hostname:/home/ubuntu/foo.txt .    # Copy foo.txt from the specified remote directory
```

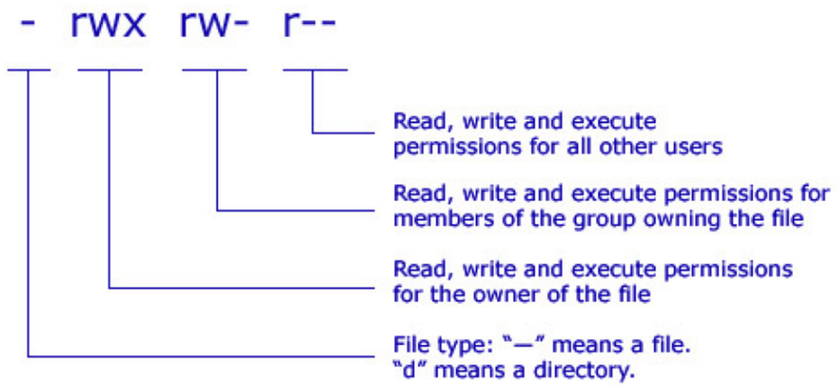
https://linuxupskillchallenge.org/

Manipulating files

Linux

Everything is a file

- File types
  - Regular files (text, images)
  - Directories (list of files)
  - Symbolic links
  - Devices and peripherals
  - Pipes
  - Sockets (inter-process communication)
- File names
  - Case sensitive, any character (/), extension not needed (but user?)
- File paths
  - Relative path: ../work/file1.txt
  - Absolute path: /home/adigenova/work/file1.txt
- File access rights
  - Types
    - Read(r), write(w), execute(x)
  - Levels
    - User (u), Group (g), Other(o)



Linux

Everything is a file

- Common text files
  - CSV -> comma-separated values
  - TSV -> Tab-separated values
  - TXT -> text file (space separated)
  - xls|xlsx -> Excel files (text or binary?)
  - SQL database

- Getting data
  - curl -LO https://burntsushi.net/stuff/worldcitiespop.csv
  - wget https://burntsushi.net/stuff/worldcitiespop.csv
  - Tar zxvf file.tar.gz, gzip , gz, zip … (compressed data)
  - Scp, cp, rsync …
- Text file editors
  - VIM/Vi
  - Nano

https://github.com/awesomedata/awesome-public-datasets

Linux

Working with text files line by line

- Viewing and Creating Files: ***cat, tac, more, less, echo, touch***
- Editing Files: ***nano, vi, vim***
- File Content Manipulation:
    - ***grep:*** Searches for patterns within files; can be used to find specific text.
    - ***sed:*** Stream editor for filtering and transforming text in a file or input from a pipeline.
    - ***awk:*** A programming language designed for text processing and manipulation.
    - ***paste:*** Merges lines of files horizontally.
    - ***cut, sort, uniq, tr, wc, head, tail.***

Linux

Working with text files line by line

- Comparing Files:
  - **diff :** Compares files line by line.
- Regular Expressions and Pattern Matching: ***grep, egrep, fgrep, sed, awk***
- Text Processing:
  - ***join:*** Joins lines of two files on a common field.
  - ***split:*** Splits a file into fixed-size pieces.
  - ***nl*** (add line numbers), ***expand*** (tab to spaces)***, unexpand*** (spaces to tab)***, fold y fmt*** (text format).
- File Encoding and Character Conversion: ***dos2unix, unix2dos***

Linux

Working with text files line by line

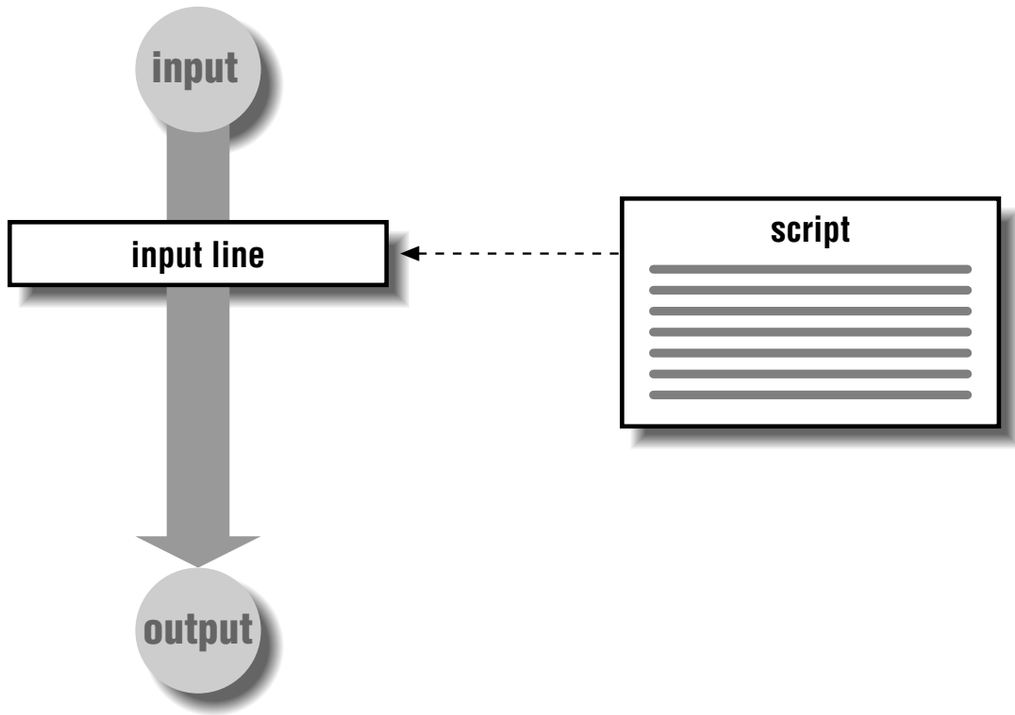
```
Country,City,AccentCity,Region,Population,Latitude,Longitude
ad,aixas,Aixàs,06,,42.4833333,1.4666667
ad,aixirivali,Aixirivali,06,,42.4666667,1.5
ad,aixirivall,Aixirivall,06,,42.4666667,1.5
ad,aixirvall,Aixirvall,06,,42.4666667,1.5
ad,aixovall,Aixovall,06,,42.4666667,1.4833333
ad,andorra,Andorra,07,,42.5,1.5166667
ad,andorra la vella,Andorra la Vella,07,20430,42.5,1.5166667
ad,andorra-vieille,Andorra-Vieille,07,,42.5,1.5166667
ad,andorre,Andorre,07,,42.5,1.5166667
```

- head worldcitiespop.csv
- wc -l worldcitiespop.csv
- head -100 worldcitiespop.csv > sample_worldcitiespop.csv
- cut -d',' -f1,5 worldcitiespop.csv > country_population.csv
- tail -n +2 worldcitiespop.csv | sort -t',' -k5 -nr | head -10
- cut -d',' -f1 worldcitiespop.csv | sort | uniq | tail -n +2
- awk -F',' ‘NR>1 && $5 > 1000000' worldcitiespop.csv
- awk -F',' 'NF > 1 {pop[$1]+=$5} END{ for(c in pop){print c" "pop[c]}}' worldcitiespop.csv | sort -rn -k2,2 | head
- Add country name using join..

Linux

Sed and Awk —- Power tools for editing files

- The basic function of awk is to search files for lines (or other units of text) that contain certain patterns.
  - A Pattern-Matching Programming Language
  - awk 'program' input-file1 input-file2 ...
  - View a text file as a textual database made up of records and fields.
  - Use variables to manipulate the database.
  - Use arithmetic and string operators.
  - Use common programming constructs such as loops and conditionals.
  - Generate formatted reports.



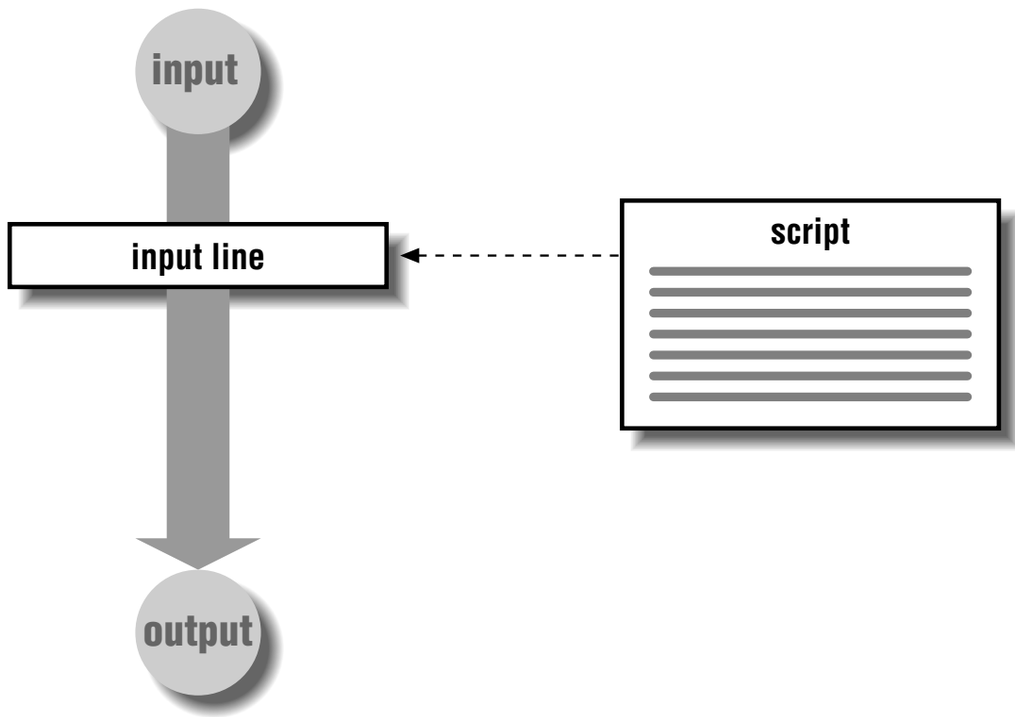
- Sed
  - Sed is a “non-interactive” stream-oriented editor.
  - To automate editing actions to be performed on one or more files.
  - To simplify the task of performing the same edits on multiple files.
  - To write conversion programs.

```
sed ’s/2022/2024/g’ file.txt
sed ’s/2022/2024/g ; s/casa/home/g’ file.txt

awk ‘{print $1}’ file.txt
awk ‘{print $1}’ *.txt
awk ‘/perro/ {print $0}’ *.txt
```

Linux

Sed and Awk —- Power tools for editing files

- Sed
  - Sed is a “non-interactive” stream-oriented editor.
  - To automate editing actions to be performed on one or more files.
  - To simplify the task of performing the same edits on multiple files.
  - To write conversion programs.



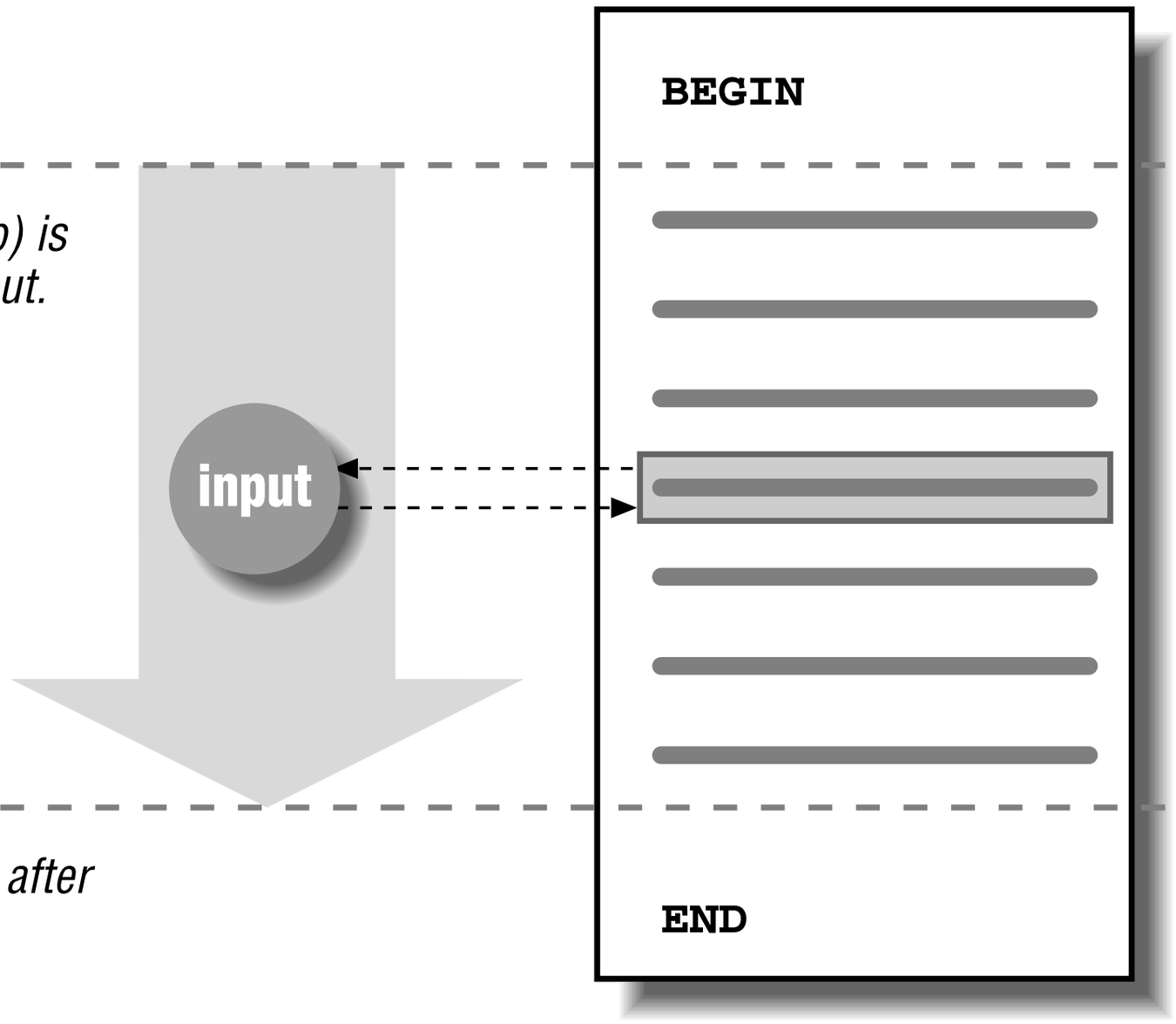

```
sed ’s/2022/2024/g’ file.txt
sed ’s/2022/2024/g ; s/casa/home/g’ file.txt

awk ‘{print $1}’ file.txt
awk ‘{print $1}’ *.txt
awk ‘/perro/ {print $0}’ *.txt
```

Linux

Awk

1 *1st routine is executed once before any input is read.*

2 *2nd routine (main input loop) is executed for each line of input.*



3 *3rd routine is executed once after all input is read.*

Linux
Awk

- if (*expression*) *action1*; [else *action2*]

expr ? *action1* : *action2*

grade = (avg >= 65) ? "Pass" : "Fail"

while (*condition*)

action

for (*set_counter* ; *test_counter* ; *increment_counter*)

action

array[*subscript*] = *value*

for (*variable* in *array*)

do something with array[*variable*

Awk Function	Description
cos(*x*)	Returns cosine of *x* (*x* is in radians).
exp(*x*)	Returns *e* to the power *x*.
int(*x*)	Returns truncated value of *x*.
log(*x*)	Returns natural logarithm (base-*e*) of *x*.
sin(*x*)	Returns sine of *x* (*x* is in radians).
sqrt(*x*)	Returns square root of *x*.
atan2(*y*,*x*)	Returns arctangent of *y*/*x* in the range -π to π.
rand( )	Returns pseudo-random number *r*, where 0 <= *r* < 1.
srand(*x*)	Establishes new seed for **rand( )**. If no seed is specified, uses time of day. Returns the old seed.

Linux

Awk

Awk Function	Description
gsub(*r*,*s*,*t*)	Globally substitutes *s* for each match of the regular expression *r* in the string *t*. Returns the number of substitutions. If *t* is not supplied, defaults to **$0**.
index(*s*, *t*)	Returns position of substring *t* in string *s* or zero if not present.
length(*s*)	Returns length of string *s* or length of **$0** if no string is supplied.
match(*s*,*r*)	Returns either the position in *s* where the regular expression *r* begins, or 0 if no occurrences are found. Sets the values of **RSTART** and **RLENGTH**.
split(*s*, *a*, *sep*)	Parses string *s* into elements of array *a* using field separator *sep*; returns number of elements. If *sep* is not supplied, **FS** is used. Array splitting works the same way as field splitting.
sprintf("*fmt*", *expr*)	Uses **printf** format specification for **expr**.
sub(*r*,*s*,*t*)	Substitutes *s* for first match of the regular expression *r* in the string *t*. Returns 1 if successful; 0 otherwise. If *t* is not supplied, defaults to **$0**.
substr(*s*,*p*, *n*)	Returns substring of string *s* at beginning position *p* up to a maximum length of *n*. If *n* is not supplied, the rest of the string from *p* is used.
tolower(*s*)	Translates all uppercase characters in string *s* to lowercase and returns the new string.
toupper(*s*)	Translates all lowercase characters in string *s* to uppercase and returns the new string.

Awk

Examples

```
$ cat table.txt
brown bread mat hair 42
blue cake mug shirt -7
yellow banana window shoes 3.14

# print the second field of each input line
$ awk '{print $2}' table.txt
bread
cake
banana

# print lines only if the last field is a negative number
# recall that the default action is to print the contents of $0
$ awk '$NF<0' table.txt
blue cake mug shirt -7

# change 'b' to 'B' only for the first field
$ awk '{gsub(/b/, "B", $1)} 1' table.txt
Brown bread mat hair 42
Blue cake mug shirt -7
yellow banana window shoes 3.14
```

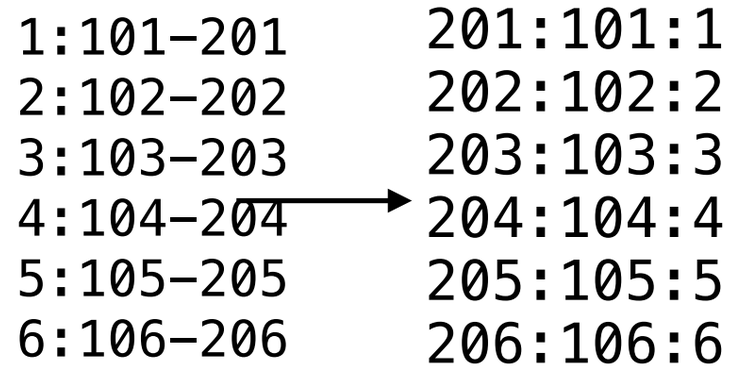
```
awk 'cond1{action1} cond2{action2} ... condN{actionN}'
```

https://learnbyexample.github.io/learn_gnuawk/

Linux

Sed

- Substitutions
  - /Pattern/replacement/flags
    - Flags : n <replace the n-matchin pattern>,g<global>,I <insensitive case>,p <print pattern>
- Transform
  - s/abc/xyz/
  - y/abcdefghijklmnopqrstuvwxyz/ABCDEFGHIJKLMNOPQRSTUVWXYZ/
- Regular expressions with Sed
  - sed 's/\(.*\):\(.*\)-\(.*\)/\3:\2:\1/' test.txt



Linux

Regular expressions

Basic Syntax

- /.../: Start and end
- |: Alternation
- (): Grouping

Groups and Ranges

- .: Any character except newline (\n)
- (a|b): a or b
- (…): Group
- (?:…): Passive (non-capturing) group
- [abc]: a, b or c
- [^abc]: Not a, b or c
- [a-z]: Letters from a to z
- [A-Z]: Uppercase letters fro
- [0-9]: Digits from 0 to 9

Quantifiers

- *: 0 or more
- +: 1 or more
- ?: 0 or 1
- {3}: Exactly 3
- {3,}: 3 or more
- {3,5}: 3, 4 or 5

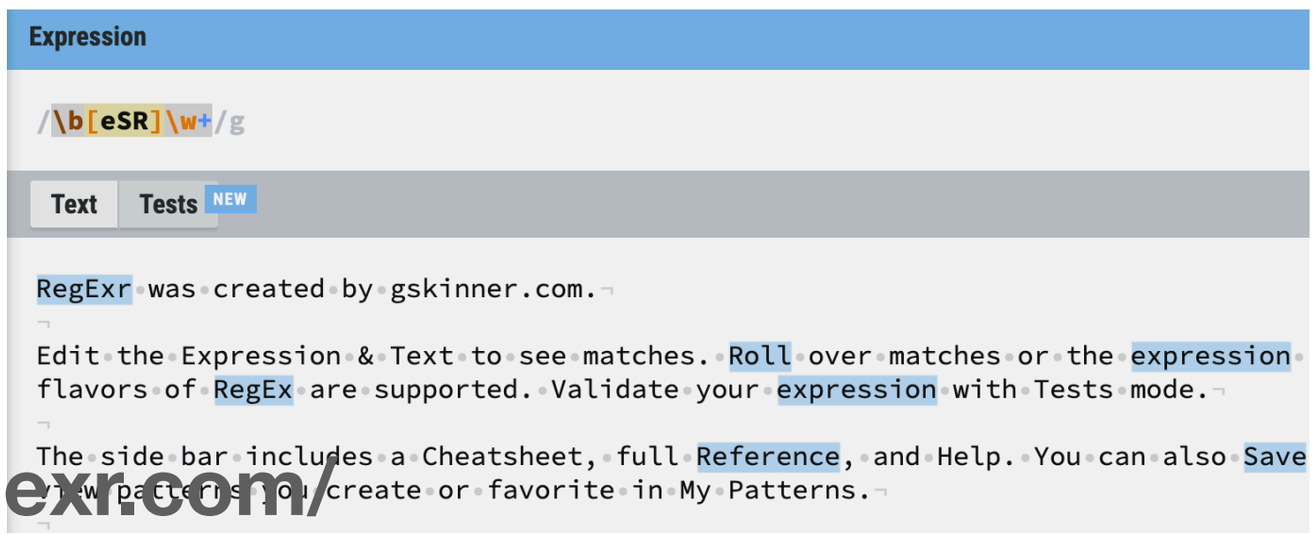
Position Matching

- ^: Start of string or start
- $: End of string or end o
- \b: Word boundary
- \B: Not word boundary

Character Classes

- \s: Whitespace
- \S: Not whitespace
- \w: Word
- \W: Not word
- \d: Digit
- \D: Not digit
- \x: Hexadecimal digit
- \O: Octal digit

https://regexr.com/



Linux

Sed & regex examples

```
$ cat anchors.txt
sub par
spar
apparent effort
two spare computers
cart part tart mart

# words starting with 'par'
$ sed -n '/\bpar/p' anchors.txt
sub par
cart part tart mart

# words ending with 'par'
$ sed -n '/par\b/p' anchors.txt
sub par
spar


$ sed -nE '/two|sub/p' anchors.txt
sub par
two spare computers

# match 'cat' or 'dog' or 'fox'
# note the use of 'g' flag for mul
$ echo 'cats dog bee parrot foxed' | sed -E 's/cat|dog|fox/--/g'
--s -- bee parrot --ed
```

```
$ echo 'ac abc abbc abbbc abbbbbbbbc' | sed -E 's/ab{1,4}c/X/g'
ac X X X abbbbbbbbc

$ echo 'ac abc abbc abbbc abbbbbbbbc' | sed -E 's/ab{3,}c/X/g'
ac abc abbc X X

$ echo 'ac abc abbc abbbc abbbbbbbbc' | sed -E 's/ab{,2}c/X/g'
X X X abbbc abbbbbbbbc

$ echo 'ac abc abbc abbbc abbbbbbbbc' | sed -E 's/ab{3}c/X/g'
ac abc abbc X abbbbbbbbc

$ echo 'fd fed fod fe:d feeeeder' | sed 's/fe*d/X/g'
X X fod fe:d Xer

# zero or more of '1' followed by '2'
$ echo '31111111111125111142' | sed 's/1*2/-/g'
3-511114-

# from the first 'b' to the last 't' in the line
$ echo 'car bat cod map scat dot abacus' | sed 's/b.*t/-/'
car - abacus

# from the first 'b' to the last 'at' in the line
$ echo 'car bat cod map scat dot abacus' | sed 's/b.*at/-/'
car - dot abacus
```

https://learnbyexample.github.io/learn_gnused/cover.html

Awk

Operators

Operator	Effect
`lvalue` `+=` *`increment`*	Add *`increment`* to the value of *`lvalue`*.
`lvalue` `-=` *`decrement`*	Subtract *`decrement`* from the value of *`lvalue`*.
`lvalue` `*=` *`coefficient`*	Multiply the value of *`lvalue`* by *`coefficient`*.
`lvalue` `/=` *`divisor`*	Divide the value of *`lvalue`* by *`divisor`*.
`lvalue` `%=` *`modulus`*	Set *`lvalue`* to its remainder by *`modulus`*.
`lvalue` `^=` *`power`*	Raise *`lvalue`* to the power *`power`*.
`lvalue` `**=` *`power`*	Raise *`lvalue`* to the power *`power`*. (c.e.)

Awk

Operators

Expression	Result
x < *y*	True if *x* is less than *y*
x <= *y*	True if *x* is less than or equal to *y*
x > *y*	True if *x* is greater than *y*
x >= *y*	True if *x* is greater than or equal to *y*
x == *y*	True if *x* is equal to *y*
x != *y*	True if *x* is not equal to *y*
x ~ *y*	True if the string *x* matches the regexp denoted by *y*
x !~ *y*	True if the string *x* does not match the regexp denoted by *y*
subscript in *array*	True if the array *array* has an element with the subscript *subscript*

Awk & sed

Problems

1.Count the frequency of words in a text file.

2.Change a hello for bye in the 10-20 lines of a file.

3.Merge two files by a common field.

4.Create a table combining one or more fields from several files.

5.Count 100 most frequent 15-mer of a fasta sequence file.

6.Transpose of a numeric matrix.

XSV, Youplot and csvtk

- **xsv** is a command line program for indexing, slicing, analyzing, splitting and joining CSV files.
  - https://github.com/BurntSushi/xsv
- **YouPlot**
  - command line tool that draws plots on the terminal.
- **Csvtk** is convenient for rapid data investigation and easily integrated into analysis pipelines.
  - https://github.com/shenwei356/csvtk

```
$ curl -sL https://git.io/ISLANDScsv \
> | sort -nk2 -t, \
> | tail -n15 \
> | uplot bar -d, -t "Areas of the World's Major Landmasses"
                Areas of the World's Major Landmasses

        Britain ┤ 84.0
          Honshu ┤ 89.0
        Sumatra ┤ 183.0
          Baffin ┤ 184.0
      Madagascar ┤ 227.0
         Borneo ┤ 280.0
      New Guinea ┤ 306.0
       Greenland ┤ 840.0
       Australia ┤ 2968.0
          Europe ┤ 3745.0
      Antarctica ┤ 5500.0
   South America ┤ 6795.0
   North America ┤ 9390.0
          Africa ┤ 11506.0
            Asia ┤ 16988.0
```

```
$ curl -sL https://git.io/IRIStsv \
> | cut -f1-4 \
> | uplot boxplot -H -t IRIS
                          IRIS

   sepal_length ├──────┤──────────┤──────┤
    sepal_width      ├──┤──┤──┤
   petal_length ├──────────┤──────┤──────┤
    petal_width ├──┤──┤──┤
                0          4          8
$
```