University of Zurich[UZH]

URPP
Evolution in action

BIO609: Introduction to Linux

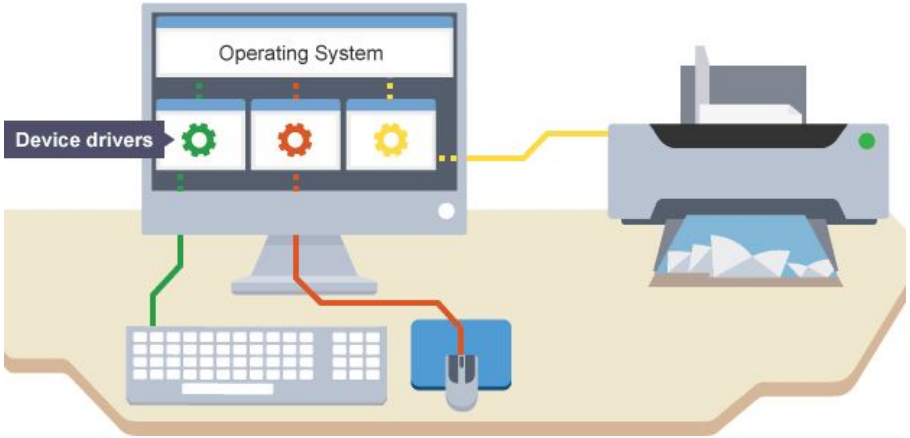
Basics of the command-line

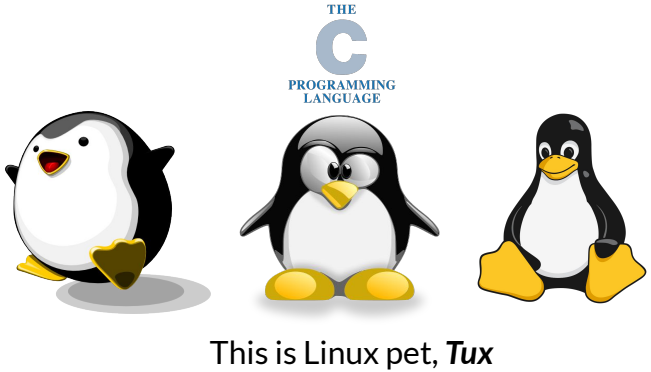
April 20th, 2021 - Carla Bello

Overview

1. **Linux terminal:** definition and environment
2. **Working directories**: Moving in the terminal
3. **Files:** types and manipulation
4. **Basic commands**: list, copy, move, create, delete, etc.
5. ***grep*** and **regular expressions**: powerful pattern matching
6. **Piping and redirection**: combining commands
7. **Manuals:** getting help from Linux

Linux is a free open-source OS



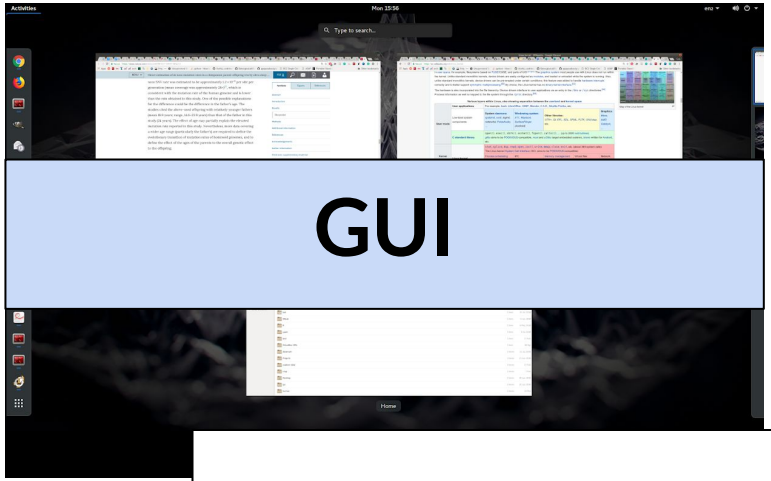


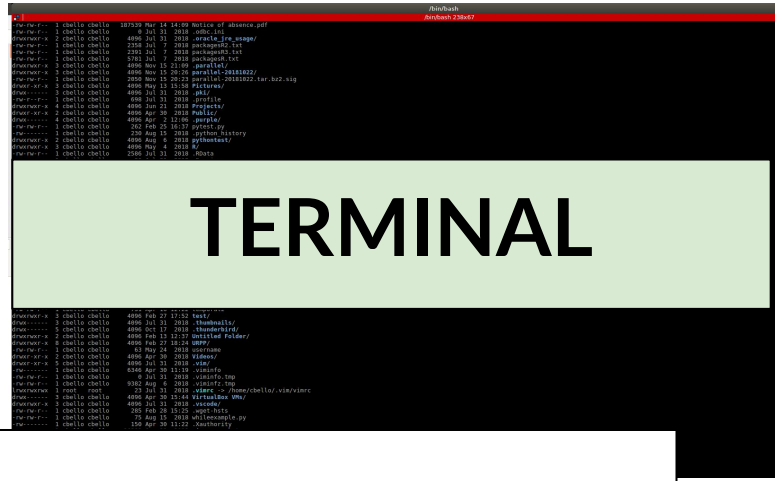
This is Linux pet, ***Tux***

- **Free open-source** operating system (OS) from the **90's** made by **Linus Torvalds**
- Based on **UNIX** from *Bells Labs* and mostly **written in C** and **Assembly**
- **Many** available **distros: Ubuntu**, Fedora, CentOS, Gentoo, Debian, etc.

Linux command-line is simple and fast

Linux has a **graphical user interface (GUI)** and its **command-line or terminal** to interact with it

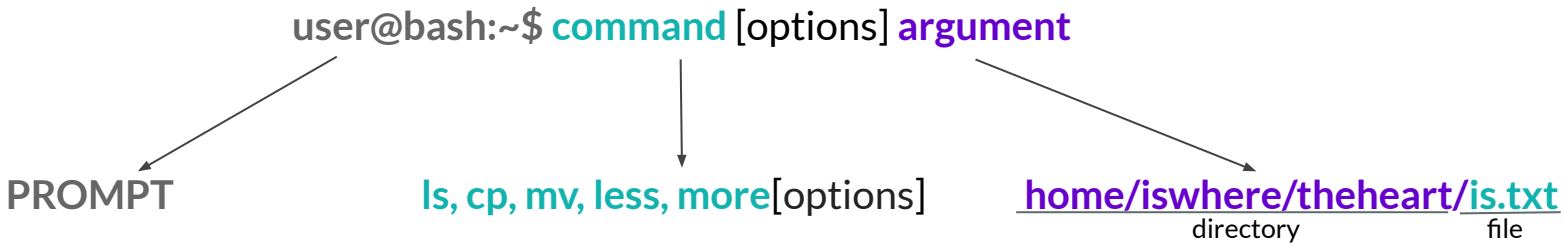




You can easily **access the terminal** by using **Ctr+Alt+T**

Linux command-line has a structure

A **command** is a **directive** to the **command-line interpreter (CLI)**, also known as **terminal**.

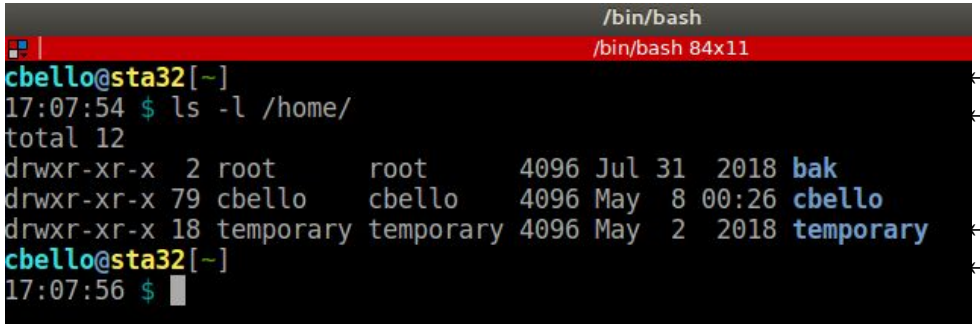



```
user@bash:~$
user@bash:~$ ls -l /home/
```

← 1st line: **terminal prompt**

← 2nd line: ***ls*** **command** together with ***-l*** **option (or flag)**
/home/ **the directory the command works on**

Basic example



Line 1: **Terminal prompt**

Line 2: ***ls*** **command** together with ***-l*** **option** ***/home/*** **the directory the command works on**

Line 3-6: Written **output**

Line 7: **Terminal prompt**

- Most of the time a command **prints an output,** but it can also be **performing a task** in which case there **will be no output** unless there is an error
- When the **command is done** or **the task is over** the **prompt will appear again**

Learning to navigate the command-line

- To know **where you are** in the command-line you can **Print the Working Directory** by using the command ***pwd*.**
- A lot of commands will **rely on the location** you **currently are**, use ***pwd*** as often as you need to remember.
- You can **modify** your profile (.bashrc) so your **prompt** always **shows your location**.
- Use the command ***ls*** **to List** the **files** in your **current location.**
- **Change Directories** by using the command ***cd*** with the **location** you want to move at, ***cd*** **without arguments** directs you to your **/home/ directory**

Most **command** names are **abbreviations** of their **function**.

Paths are means to get to a directory or file

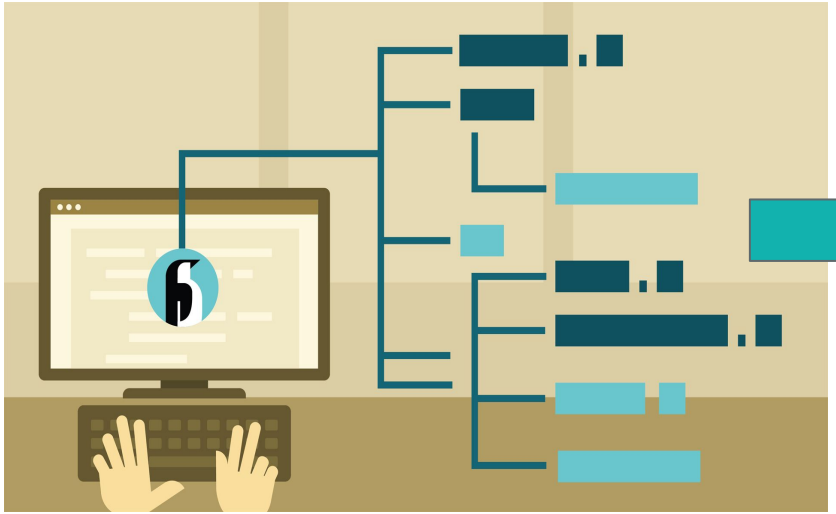


Image from Linux: System Information and Directory Structure Tools

Hierarchical system of Linux:

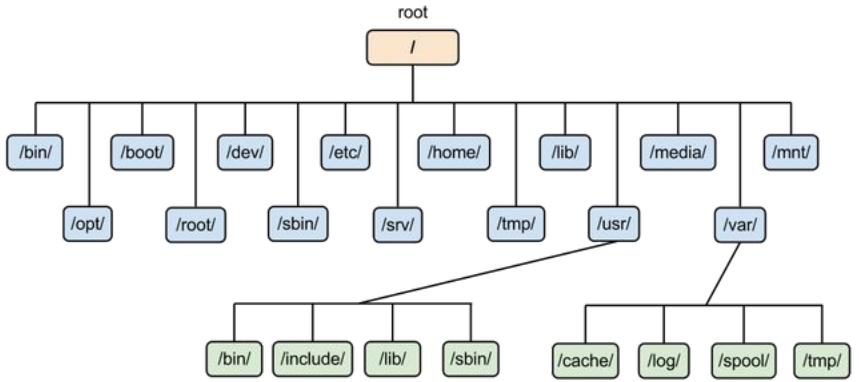
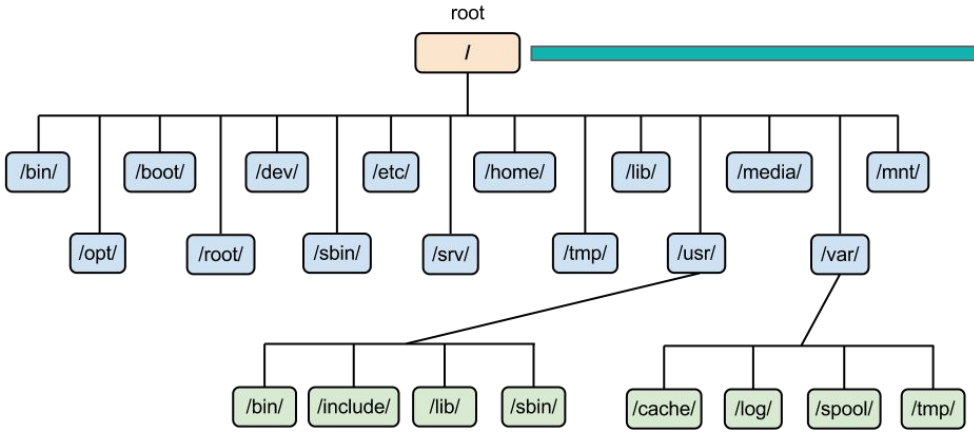


Image from http://beyondthegeek.com/2016/08/24/linux-hierarchy-structure/

Paths can be relative or absolute



Absolute paths:

- Specify a location (file or directory) in relation to the **root directory**
- They **begin** with a **forward slash (/)**

Relative paths:

- Specify a location (file or directory) in relation to the **your current location in the system**
- They do not **begin** with a **forward slash.**

Image from http://beyondthegeek.com/2016/08/24/linux-hierarchy-structure/

Example

- **Absolute paths:**

```
user@bash:~$
user@bash:~$ ls /home/Documents
```

← 1st line: **terminal prompt**

← 2nd line: ***ls*** **command** together with **/home/Documents/**

You **can be anywhere in the system** and you will get the **list** of files in the **/home/Documents/** directory

- **Relative paths:**

```
user@bash:~$
user@bash:~$ ls Documents/
```

← 1st line: **terminal prompt**

← 2nd line: ***ls*** **command** together with **Documents/**

You have to be at the **same the location** of the **Documents/** directory, in this particular case at the **/home/** directory

Everything in Linux is a file

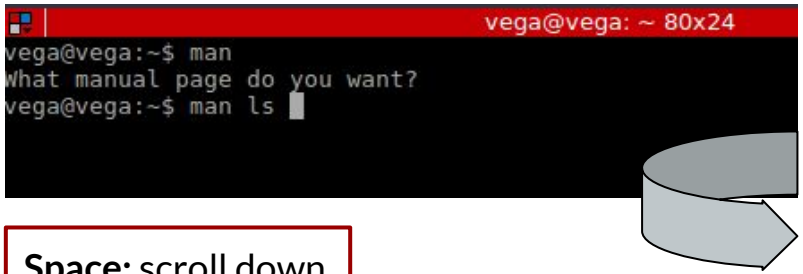
- A **text file** is a file, a **directory** is a file, a **monitor** is a file (one that the system writes only), the **keyboard** is a file (one that the system reads only).
- Linux is an **extensionless system**, i.e. it works without having to add the file type at the end of the file name (e.g. *.txt*, *.gzip*, *.fa*, etc.).
- Linux is **case sensitive,** all of these files are different: `FILE1.txt File1.txt file1.TXT`
- **Spaces** in **names should be avoided**, remember that spaces are the way we separate our commands. Use **underscore(_), dash (-),** or **dot(.)** instead.

Finding help by using manual pages

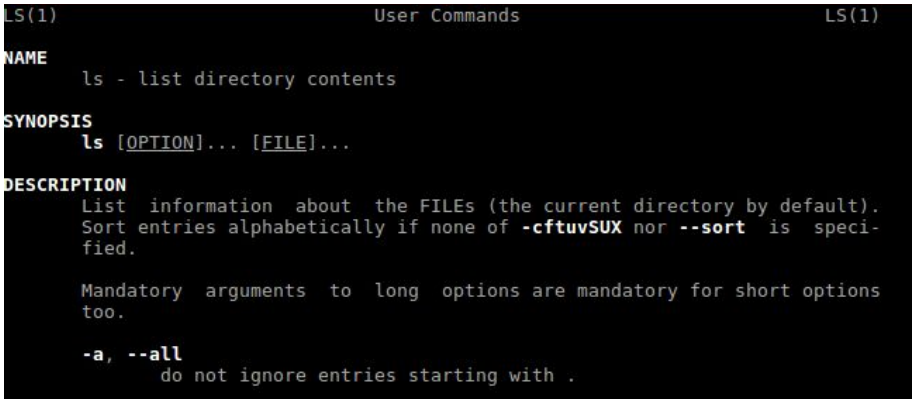
It is difficult **to remember** what **every command does**. Luckily we have **manual pages** to the rescue! 😊

`man [command]` ← Invoke the **manual page** of a **command** with **'man'**

Example: Manual page for **ls** (listing files/directories)



Space: scroll down
b : scroll up
q: quit



File manipulation

Remember in Linux **everything is a file** and there are **powerful commands** to manipulate files.

Task	Command
Make or **remove** directories	***mkdir, rmdir***
Copy files/directories	***cp***
Move or rename files/directories	***mv***
Remove files/directories	***rm***

There is **no undo for deleting** a file or directory. **Be careful!**

More powerful commands

Task	Command
Show content	*less, more, head, tail, cat*
Search/Extract/Modify	*grep, cut, uniq, awk*
Manipulate	*sort, tr, sed, awk, join, paste*
Replace	*tr, sed, awk*
Count	*wc, uniq*
Compare	*comm, diff*

Example

```
user@bash:~$ mkdir Intro2Linux
user@bash:~$ cd Intro2Linux/
user@bash:~/Intro2Linux$ touch emptyfile.txt
user@bash:~/Intro2Linux$ less emptyfile.txt
user@bash:~/Intro2Linux$
```

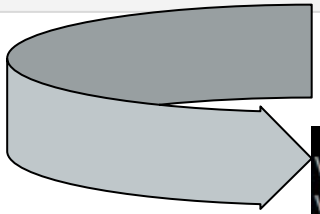
← Line 1: **make** directory **“Intro2Linux”**

← Line 2: **change** directory **to “Intro2linux”**

← Line 3: **Make** an **empty file**, called **emptyfile.txt**

← Line 4: **show emptyfile.tx**t and **exit** with **Ctrl+Z**

← Line 5: **Back** to **terminal prompt**



```
vega@vega:~$ mkdir Intro2Linux
vega@vega:~$ cd Intro2Linux/
vega@vega:~/Intro2Linux$ touch emptyfile.txt
vega@vega:~/Intro2Linux$ less emptyfile.txt

[14]+  Stopped                 less emptyfile.txt
vega@vega:~/Intro2Linux$ ls
emptyfile.txt
vega@vega:~/Intro2Linux$
```

Regular expressions and grep

A regular expression (RE) is a **sequence of characters** that defines a **search pattern**

```
pizza (matches the word pizza), “\t” (matches a tabular separation), “\d” (matches a digit)
```

grep is a **command-line utility** or program that **allows** you to **find REs**

- **g/re/p** : Globally search a Regular Expression and Print

There are **many regular expressions** and many uses for **grep**

Regular expressions and grep

- They allow you to create a **pattern**, **identify** and **manipulate specific bits of data**.
- They are very **powerful** and **fast**.
- They are **widely used** in programming to **match patterns** of **interest**.
- Useful for data **validation**, data **scraping**, string **parsing**, string **replacement**, **syntax highlighting**, **file renaming**, etc.

A REGEX sheet cheat

Anchors

^	Start of line
$	End of line

Character Classes

\s	White space character
\S	Non-white space character
\d	Digit character
\D	Non-digit character
\w	Word
\W	Non-word (e.g. punctuation, spaces)

Metacharacters (must be escaped)

^	[	]
$	(	)
.	{	}
*	+	?
\	|	-

GA Filter group accessors

$Ax	Access group x in field A (e.g. $A1)
$Bx	Access group x in field B (e.g. $B1)

Quantifiers

*	Zero or more (greedy)
*?	Zero or more (lazy)
+	One or more (greedy)
+?	One or more (lazy)
?	Zero or one (greedy)
??	Zero or one (lazy)
{X}	Exactly X (e.g. 3)
{X,}	X or more, (e.g. 3)
{X, Y}	Between X and Y (e.g. 3 and 5) (lazy)

Ranges and Groups

.	Any character
(a|b)	a or b (case sensitive)
(...)	Group, e.g. (keyword)
(?:...)	Passive group, e.g. (?:keyword)
[abc]	Range (a or b or c)
[^abc]	Negative range (not a or b or c)
[A-Z]	Uppercase letter between A and Z
[a-z]	Lowercase letter between a and z
[0-7]	Digit between 0 and 7

Sample Patterns

^/directory/(.*)
Any page URLs starting with /directory/

(brand\s*?term)
Brand term with or without whitespace between words

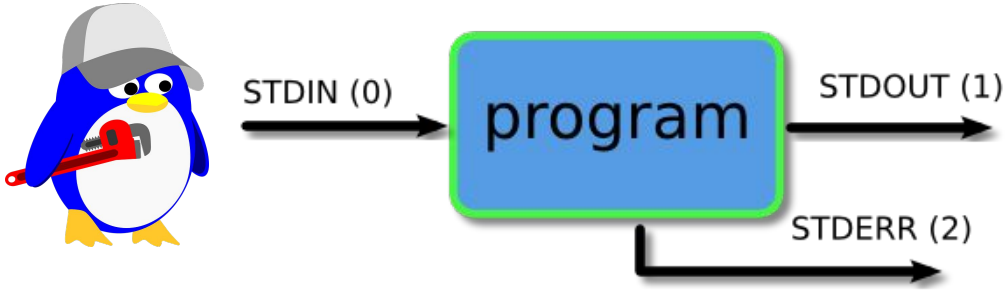
^brand\s+[^cf]
Key phrases beginning with 'brand' and the second word not starting with c or f

\.aspx$
URLs ending in '.aspx'

ORDER\-\d{6}
"ORDER-" followed by a six digit ID

(?:\?|&)utm=([^&$]+)
Value of 'utm' querystring parameter

Piping and redirection are means to connect

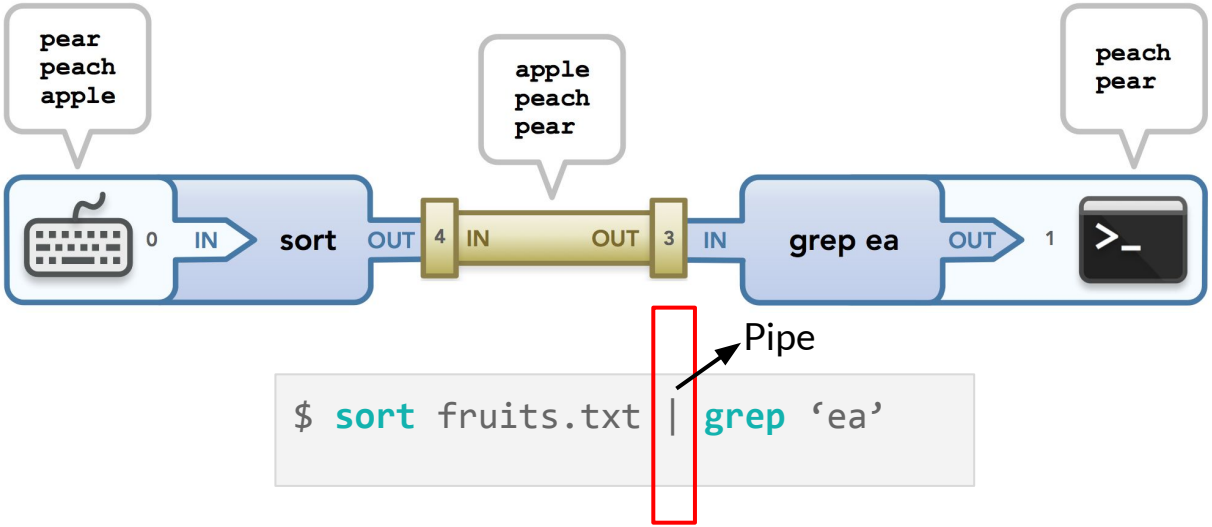


- **STDIN (0)** - Standard **input** (data **fed** into the program)
- **STDOUT (1)** - Standard **output** (data **printed** by the program, defaults to the terminal)
- **STDERR (2)** - Standard **error** (for **error** messages, also defaults to the terminal)

The means by which we may **connect** these **streams between programs** and **files** to **direct data** in **interesting ways**

https://ryanstutorials.net/linuxtutorial/piping.php

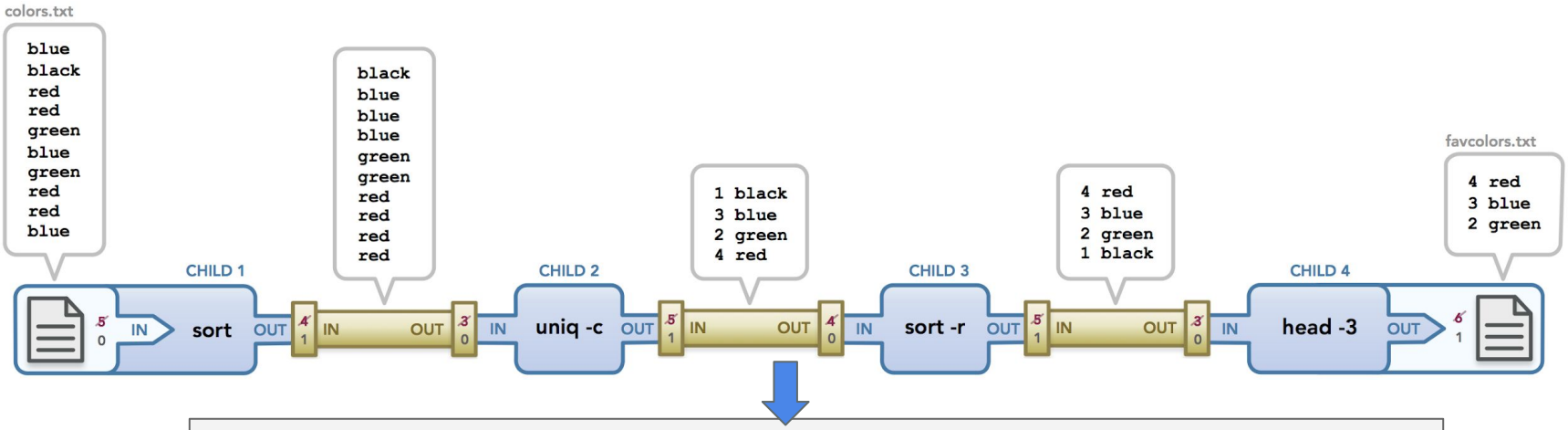
Combining commands with pipes



- **Sorting** the **file** alphabetically by fruit names
- Grabbing (with grep) **only** the fruits **matching** the "**ea**" sequence of letters

Image from: http://www.rozmichelle.com/pipes-forks-dups/

Making a new file



- **Sorting** file by color names (**default**)
- Taking only those that are **unique counts** (**-c flag**),
- Sorting by reverse order (**-r flag**)
- Taking only the first three lines (**-3**)
- **Redirection (>)** of output to a **new file**

Image from: http://www.rozmichelle.com/pipes-forks-dups/

Summary

1. **Commands** are **directives** to the CLI: command [options] **argument**
2. **Paths**, **relative** and **absolute**: everything is under **root (/).**
3. **Navigating directories**: ***pwd, cd, ls***
4. **Everything in Linux is a file**: ***mkdir, rmdir, touch, mv, cp***
5. ***grep*** and **regular expressions**: powerful **pattern matching**
6. **Piping and redirection**: combining commands using pipes "**|**" and redirecting with "**>**"
7. Don't forget to get help with **manual pages**: use "**man**"

Courses

1. **O'reilly book: Learning the Bash Shell:**

   http://the-eye.eu/public/Books/HumbleBundle/learningthebashshell_3rdedition.pdf

2. **SIB (Swiss Institute of Bioinformatics) UNIX Course :**

   https://edu.isb-sib.ch/pluginfile.php/2878/mod_resource/content/3/couselab-html/content.html

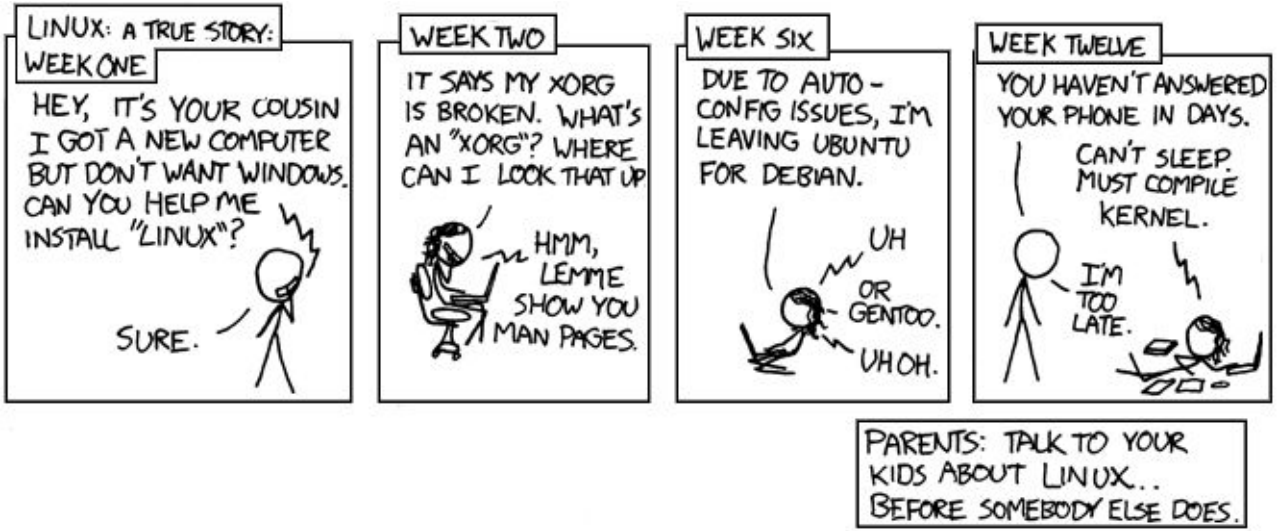
3. **StackOverflow (**www.stackoverflow.com**)**
4. **The internet (**Google, Duckduckgo)

Hands on!

Please go to:

carlalbc that is an **L**

- https://github.com/carlalbc/BIO609_2021/
- **BIO609.Linux_tutorial.md**



Additional information

Installing software

- With privileges or available packages

```
$ sudo apt-get install some-app
$ sudo update
$ sudo upgrade
$ sudo apt-get remove
```

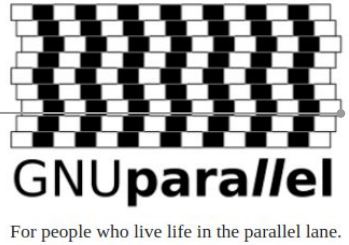
- It’s not always that easy:

  - **Dependencies** / libraries
  - **No *root*** privileges
  - **Not all packages are the same**

Common steps for installation

1. Go to the webpage and **copy the download link**
2. **Download source package/s** (you can use *wget*, *curl*, save as)
3. **Unpack the files** (you can use the **untar** alias!)
4. Go inside the folder and check the **README** file.
5. **Build if necessary** and use the **binary files** once built.
6. **Add** the **binaries PATH** to your **.bashrc**

Parallel GNU



- It's a **shell tool** for executing **jobs in parallel** using **one or more** computers.
- A job can be a **single command** or a **small script** that has to be run for **each of the lines** in the input.
- The **simultaneous execution** can occur on **remote machines** as well.

https://www.gnu.org/software/parallel/
https://www.gnu.org/software/parallel/parallel_tutorial.html