



University of  
Zurich<sup>UZH</sup>

**URPP**  
Evolution in action

# BI0609: Introduction to Linux

## Basics of the command-line

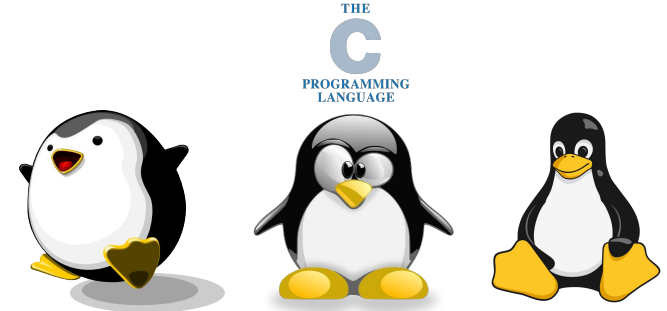
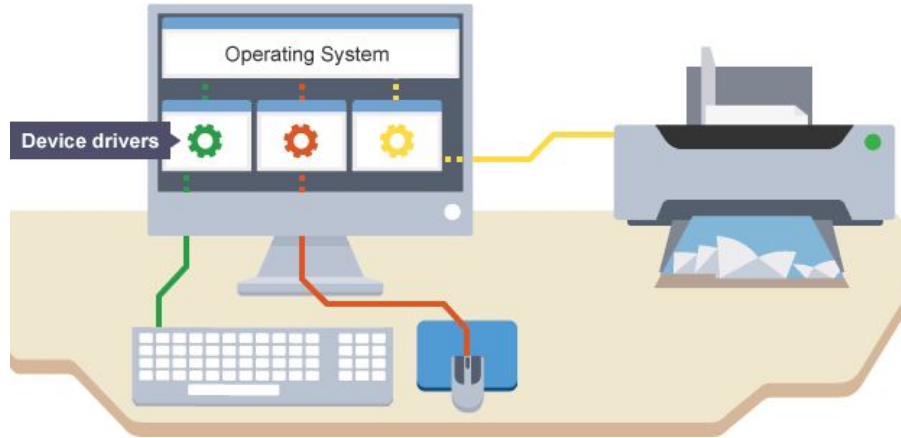
09th Nov 2020 - Carla Bello

# 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

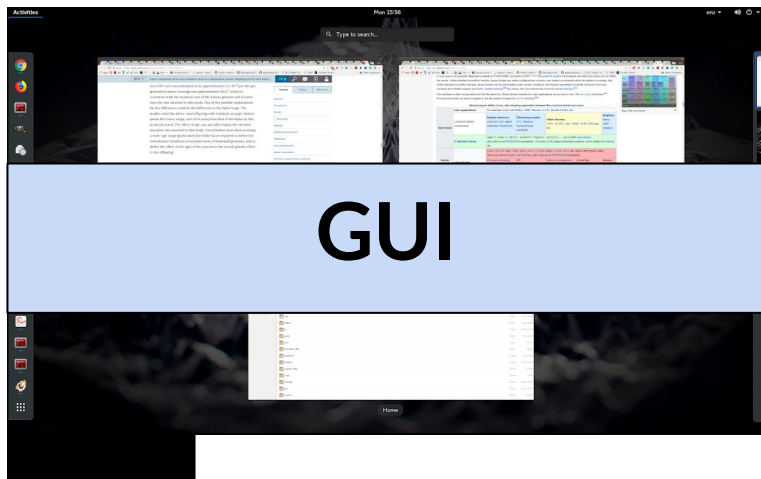


This is Linux pet, *Tux*

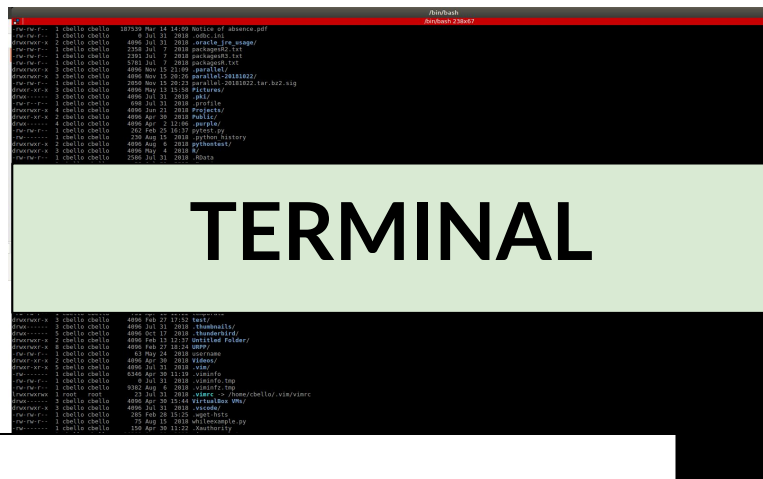
- **Free open-source** operating system (OS) from the **90's** made by **Linus Torvalds**
- Based on **UNIX** from *Bells Labs* and mostly **written in C** and **Assembly**
- **Many** available **distros**: **Ubuntu**, Fedora, CentOS, Gentoo, Debian, etc.

# Linux command-line is simple and fast

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



**GUI**

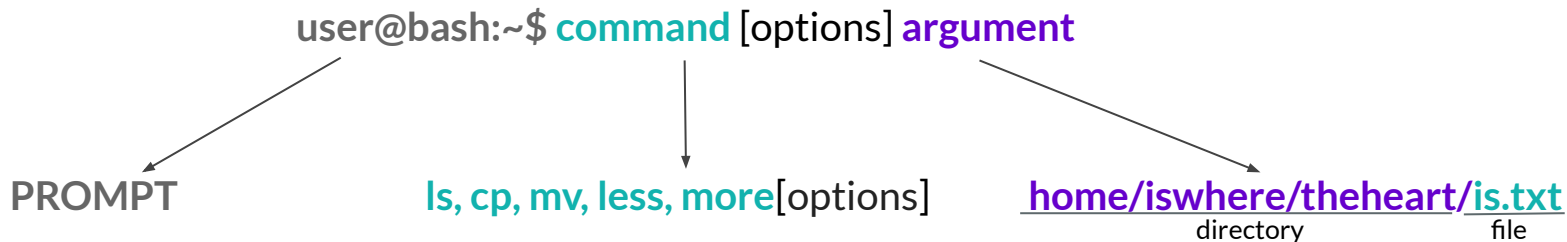


**TERMINAL**

You can easily access the terminal by using **Ctrl+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

```
/bin/bash
/bin/bash 84x11
cbello@sta32[~]
17:07:54 $ ls -l /home/
total 12
drwxr-xr-x  2 root    root    4096 Jul 31  2018 bak
drwxr-xr-x 79 cbello  cbello  4096 May  8 00:26 cbello
drwxr-xr-x 18 temporary temporary 4096 May  2  2018 temporary
cbello@sta32[~]
17:07:56 $ █
```

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:

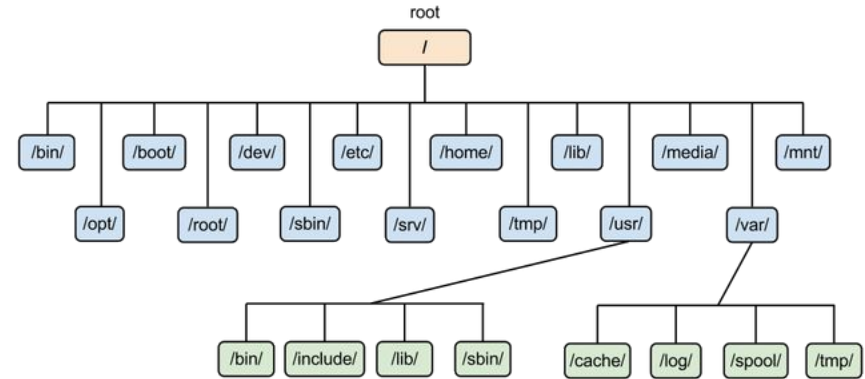
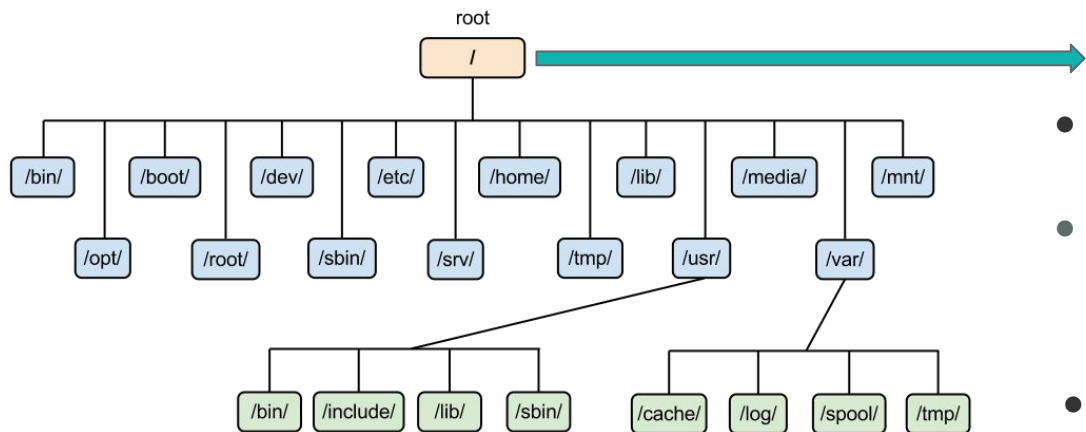


Image from <http://beyondthegEEK.com/2016/08/24/linux-hierarchy-structure/>



# 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**.

# Example

- Absolute paths:

```
user@bash:~$
```

← 1st line: **terminal prompt**

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

← 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:~$
```

← 1st line: **terminal prompt**

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

← 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)

```
vega@vega: ~ 80x24
vega@vega:~$ man
What manual page do you want?
vega@vega:~$ man ls
```

Space: scroll down  
b : scroll up  
q: quit

```
LS(1)                                User Commands                                LS(1)

NAME
    ls - list directory contents

SYNOPSIS
    ls [OPTION]... [FILE]...

DESCRIPTION
    List information about the FILES (the current directory by default).
    Sort entries alphabetically if none of -cftuvSUX nor --sort is speci-
    fied.

    Mandatory arguments to long options are mandatory for short options
    too.

    -a, --all
        do not ignore entries starting with .
```

# File manipulation

---

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

Task	Command
<b>Make or remove</b> directories	<i>mkdir, rmdir</i>
<b>Copy</b> files/directories	<i>cp</i>
<b>Move or rename</b> files/directories	<i>mv</i>
<b>Remove</b> files/directories	<i>rm</i>

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

# More powerful commands

---

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

# Example

```
user@bash:~$ mkdir Intro2Linux
```

```
user@bash:~$ cd Intro2Linux/
```

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

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

```
user@bash:~/Intro2Linux$
```

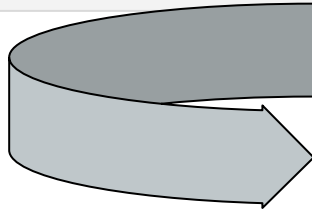
← Line 1: **make** directory “Intro2Linux”

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

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

← Line 4: **show emptyfile.txt** 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`: **G**lobally search a **R**egular **E**xpression and **P**rint

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

<code>^</code>	Start of line
<code>\$</code>	End of line

## Character Classes

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

## Metacharacters (must be escaped)

<code>^</code>	<code>[</code>	<code>]</code>
<code>\$</code>	<code>(</code>	<code>)</code>
<code>.</code>	<code>{</code>	<code>}</code>
<code>*</code>	<code>+</code>	<code>?</code>
<code>\</code>	<code> </code>	<code>-</code>

## GA Filter group accessors

<code>\$Ax</code>	Access group x in field A (e.g. \$A1)
<code>\$Bx</code>	Access group x in field B (e.g. \$B1)

## Quantifiers

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

## Ranges and Groups

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

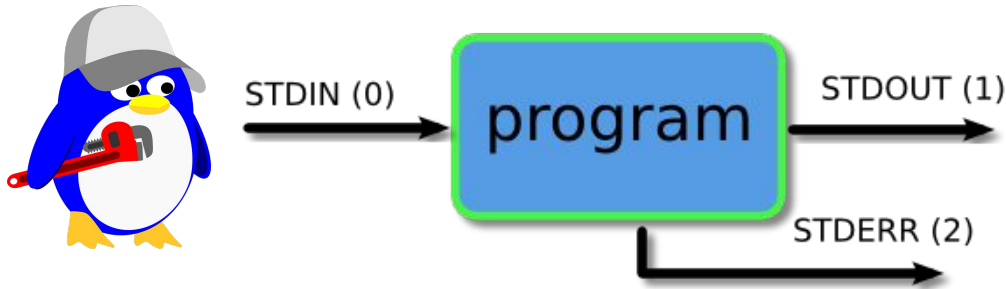
`^brand\s+[^cf]`  
*Key phrases beginning with 'brand' and the second word not starting with c or f*

`\.aspx$`  
*URLs ending in '.aspx'*

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

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

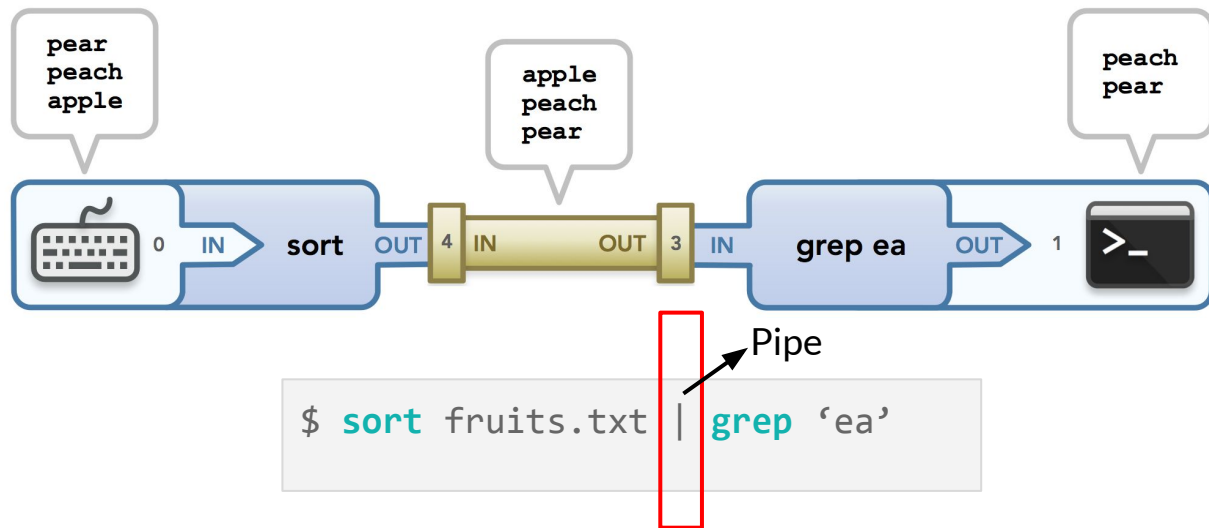
# Piping and redirection are means to connect



- **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)

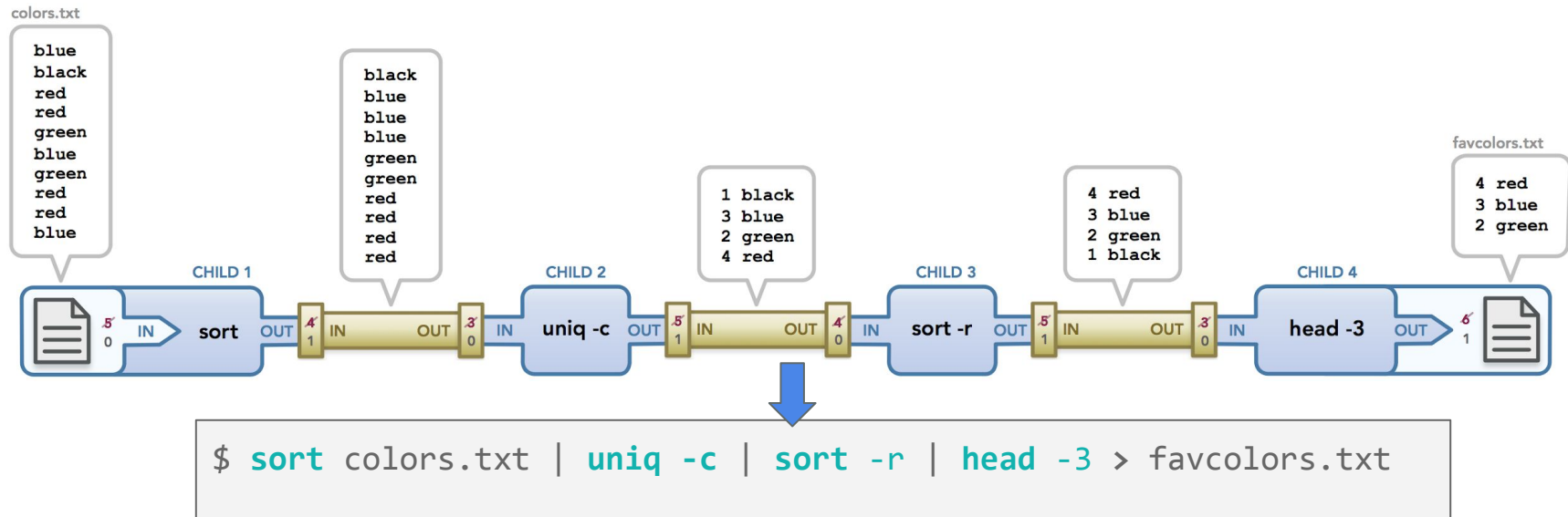
The means by which we may **connect** these **streams between programs** and **files** to **direct data in interesting ways**

# Combining commands with pipes



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

# 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**

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

**3. StackOverflow ([www.stackoverflow.com](http://www.stackoverflow.com))**

**4. The internet (Google, Duckduckgo)**

# Hands on!

Please go to:

carlalbc that is an L

- [https://github.com/carlalbc/BIO609\\_2020/](https://github.com/carlalbc/BIO609_2020/)
- **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



GNU Operating System

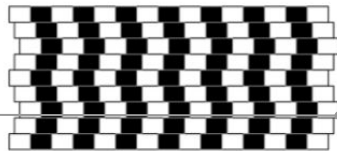
Sponsored by the *Free Software Foundation*

JOIN THE FSF

Free Software Supporter

email address

Sign up



GNUparallel

For people who live life in the parallel lane.

ABOUT GNU PHILOSOPHY LICENSES EDUCATION SOFTWARE DOCUMENTATION HELP 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](https://www.gnu.org/software/parallel/parallel_tutorial.html)