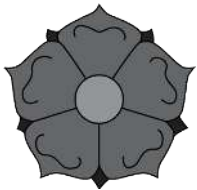


Workshop on Genomics 2024

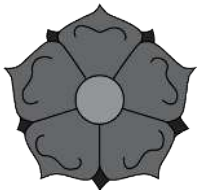


unix



Mercè Montoliu Nerín

January 9th, 2024



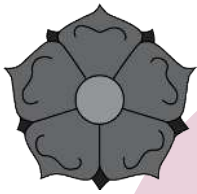
What is UNIX?

Operating system

powerful

multi-user

multitasking



Why is it important for bioinformatics?

Facilitates sharing and reproducing analyses

Handling large datasets and running analyses efficiently

**Efficiency
and
speed**

Access to powerful tools and applications

Using scripts to automate repetitive tasks



The terminal



Make it comfortable to work in

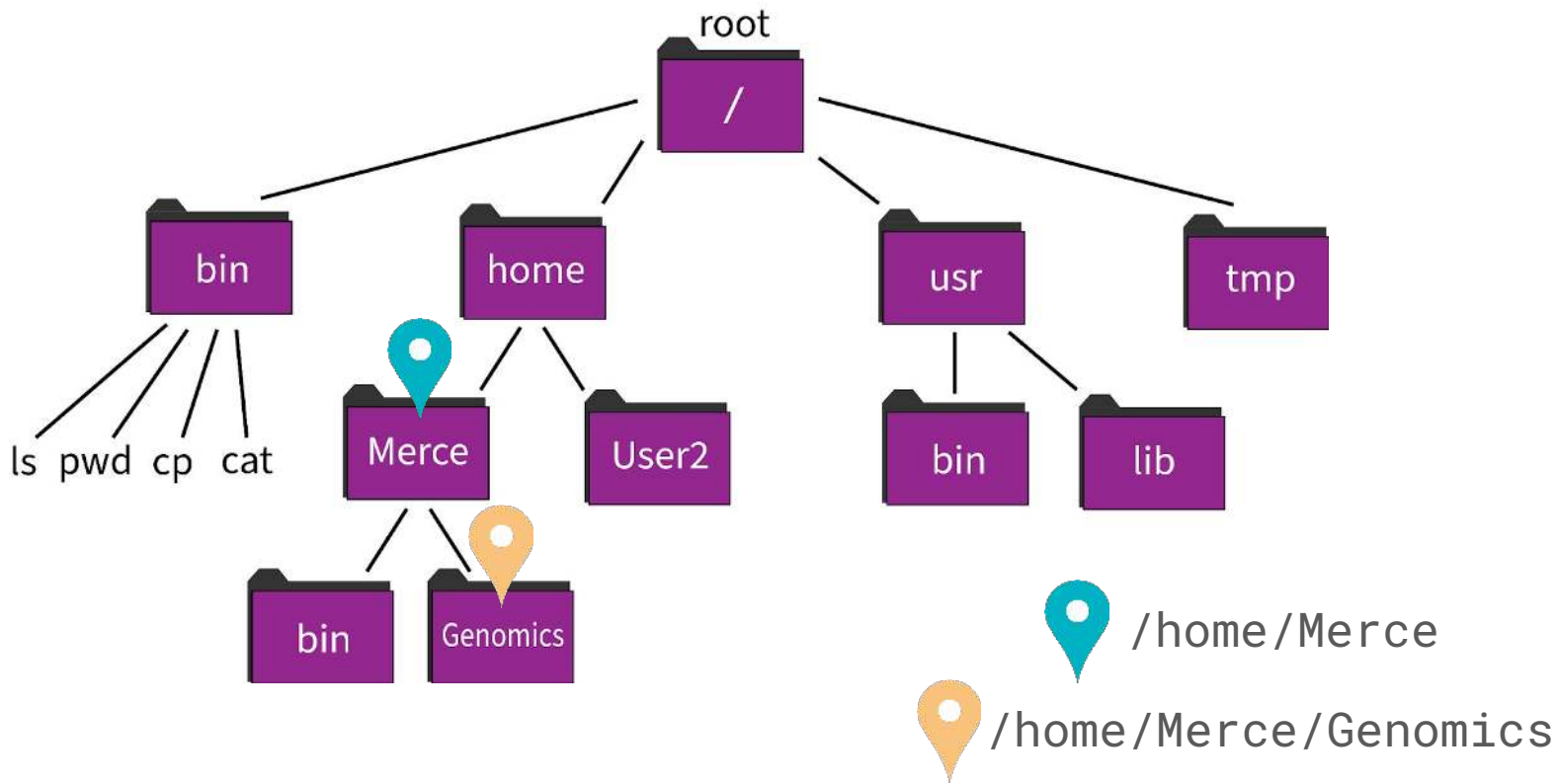
- Resize the window
- Change the font size
- Open multiple terminal windows (or tabs)
- Make sure you have the right combination of colours that work for **you**.



[illegible]

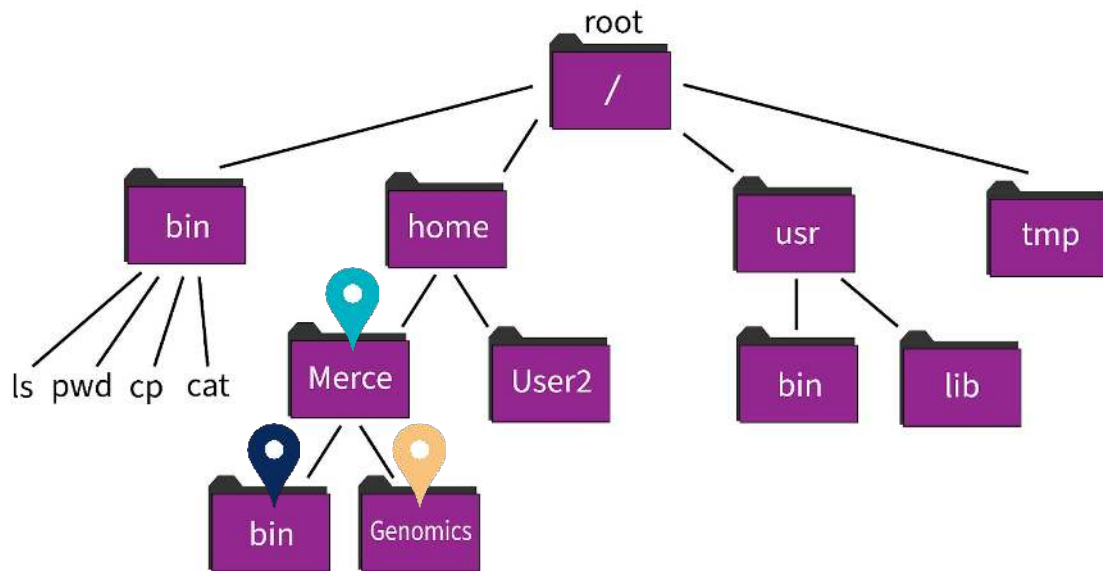


File system organization





Paths - Absolute vs Relative



Absolute paths



/home/Merce



/home/Merce/Genomics



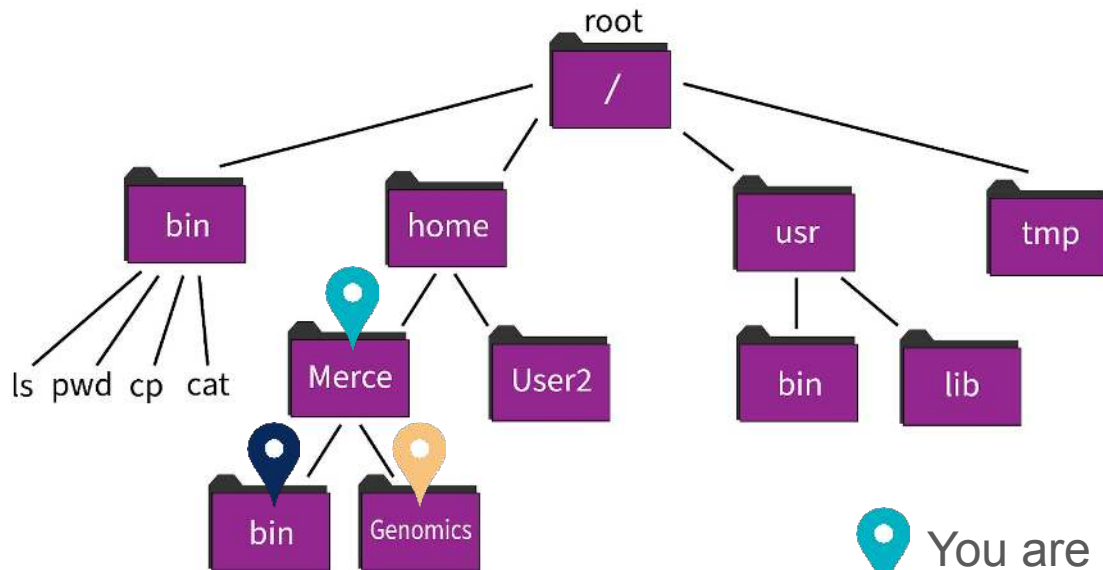
/home/Merce/bin

. refers to our current location

.. refers to the location above us



Paths - Absolute vs Relative



Absolute paths



/home/Merce



/home/Merce/Genomics



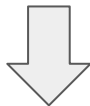
/home/Merce/bin



You are here!

Absolute: /home/Merce

Relative: ./



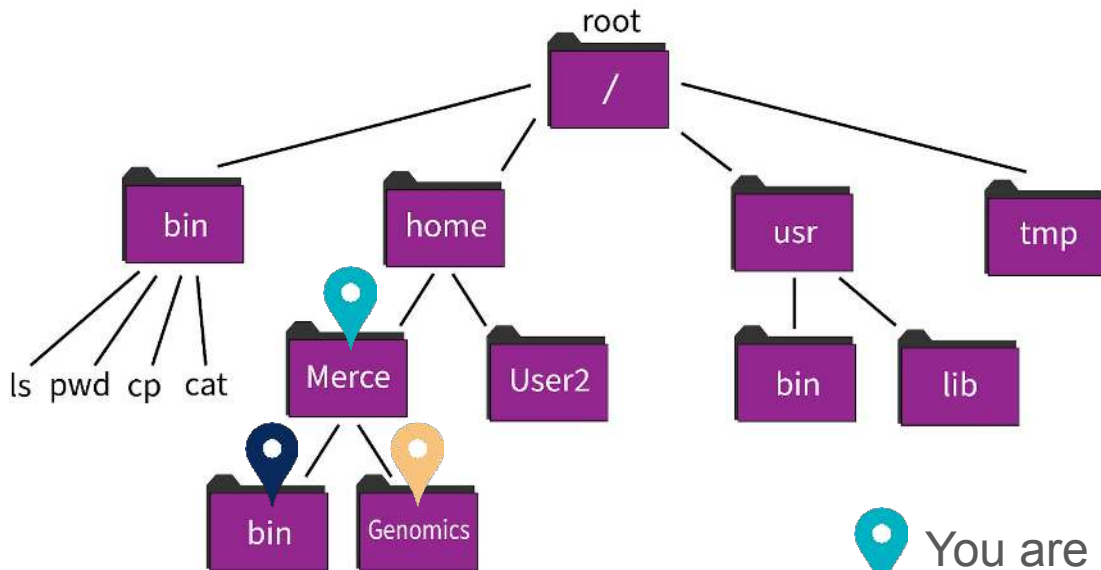
../Genomics

. refers to our current location

.. refers to the location above us



Paths - Absolute vs Relative



Absolute paths



/home/Merice



/home/Merice/Genomics



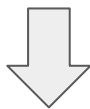
/home/Merice/bin



You are here!

Absolute: /home/Merice

Relative: ./



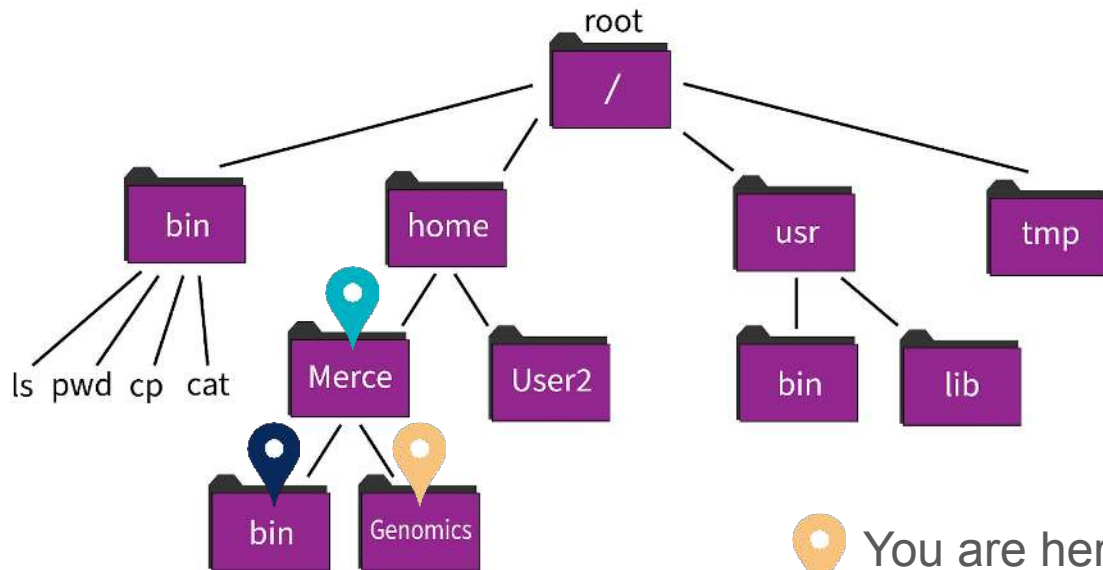
./bin

. refers to our current location

.. refers to the location above us



Paths - Absolute vs Relative



Absolute paths



/home/Merce



/home/Merce/Genomics

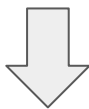


/home/Merce/bin



You are here! Absolute: /home/Merce/Genomics

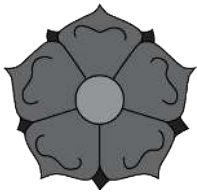
Relative: ./



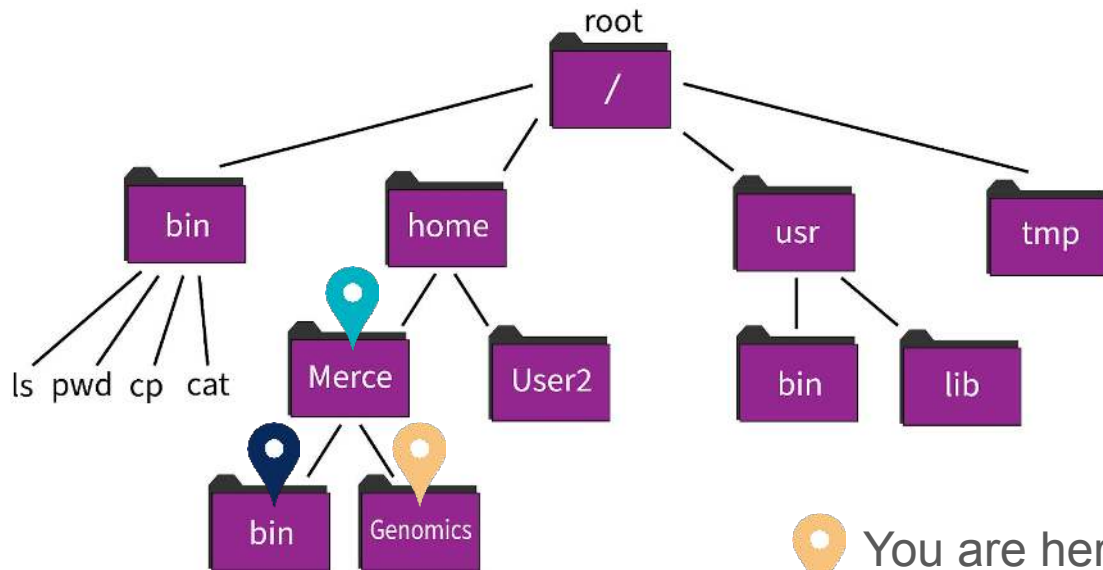
..

. refers to our current location

.. refers to the location above us



Paths - Absolute vs Relative



Absolute paths



/home/Merce



/home/Merce/Genomics

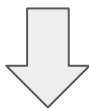


/home/Merce/bin



You are here! Absolute: /home/Merce/Genomics

Relative: ../



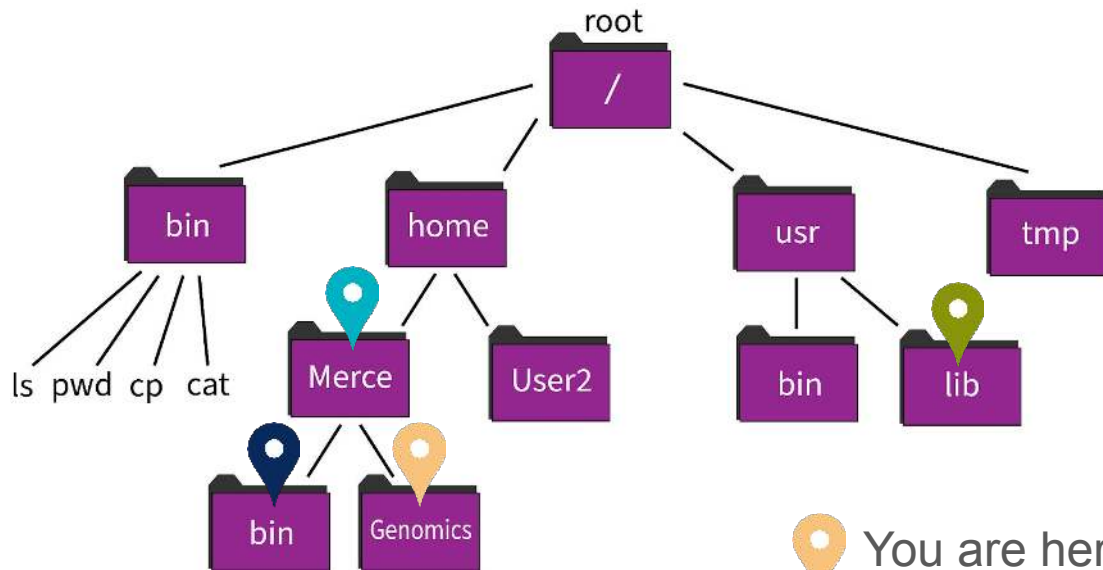
../bin

. refers to our current location

.. refers to the location above us



Paths - Absolute vs Relative



Absolute paths



/home/Merce



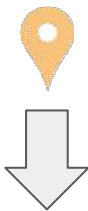
/home/Merce/Genomics



/home/Merce/bin

. refers to our current location

.. refers to the location above us



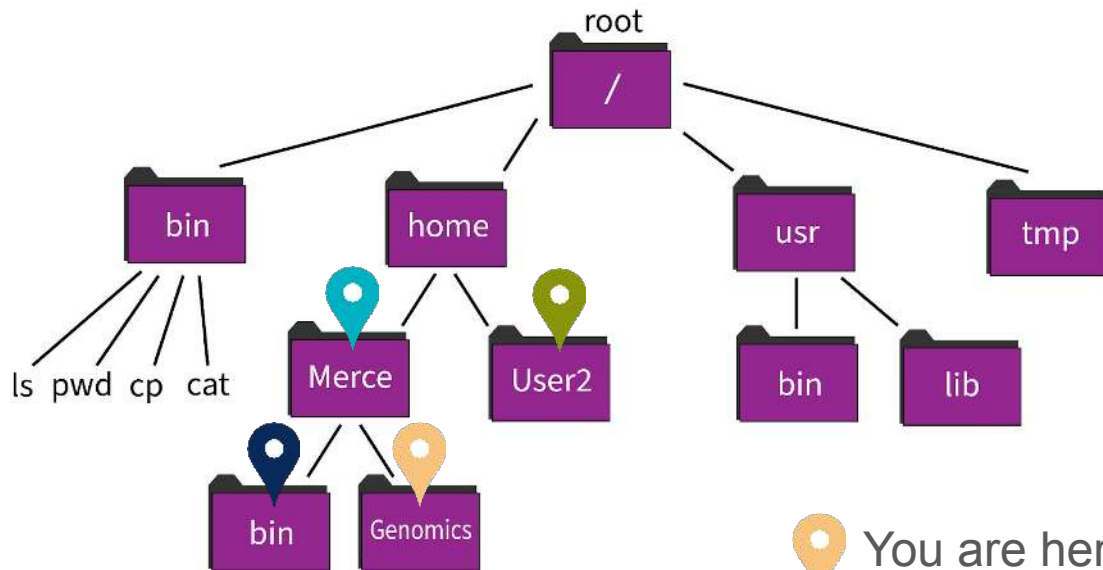
You are here!



How do we refer to this directory?



Paths - Absolute vs Relative



Absolute paths



/home/Merce



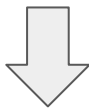
/home/Merce/Genomics



/home/Merce/bin



You are here!



../ ../User2

. refers to our current location

.. refers to the location above us



File system navigation

pwd - where am I?



cd - change directory





File system navigation

pwd - where am I?



/home/Merce

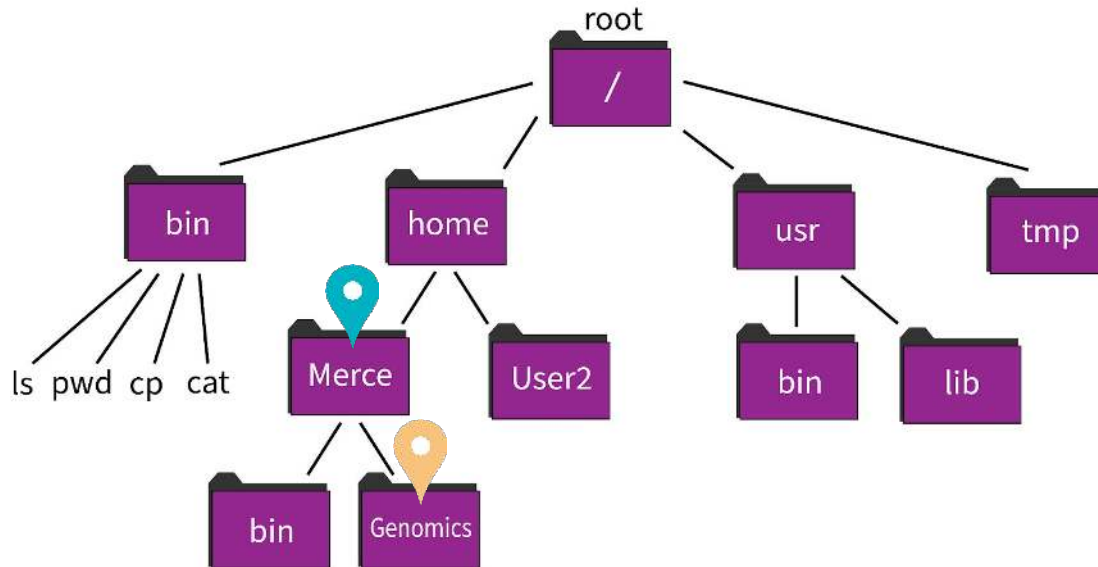


cd - change directory



```
> cd /home/Merce/Genomics
```

```
> cd ./Genomics
```





File system navigation

pwd - where am I?



/home/Merce/Genomics

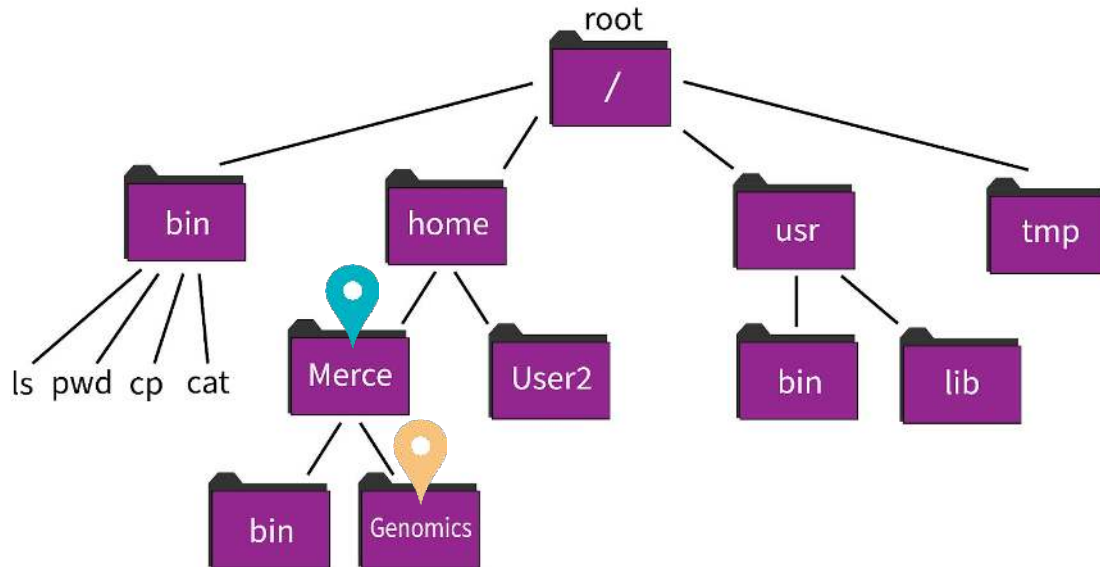


cd - change directory



```
> cd /home/Merce
```

```
> cd ../
```

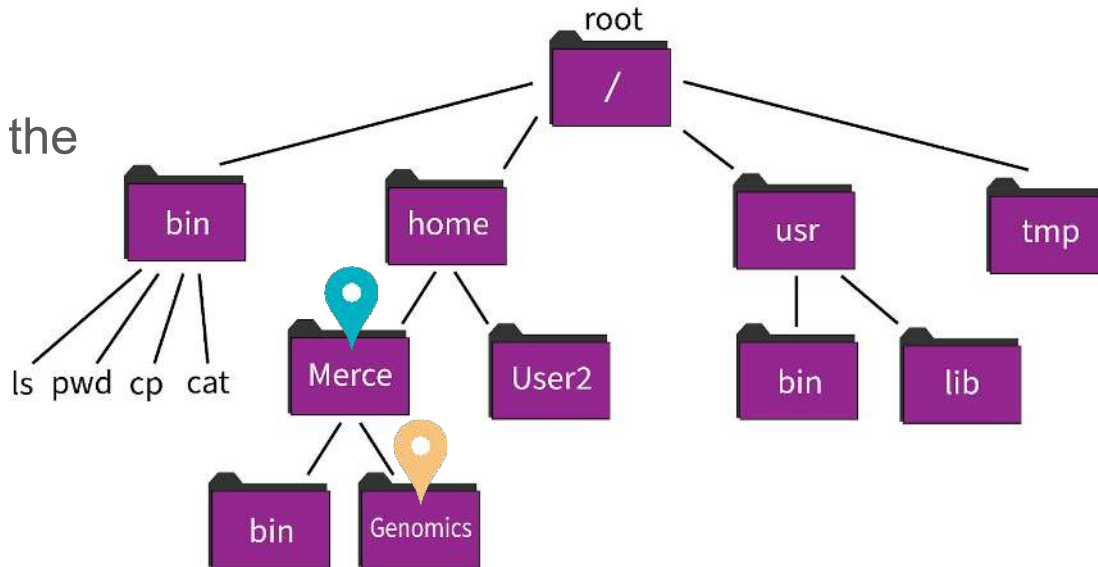




File system visualization

ls - shows you the contents the
directory you are in

```
> ls  
  
> ls .  
  
> ls ./
```



```
> ls ../
```



Key shortcuts

Ctrl + C	halts current command
Ctrl + Shift + C	copy (linux) - Cmd + C (mac)
Ctrl + Shift + V	paste (linux) - Cmd + V (mac)
Ctrl + W	erases one word in current line
Ctrl + U	erases whole line
Ctrl + A	go to begining of line
Ctrl + E	go to end of line
Type <code>exit</code>	log out of current session



Create, copy, move, and remove files and folders

mkdir - create new directory

cp - copy file

mv - move file or directory

rm - remove file

"Unix was not designed to stop its users from doing stupid things, as that would also stop them from doing clever things." - Doug Gwyn



Symbolic links

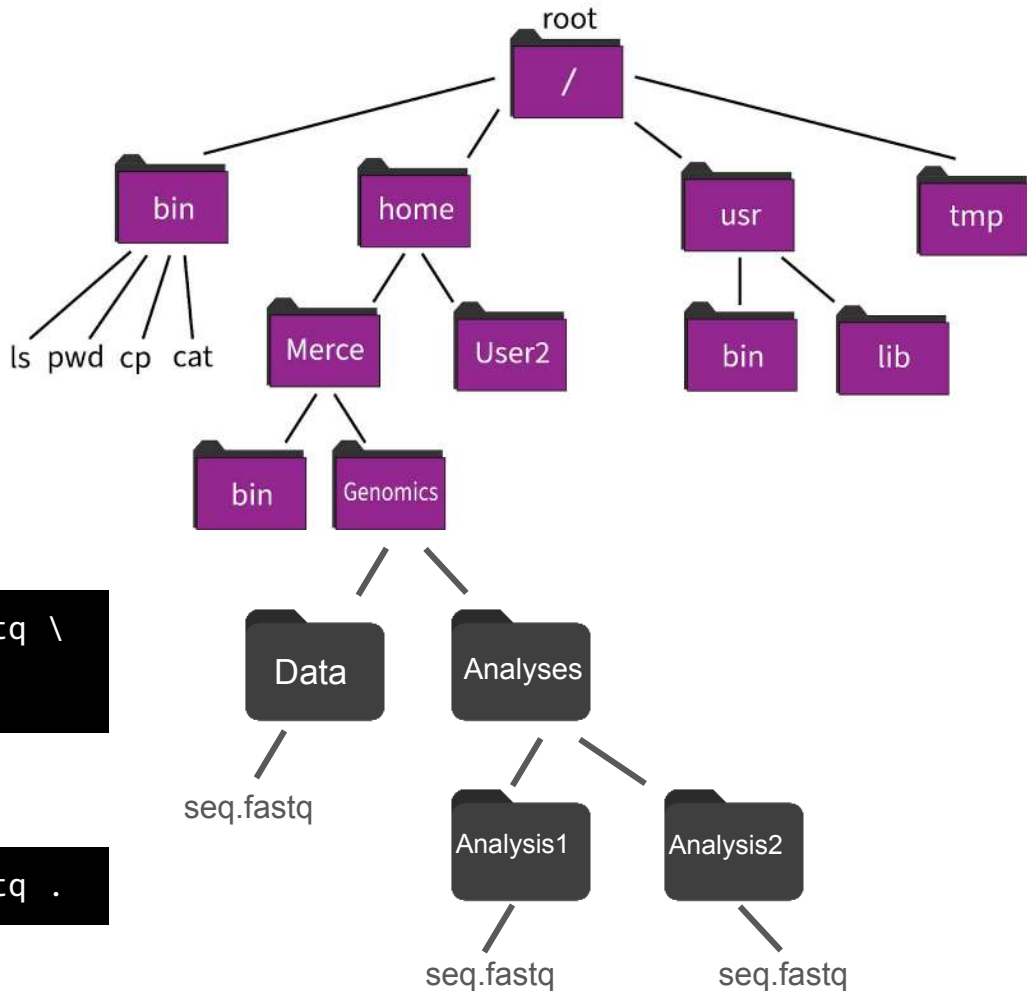
In **-s** /path/to/file link

create a symlink of file

```
> ln -s /home/Merce/Genomics/Data/seq.fastq \  
/home/Merce/Genomics/Analyses/Analysis1/
```

If we are already inside the folder Analysis1:

```
> ln -s /home/Merce/Genomics/Data/seq.fastq .
```

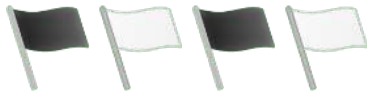




Manual

man *command* - manual of the command

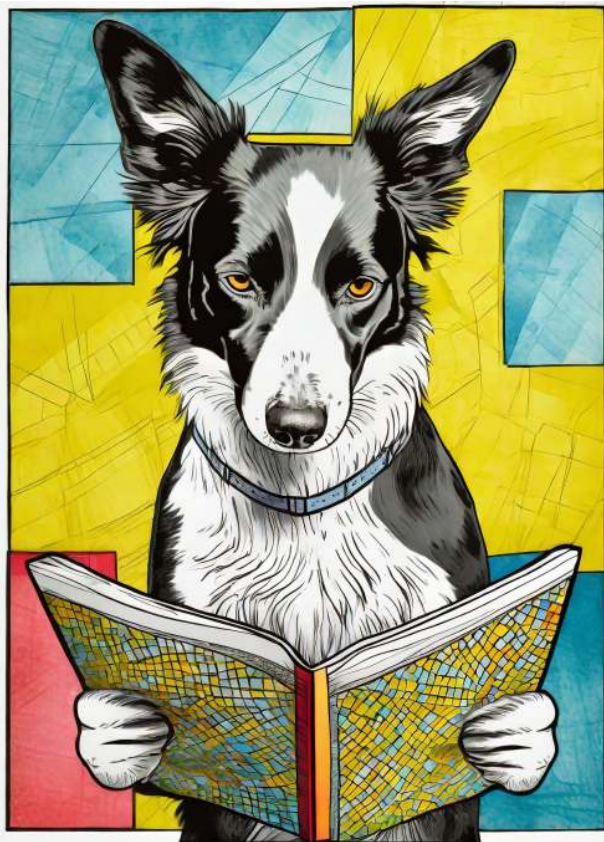
```
> man ls
```



ls -l formatted list

ls -h “human” formatted list

ls -lh combination of flags



Inputs and outputs

stdin It stands for standard input, and is used for taking text as an input.

stdout It stands for standard output, and is used to text output of any command you type in the terminal, and then that output is stored in the stdout stream.

stderr It stands for standard error. It is invoked whenever a command faces an error, then that error message gets stored in this data stream.



stdin, stdout, stderr

command *stdin*

if it works: prints in our terminal the *stdout*

if it fails: prints in our terminal the *stderr*

command *stdin* > *stdout*

if it works: *stdout* is redirected to a file

if it fails: prints in our terminal the *stderr*

command2 *stdin2* > *stdout*

stdout is redirected to a file and rewrites its contents

command2 *stdin2* >> *stdout*

stdout is redirected to a file and appended after its contents



stdin, stdout, stderr

command *stdin*

if it works: prints in our terminal the *stdout*

if it fails: prints in our terminal the *stderr*

command file1 > output.txt

stdin *stdout*

if it works: *stdout* is redirected to a file

if it fails: prints in our terminal the *stderr*

command2 file2 > output.txt

stdin *stdout*

stdout is redirected to a file and rewrites its contents

command2 file2 >> output.txt

stdin *stdout*

stdout is redirected to a file and appended after its contents



stdin, stdout, stderr

command file1 2> errors.txt
stdin *stderr*

if it works: prints in our terminal the *stdout*
if it fails: *stderr* is redirected to a file

command file1 &> output.txt
stdin *stdout&stderr*

redirects both *stdout* and *stderr* to a file

command file1 > output.txt 2> errors.txt
stdin *stdout* *stderr*

redirects both *stdout* and *stderr* to a
separate file each.



Explore file content

wc - word count (-l lines, -c characters, -w words)

less - visualize file contents in your terminal screen (press q to exit)

cat - prints contents of your file as *standard output* in your terminal

head - visualize the first 10 lines of a file

tail - visualize the last 10 lines of a file

The character **|** (**pipe**) is used to concatenate commands, so that we can run one command after the other, avoiding the creation of intermediate files.

```
command1 input | command2 > output
```

Instead of :

```
command1 input > output1  
command2 output1 > output2
```

Using pipe, the output of running *command1* on a given input gets directly piped into *command2*, and we obtain an output of these two consecutive commands, generating only one output.



A bit more advanced file-handling commands

cat - prints contents of your file as *standard output* in your terminal

redirect to a command

```
cat fileA | command > output.txt
```

concatenate files

```
cat fileA fileB >> fileC
```

```
cat fileA > fileC
```

```
cat fileB >> fileC
```



A bit more advanced file-handling commands

sort - puts in certain order a series of lines in our file

`sort -r fileA` sorts in reverse order

`sort -n fileA` sorts lines in fileA numerically

`sort -k 2 fileA` sort fileA by column 2

`sort -k 2nr fileA` sort fileA by column 2, numerically and in reverse order

`sort -V fileA` sort lines in fileA numerically natural.

`sort -u fileA` sort lines and removes duplicates -> `sort fileA | uniq`



A bit more advanced file-handling commands

Are these two files different?

diff - can tell us if there are differences between two files

```
diff -q fileA fileB
```

“Files fileA and fileC differ”

```
diff fileA fileB
```

prints differences





A bit more advanced file-handling commands

Splitting a file

split - split a given file into multiple files (default 1000)

```
split -l 20 fileA
```

produce x number of files from fileA, each containing 20 lines.

cut - extract specific parts of a file

```
cut -c 2 fileA
```

extract specific columns from a file



Text editors

Nano - The simpler option of text editor. All commands within the nano text editor are given by pressing the Control-key, usually represented as ^

- ^S save current file
- ^O save to (a different file)
- ^X exit from nano



Text editors

Vim - a highly configurable text editor built to make creating and changing any kind of text very efficient

- i start insert mode (you can start typing after where your cursor is)
- ESC exits insert mode (also Ctrl + C)
- :w save file without exiting
- :q exit file (if there are unsaved changes, it fails)
- :wq save and exit
- :q! exit without saving changes



Text editors

emacs - a text editor characterized by its extensibility and configurability. Some essential commands get activated by typing Control + X, then the command (while holding the control key), but there is a wide range of key combinations to be used to move and edit the text

Ctrl + x + s save file

Ctrl + x + c exit editor (if not saved, it ask if you want to save, then type "yes")

VIM

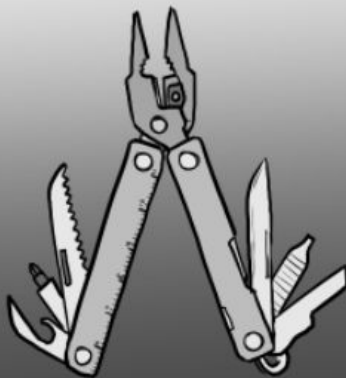
usable in just about
any environment.

does one thing, well.



EMACS

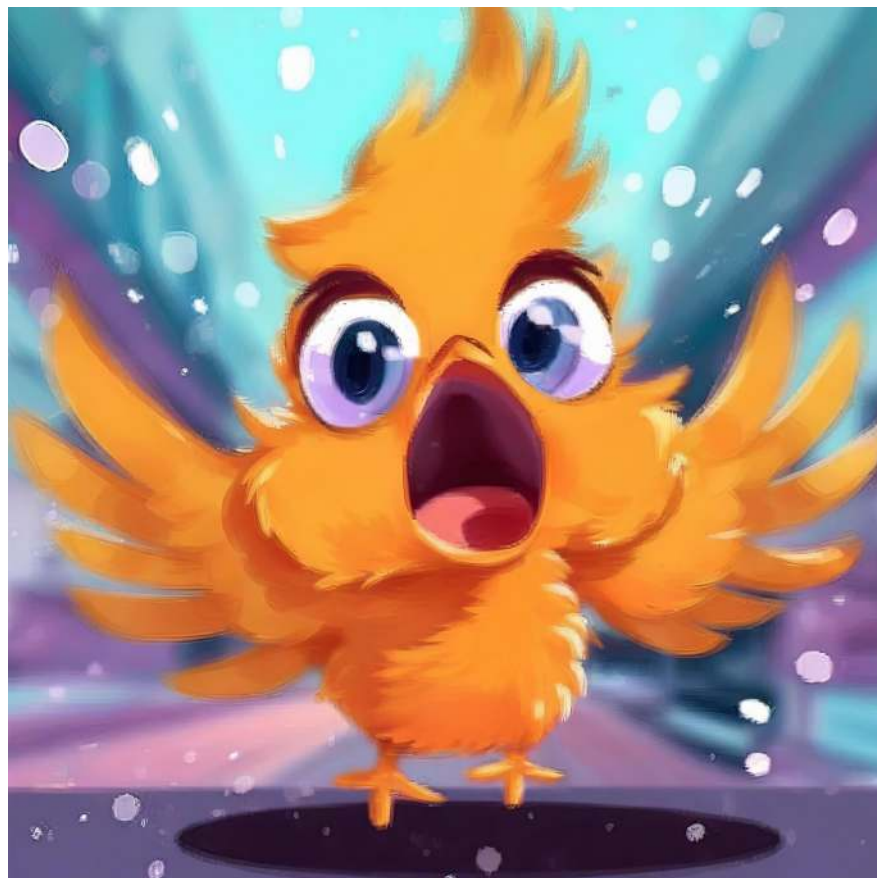
flexible, customizable, and
packed with every feature
known to man.



NANO

mostly used by people
who do not know
what they are doing;
or psychopaths.







What my family and friends
think I do



What my supervisor thinks I do



What I actually do

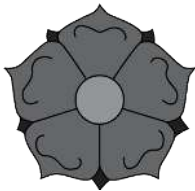
Google



ChatGPT



stackoverflow



Cheat-sheet

Workshop on Genomics 2024

unix cheatsheet

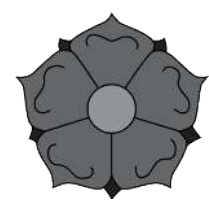
<code>pwd</code>	show current path / directory
<code>ls</code>	list directory
<code>cd <i>dir</i></code>	change to directory <i>dir</i>
<code>cd</code>	change to home
<code>cd ~</code>	change to home
<code>cd -</code>	change to previous working directory
<code>.</code>	current directory
<code>..</code>	parent directory
<code>mkdir <i>dir</i></code>	create directory <i>dir</i>
<code>cp <i>file1 file2</i></code>	copy <i>file1</i> to <i>file2</i>
<code>mv <i>file1 file2</i></code>	move <i>file1</i> to <i>file2</i> or rename <i>file1</i> to <i>file2</i>
<code>rm <i>file1</i></code>	delete <i>file1</i>
<code>ln -s <i>file link</i></code>	create symbolic link

<code>wc</code>	count (-l lines, -w words, -c characters)
<code>tail <i>file</i></code>	output last 10 lines of file
<code>head <i>file</i></code>	output first 10 lines of file
<code>less <i>file</i> / more <i>file</i></code>	visualize contents of file
<code>cat <i>file</i></code>	output file to standard output
<code>sort</code>	sort rows
<code>diff <i>fileA fileB</i></code>	differences?
<code>uniq</code>	keep unique rows
<code>cut -c 2</code>	cut column 2

<code>man command</code>	manual for command
<code>chmod +x <i>file</i></code>	makes <i>file</i> executable
<code>command1 command2</code>	runs <i>command2</i> on output of <i>command1</i>
<code>wget <i>web-address-to-file</i></code>	download file into current dir
<code>ssh <i>user@server</i></code>	connect to cluster
<code>tar -xzf <i>archive.tar.gz</i></code>	uncompress <i>tar.gz</i>
<code>tar -czf <i>archive.tar.gz archive</i></code>	compress <i>archive</i> to <i>.tar.gz</i>
<code>gzip</code>	compress <i>.gz</i>
<code>gunzip</code>	uncompress <i>.gz</i>

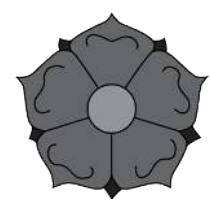
<code>Ctrl + C</code>	halts current command
<code>Tab</code>	autocomplete current line
<code>Arrow up</code>	previous commands
<code>Ctrl + Shift + C</code>	copy (linux)
<code>Cmd + C</code>	(mac)
<code>Ctrl + Shift + V</code>	paste (linux)
<code>Cmd + V</code>	(mac)
<code>Ctrl + W</code>	erases one word in current line
<code>Ctrl + U</code>	erases whole line
<code>Ctrl + A</code>	go to beginning of line
<code>Ctrl + E</code>	go to end of line
<code>exit</code>	log out of current session

#evomics2024



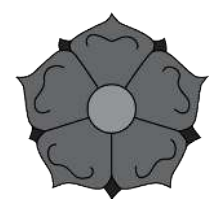
Cheat-sheet





Rubber duck





Now it is your turn to practice!
And become each other's ducks!

