



University of Antwerp  
| Human Molecular Genetics  
| Centre of Medical Genetics

# Linux for life scientists

Introduction to Linux CLI

Eline Turcksin

biomina

# Course overview: (tentative)

1. **What is Linux?**
2. **Exploring the command line**
3. **Navigating the file system**
4. **Manipulating files and directories**
5. **File contents**
6. **Input, output and pipelines**
7. **Installing software & error handling**
8. **Bash scripting**

# 0. TO DO beforehand

# TO DO

- **Install WSL (Windows users)**
- **Install homebrew (MacOS users)**
- **Install VSCode**
- **Request VSC account**
- **Check out terminal environment => will do together**
  - <https://sandbox.bio/> to practice commands during course

# Download course material

- <https://tinyurl.com/linuxcourse2024>
- Click 'Download'

## Download

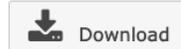
Preferred language English

Select which file(s) you wish to download. You can right click on the download button and "Copy Link Location" to download the file using another tool.

From : pholmgre@ua.ac.be  
Created : 25 Nov 2024  
Expires : 25 Dec 2024  
Size : 1.1 GB  
Subject : data cursus

data_files.tar.gz	1.1 GB
-------------------	--------

Direct Link: [https://filesender.be/net.be/download.php?token=2cc8d3bb-afce-43bb-8038-c6ec8271d6c6&file\\_ids=3157729](https://filesender.be/net.be/download.php?token=2cc8d3bb-afce-43bb-8038-c6ec8271d6c6&file_ids=3157729)



# 1. Introduction

# Some Linux (mis)conceptions

- **It's difficult to install and work with**
- **It's totally different from Windows**
- **For everything you want to do, you need to write commands on a 'black screen'**
- **To work with Linux, you need to know/memorize so much commands, paths, settings**
- **These Linux/programming guys just seem like wizards**

# Some Linux (mis)conceptions

- It's difficult to install and work with  
**No**

# Some Linux (mis)conceptions

- It's difficult to install and work with  
**No**
- It's totally different from Windows
  - Graphical part: **No**
  - Command line interface: **Yes**

# Some Linux (mis)conceptions

- **It's difficult to install and work with**  
**No**
- **It's totally different from Windows**
  - Graphical part: **No**
  - Command line interface: **Yes**
- **For everything you want to do, you need to write commands on a 'black screen'**
  - Not necessarily
  - Depends on what you want/need to do and how advanced you need to go
  - For data handling, this is usually true.

# Some Linux (mis)conceptions

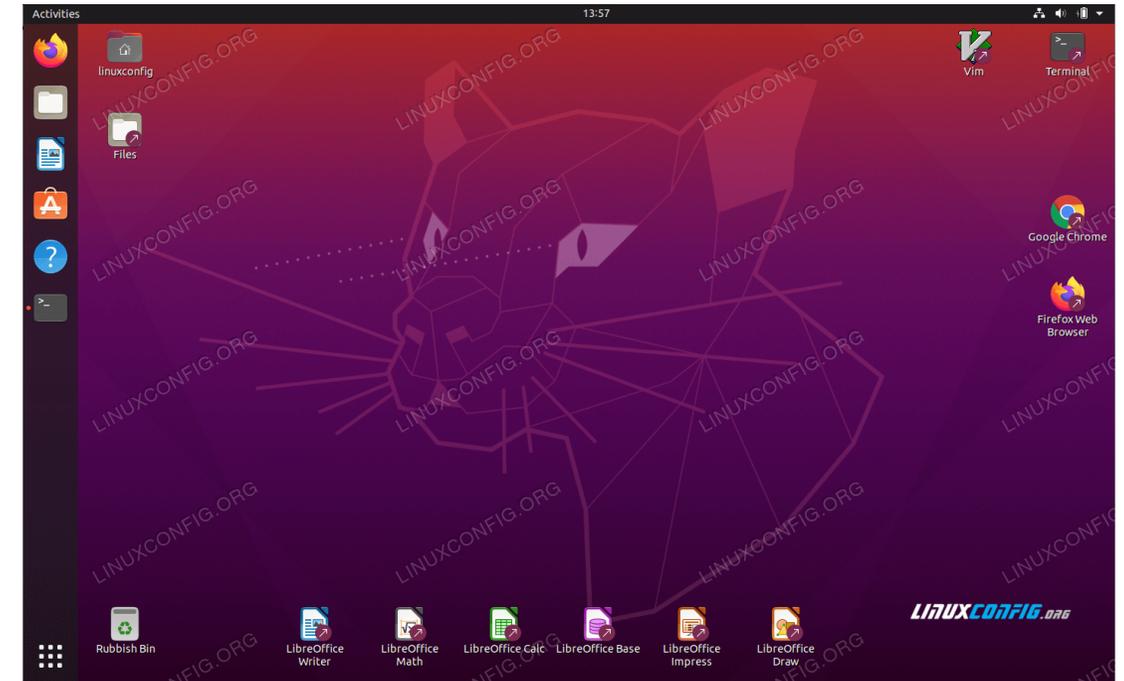
- **It's difficult to install and work with**  
**No**
- **It's totally different from Windows**
  - Graphical part: **No**
  - Command line interface: **Yes**
- **For everything you want to do, you need to write commands on a 'black screen'**
  - Not necessarily
  - Depends on what you want/need to do and how advanced you need to go
  - For data handling, this is usually true.
- **To work with Linux, you need to know/memorize so much commands, paths, settings**
  - **No**
  - Need to learn where to efficiently find all the information (memorizing come gradually)

# Some Linux (mis)conceptions

- **It's difficult to install and work with**  
**No**
- **It's totally different from Windows**
  - Graphical part: **No**
  - Command line interface: **Yes**
- **For everything you want to do, you need to write commands on a 'black screen'**
  - Not necessarily
  - Depends on what you want/need to do and how advanced you need to go
  - For data handling, this is usually true.
- **To work with Linux, you need to know/memorize so much commands, paths, settings**
  - **No**
  - Need to learn where to efficiently find all the information (memorizing come gradually)
- **These Linux/programming guys just seem like wizards**  
**Totally true 😊**

# What is Linux?

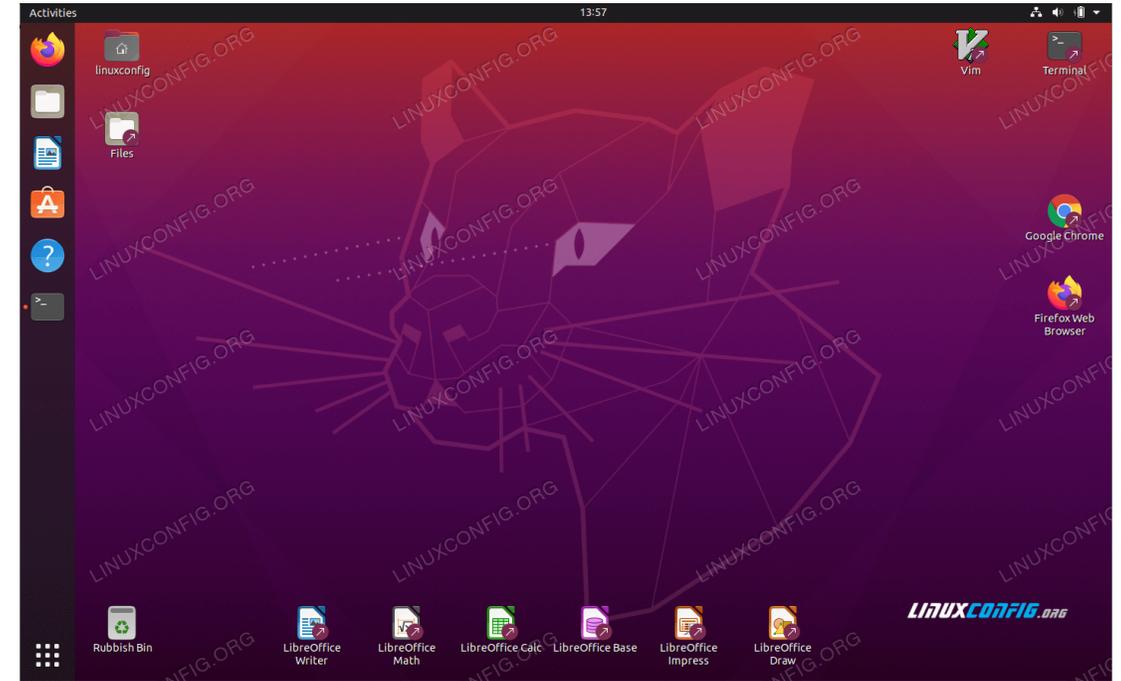
- **Operating system (OS)**
  - An alternative to Windows (macOS)
- **Graphic User Interface (GUI)**
  - Similar in use as Windows
- **Command-Line Interface (CLI)**
  - Terminal
  - “Under the hood”
- **Availability**
  - Many Linux distributions (=version)
  - Usually free (not always e.g., RedHat)



```
mark@linux-desktop: /tmp/tutorial
File Edit View Search Terminal Help
mark@linux-desktop:~$ mkdir /tmp/tutorial
mark@linux-desktop:~$ cd /tmp/tutorial
mark@linux-desktop:/tmp/tutorial$ mkdir dir1 dir2 dir3
mark@linux-desktop:/tmp/tutorial$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
mark@linux-desktop:/tmp/tutorial$ cd /etc ~/Desktop
bash: cd: too many arguments
mark@linux-desktop:/tmp/tutorial$ ls
dir1 dir2 dir3
mark@linux-desktop:/tmp/tutorial$
```

# What is Linux?

- **Operating system (OS)**
  - An alternative to Windows (macOS)
- **Graphic User Interface (GUI)**
  - Similar in use as Windows
- **Command-Line Interface (CLI)**
  - Terminal
  - “Under the hood”
- **Availability**
  - Many Linux distributions (=version)
  - Usually free (not always e.g., RedHat)



```
mark@linux-desktop: /tmp/tutorial
File Edit View Search Terminal Help
mark@linux-desktop:~$ mkdir /tmp/tutorial
mark@linux-desktop:~$ cd /tmp/tutorial
mark@linux-desktop:/tmp/tutorial$ mkdir dir1 dir2 dir3
mark@linux-desktop:/tmp/tutorial$ mkdir
mkdir: missing operand
Try 'mkdir --help' for more information.
mark@linux-desktop:/tmp/tutorial$ cd /etc ~/Desktop
bash: cd: too many arguments
mark@linux-desktop:/tmp/tutorial$ ls
dir1 dir2 dir3
mark@linux-desktop:/tmp/tutorial$
```



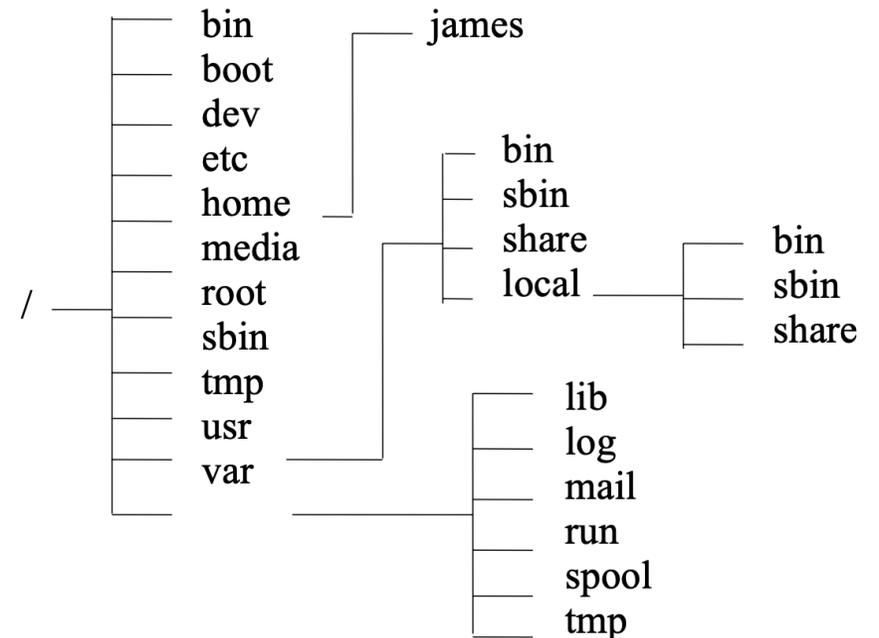
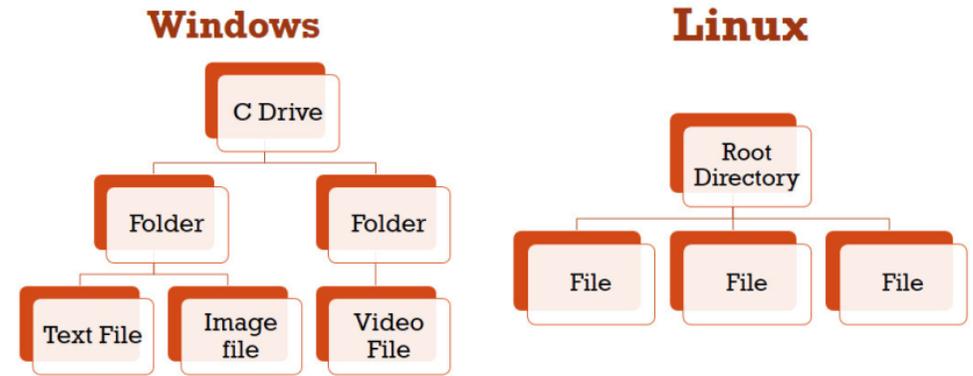
# What is the shell?

- **A program**
- **Command line interpreter: waits for input and performs tasks**
- **Input language is a scripting language**
- **Shell provides access to 100s of commands**
- **Different shell programs exist**
  - Linux systems: bash (Bourne Again Shell)
  - On MacOS: default is zsh, but making use of homebrew, also all Linux commands are available (though bash should be installed by default)

# Linux Filesystem

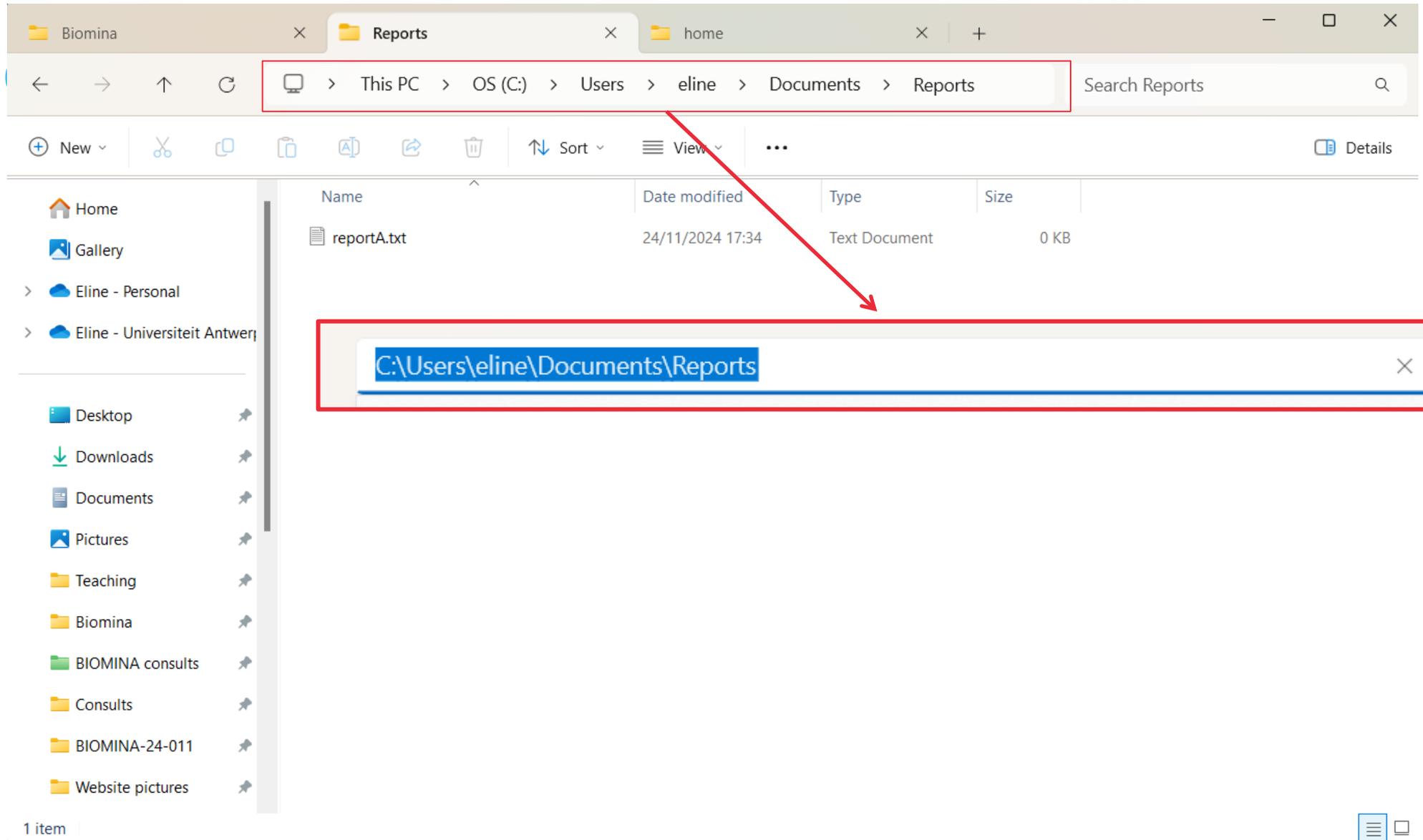
## File System Tree

- 1 big directory structure
  - No separate disks (Windows C:\ , D:\ )
- Directory separators (path)
  - Linux = forward slash /
  - Windows = backwards slash \



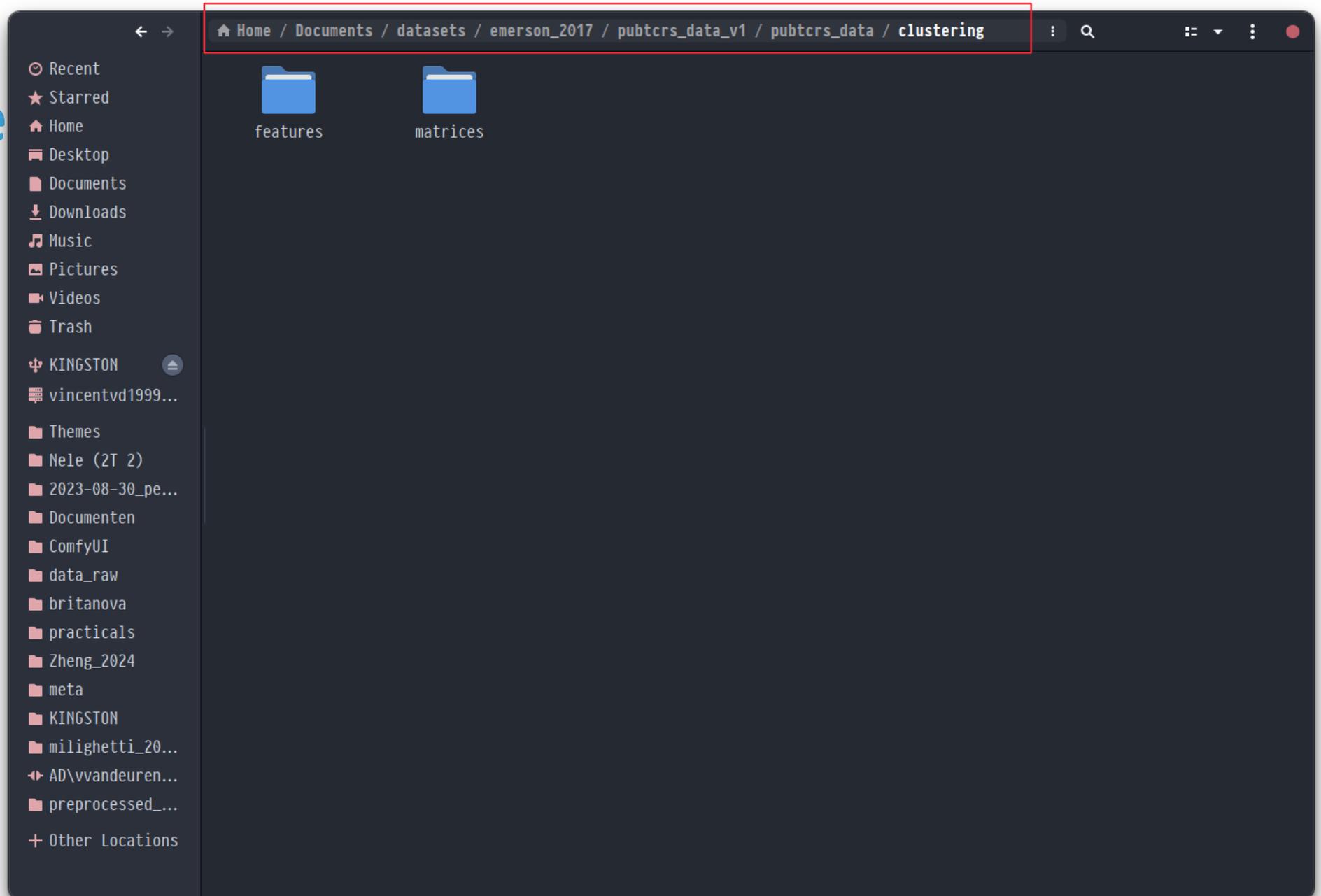
# Linux File

- Paths
- Windows (GUI)



# Linux File

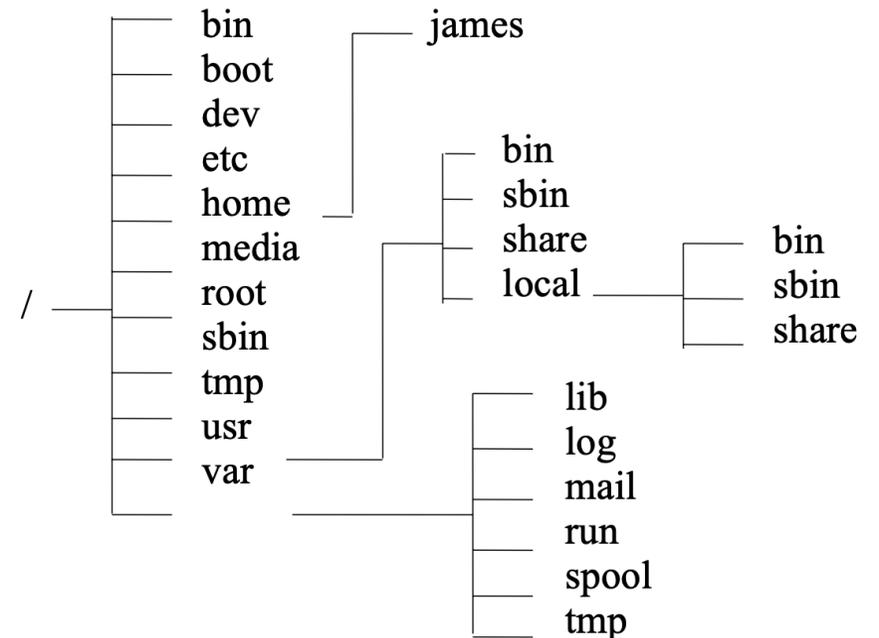
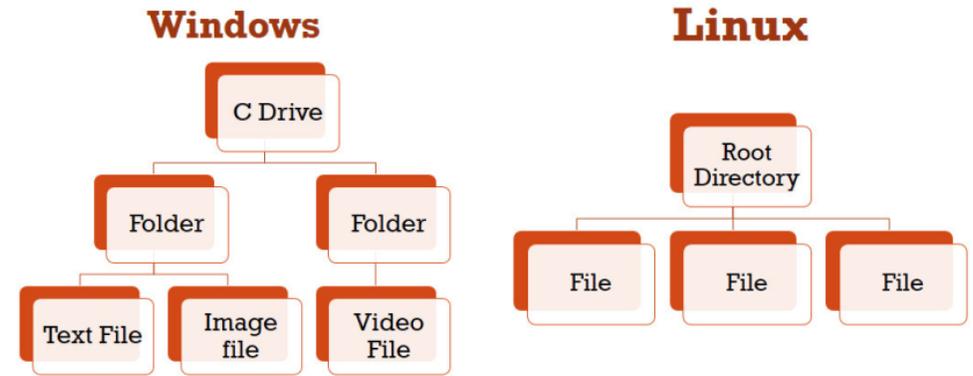
- Paths
- Linux (GUI)



# Linux Filesystem

## File System Tree

- 1 big directory structure
  - No separate disks (Windows C:\ , D:\ )
- Directory separators (path)
  - Linux = forward slash /
  - Windows = backwards slash \
- Top level = root directory, or just /
- Subdirectories
  - /home/james
- Different folders = different purpose
  - /bin : holds programs (=binaries)
  - /tmp: to store temporary files
  - /home: user-specific folders



# Documentation

- **Essential when using CLI (or when programming)!**

- Learning to get the information you need is a major part of working in Linux/programming



- Sources:

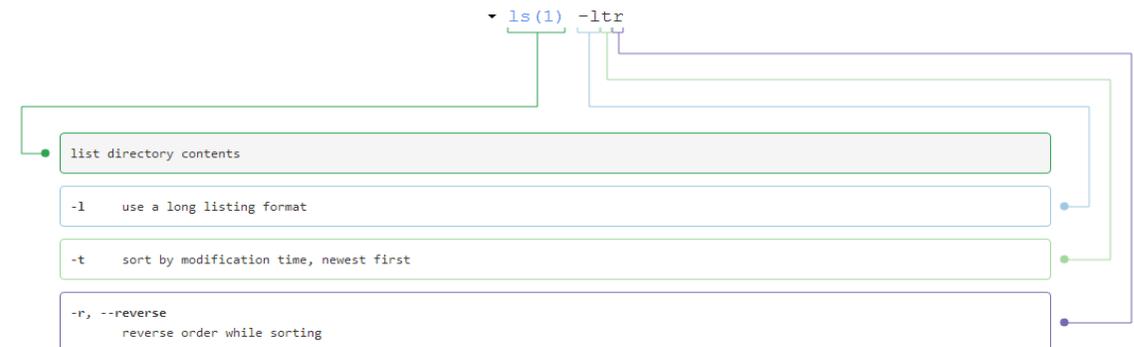
- Google (duh 😊 )

- Useful websites like StackOverflow (SO) (forums for variety of topics, including Linux commands)

- But Google as a starting point is usually way to go

- [explainshell.com](https://explainshell.com)

- Dissects bash command and explains every option



# Documentation

## ▪ Sources

- All (or almost all) commands have a 'help' option (either -h or --help):

- `$ mkdir --help`
  - Summary of most useful options
  - Usually not exhaustive
- Syntax
  - Optional parts are in between [ ]
  - ... to indicate it can be more than one

```
(exporter) pholmgren@ngsvm-pipelines:~/wes/aws_utils$ mkdir --help
Usage: mkdir [OPTION]... DIRECTORY...
Create the DIRECTORY(ies), if they do not already exist.

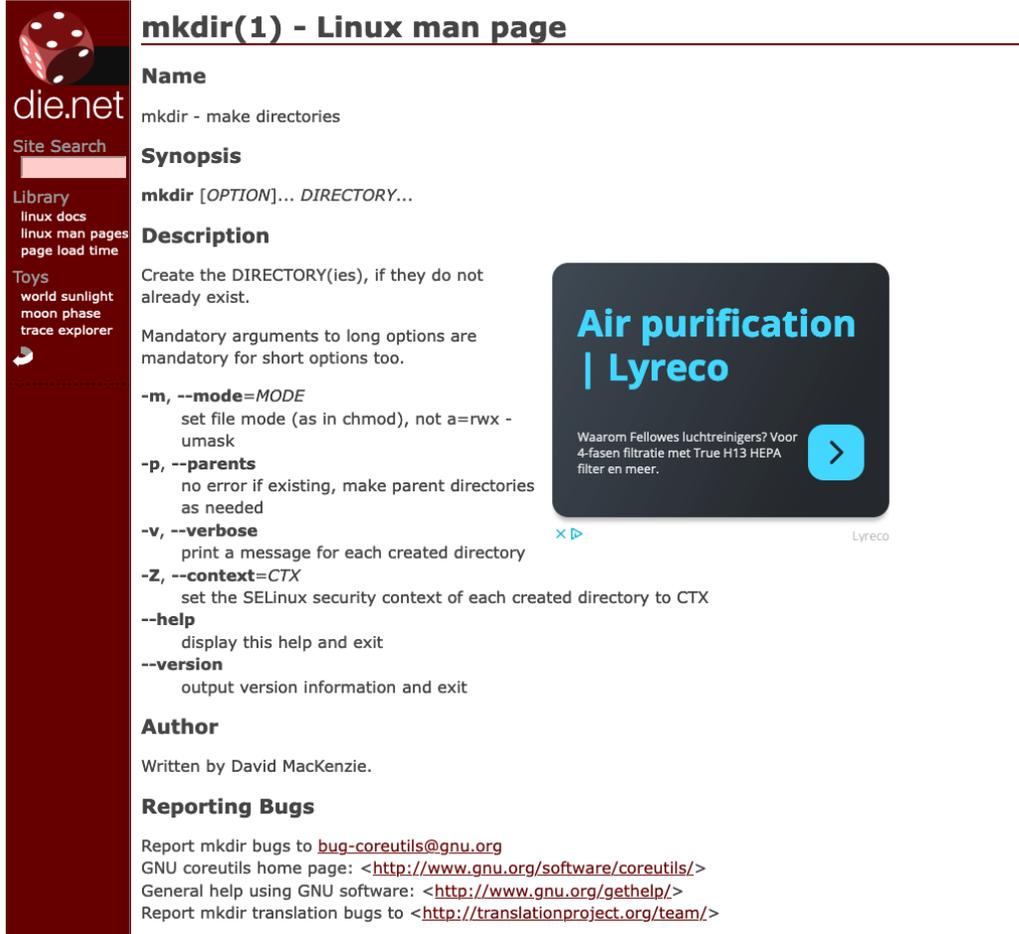
Mandatory arguments to long options are mandatory for short options too.
-m, --mode=MODE    set file mode (as in chmod), not a=rwx - umask
-p, --parents      no error if existing, make parent directories as needed
-v, --verbose      print a message for each created directory
-Z                set SELinux security context of each created directory
                  to the default type
--context[=CTX]   like -Z, or if CTX is specified then set the SELinux
                  or SMACK security context to CTX
--help            display this help and exit
--version        output version information and exit

GNU coreutils online help: <http://www.gnu.org/software/coreutils/>
Full documentation at: <http://www.gnu.org/software/coreutils/mkdir>
or available locally via: info '(coreutils) mkdir invocation'
(exporter) pholmgren@ngsvm-pipelines:~/wes/aws_utils$
```

# Documentation

## ■ Sources

- Man pages
  - The full manual of a command
- Easiest with Google
  - Example: man <command>
  - <https://linux.die.net/man>
- CLI
  - \$ man <COMMAND>
  - Opens a reader
    - Scrollable: up/down arrows
    - To quit: press 'q'



**mkdir(1) - Linux man page**

**Name**  
mkdir - make directories

**Synopsis**  
mkdir [OPTION]... DIRECTORY...

**Description**  
Create the DIRECTORY(ies), if they do not already exist.  
Mandatory arguments to long options are mandatory for short options too.

**-m, --mode=MODE**  
set file mode (as in chmod), not a=rwx - umask

**-p, --parents**  
no error if existing, make parent directories as needed

**-v, --verbose**  
print a message for each created directory

**-Z, --context=CTX**  
set the SELinux security context of each created directory to CTX

**--help**  
display this help and exit

**--version**  
output version information and exit

**Author**  
Written by David MacKenzie.

**Reporting Bugs**  
Report mkdir bugs to [bug-coreutils@gnu.org](mailto:bug-coreutils@gnu.org)  
GNU coreutils home page: <<http://www.gnu.org/software/coreutils/>>  
General help using GNU software: <<http://www.gnu.org/gethelp/>>  
Report mkdir translation bugs to <<http://translationproject.org/team/>>

**Air purification**  
| Lyreco

Waarom Fellowes luchtreinigers? Voor 4-fasen filtratie met True H13 HEPA filter en meer.



# Documentation



## ■ Sources

### ■ ChatGPT

### ■ Advantage

- Quick/elaborate explanations to your specific prompt
- Works well for CLI but also when programming!

### ■ Disadvantages

- Sometimes false/inaccurate information
- Need for double checking/testing

P can you explain this sed command for me?  
`sed -i '1n; s/,/\./g ; s/XY/X/g' '$file'`

Certainly! The given `sed` command performs several operations on a file. Let's break it down step by step:   

1. `sed` is a command-line utility for text stream editing in Unix-like operating systems.
2. The `-i` option is used to edit the file in-place, meaning the changes will be made directly to the file rather than printing the modified output to the terminal.
3. `'1n; s/,/\./g ; s/XY/X/g'` is the set of sed commands enclosed in single quotes. Each command is separated by a semicolon (;). Let's break down each command:
  - `'1n'`: This command prints the first line of the file.
  - `'s/,/\./g'`: This command substitutes all occurrences of a comma (`,`) with a period (`. `) in the file. The `s/` indicates a substitution command, `,` is the pattern to be replaced, `\.` is the replacement pattern, and `g` is the global flag to replace all occurrences in each line.
  - `'s/XY/X/g'`: This command replaces all occurrences of "XY" with "X" in the file. It follows a similar pattern as the previous substitution command.
4. `'$file'` is the file name provided as an argument to the `sed` command. The ``${code>sign is used to refer to the value of the variable "file" (assuming it has been previously defined).`

In summary, the `sed` command edits the file in-place by performing three operations:

1. Prints the first line of the file.
2. Replaces all commas (`,`) with periods (`. `) throughout the file.
3. Replaces all occurrences of "XY" with "X" throughout the file.

# Documentation

## ■ Sources

- Make sure to double check
  - test code, critically inspect output
  - There might be easier solutions
  - Compare to other documentation



P can you explain this sed command for me?  
`sed -i '1n; s/,/\./g ; s/XY/X/g' '$file'`



Certainly! The given `sed` command performs several operations on a file. Let's break it down step by step:



1. `sed` is a command-line utility for text stream editing in Unix-like operating systems.
2. The `-i` option is used to edit the file in-place, meaning the changes will be made directly to the file rather than printing the modified output to the terminal.
3. `'1n; s/,/\./g ; s/XY/X/g'` is the set of sed commands enclosed in single quotes. Each command is separated by a semicolon (;). Let's break down each command:
  - `1n`: This command prints the first line of the file.
  - `s/,/\./g`: This command substitutes all occurrences of a comma (`,`) with a period (`. `) in the file. The `s/` indicates a substitution command, `,` is the pattern to be replaced, `\.` is the replacement pattern, and `g` is the global flag to replace all occurrences in each line.
  - `s/XY/X/g`: This command replaces all occurrences of "XY" with "X" in the file. It follows a similar pattern as the previous substitution command.
4. `'$file'` is the file name provided as an argument to the `sed` command. The ```` sign is used to refer to the value of the variable "file" (assuming it has been previously defined).`

In summary, the `sed` command edits the file in-place by performing three operations:

1. Prints the first line of the file.
2. Replaces all commas (`,`) with periods (`. `) throughout the file.
3. Replaces all occurrences of "XY" with "X" throughout the file.

# 2. Exploring the command line

# Command line basics

- **“\$”**: text preceding it = prompt; text followed by it = command
- **Type a command, press enter to execute**
- **Autocomplete with TAB key**
- **Commands are**
  - Case sensitive (myFile vs myfile)
  - Space sensitive (my file vs myfile)
- **Mouse/clicking doesn't work! 😊**
  - Edit commands: use arrow keys (left and right)
  - Command history: arrow keys (up and down)

# Command line basics

- **Copy pasting does not always work the same (depends on OS)**
  - Instead of CTRL + C/V, try CTRL + shift + C/V
- **No undo button => watch out with replacing/removing files/dirs.**

# What are commands?

- **Command = specific instruction**
  - Given by the user, directed at the computer's operating system (windows, linux,...)
  - Goal: let the computer perform a task or function
- **Examples**
  - Changing location
  - Creating directories (folder), files, ...
  - Change name of files, move things around
  - Download software
- **Why? (not GUI?)**
  - Faster, more powerful, can combine multiple actions in one command
  - Not all tools/programs have a GUI

# Anatomy of a command - simple

- **Single command**

- Some commands can be run by themselves, e.g.:
  - \$ cd = go to home directory (/home/user)
  - \$ ls = list the contents of the folder you're currently in
- Most of the cases, they need some extra information...

# Anatomy of a command - simple

- **Adding arguments: tell a command what to perform the action on**
  - **command** + *argument(s)*
    - `$ cd /home/training`
    - `$ ls my_data/`
  - How many arguments?
    - Depends on command, sometimes optional
      - E.g. `$ mkdir folder1 folder2`
      - Creates 'subfolder1' and 'subfolder2'
    - Sometimes required
      - E.g. `$ mv important_file.txt important_folder`
      - Moves file (first argument) into directory (second argument)

# Anatomy of a command - intermediate

- **Adding options: modify a command's behaviour**

- **command** + options(s) + *argument(s)*
- Options change behavior of commands
  - \$ **ls** -l /home/training
  - Print the content of the directory as before, but with more details (date created/modified, file size...)
- Multiple options:
  - \$ **ls** -l -a -i
- Options start with one – or two - -; e.g.:
  - \$ **ls** -a : short option, usually one letter
  - \$ **ls** --all: long option, usually a word
  - *Both do the same*

# Anatomy of a command - options

- **Options:**

- Short options can be grouped, long options can't
  - `$ ls -l -a -t -r ≡ ls -latr`
  - `$ ls --list --all` : can't group, write in full
- Options don't always have both a short and a long version
  - Depends on the command
  - Check documentation (see later)
- Some options need a value (integer or string; number or text)
  - `$ head -n 15`
  - `$ biotool --input /home/user/RNAseq_data --output_type csv`

# Anatomy of a command: options

- **Other example:**

- \$ rm and \$ cp : when moving/copying folders you need to add the option -r
- Error often hints at the solution

```
training@ngscoursevm:~$ ls
folder1
training@ngscoursevm:~$ cp folder1 folder2
cp: -r not specified; omitting directory 'folder1'
training@ngscoursevm:~$ cp -r folder1 folder2
training@ngscoursevm:~$ ls
folder1 folder2
training@ngscoursevm:~$ █
```

# The terminal

```
elinet@ELinformatica: ~  
elinet@ELinformatica:~$ ls  
BIOMINA-24-005      project  
BIOMINA-24-011      results  
Python-3.12.0       snap  
Python-3.12.0.tgz   software  
R                   teaching  
env_squiggle_linux.yaml  test1  
get-docker.sh        training  
guppy_basecaller_log-2023-10-10_16-06-49.log  wget-log  
mambaforge           work  
elinet@ELinformatica:~$
```

```
john@Johns-MacBook-Pro ~ % sudo /Applications/Install\ macOS\ Sonoma.app/Contents  
/Resources/createinstallmedia --volume /Volumes/MyVolume  
Password:  
Ready to start.  
To continue we need to erase the volume at /Volumes/MyVolume.  
If you wish to continue type (Y) then press return: Y  
Erasing disk: 0%... 10%... 20%... 30%... 100%  
Making disk bootable...  
Copying to disk: 0%... 10%... 20%... 30%... 40%... 50%... 100%  
Install media now available at "/Volumes/Install macOS Ventura"  
john@Johns-MacBook-Pro ~ %
```

# The terminal: command prompt

- `eturcksin@DESKTOP1234:~$` *<here your command will go>*
  - `eturcksin` = username
  - `DESKTOP-1234` = name of device you're logged into (pc, server, ...)
  - `~` = current directory (after login, your home folder, so `~`)
  - `$` : separator between the end of the prompt and the command your typing
    - *Can differ between Linux flavors, can also be # or %*
    - *Command prompt is often abbreviated to "\$" instead of e.g. "eturcksin@DESKTOP1234:~\$"*

# Quick recap

```
dtrump@imthebest:~/my_projects/US_elections$ rm -r evidence/
```

Diagram illustrating the components of the command:

- `dtrump`: user
- `@imthebest`: device
- `:~/my_projects/US_elections`: current directory
- `$`: prompt
- `rm`: cmd
- `-r`: option
- `evidence/`: argument

# Help with commands: shortcuts

Your cursor does not work in the terminal 😊

→solution: arrows and shortcuts

Shortcut	Function
CTRL L	Prompt moves up; clears terminal
CTRL C	Kill process you're executing
CTRL arrows	Move along words instead of characters
CTRL A/CTRL E	Go to beginning/end of command
CTRL D	Exit terminal

# Help with commands: TAB completion



- **The TAB key is very useful when typing in the CLI**
  - Assists in completing command names and file/folder paths
  - Single TAB: completes your prompt when only one possibility is present

```
training@ngscoursevm:~/linux_intro/philip_holmgren$ ls
data_files file2
training@ngscoursevm:~/linux_intro/philip_holmgren$ cd d
```



```
training@ngscoursevm:~/linux_intro/philip_holmgren$ ls
data_files file2
training@ngscoursevm:~/linux_intro/philip_holmgren$ cd data_files/
```

- Double TAB: when multiple possibilities exist, a suggestion list is given
  - Add the specific part, then TAB again to complete

```
training@ngscoursevm:~/linux_intro/philip_holmgren$ cd data
data2/      data_files/
training@ngscoursevm:~/linux_intro/philip_holmgren$ cd data
```

- It works similar for commands
- Saves a lot of typing!

# Very useful: wildcards

- **Wildcards = characters that can represent one or more other characters**
  - Useful to collect file/folders with a common name part
  - Pattern searching
  - \* : any number of any character
    - `$ ls *.fastq.gz` (Lists all the files ending in fastq.gz)
  - ? : any single character
    - `$ ls sample.chr?.bam`
    - Will match `sample.chr8.bam` but not `sample.chr10.bam`
  - `[0-9]/[a-z]/[A-Z]` : a range of digits or letters
    - `$ ls sample.chr1[0-9].bam`
    - Will match `sample.chr10.bam`, `sample.chr11.bam` ... `sample.chr19.bam`
  - You can combine wildcards for more complex patterns

# Help with commands: good to know

- **Case sensitivity**
  - Linux commands are (mostly) all lower-case
  - CD : will return the error 'CD: command not found'
  - Common source of errors on CLI caused by using the wrong case
- **There is no undo in Linux**
  - Deleting/overwriting is permanent!
  - 'Are you sure you want to delete this?' : you better 😊
  - Be mindful of 'destructive' commands and check your commands before executing them
    - remove, overwrite...

# Exercises part 2

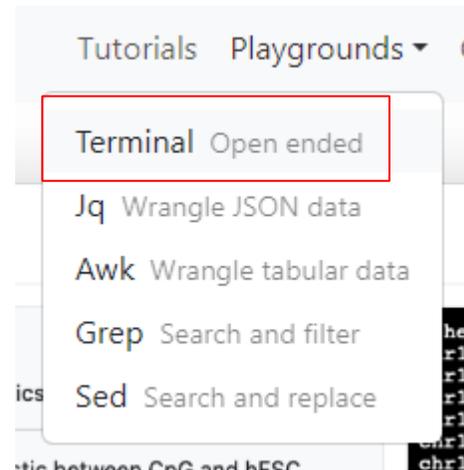
# Terminal simulator

- **Uniform terminal simulator to practice commands**

- <https://sandbox.bio/>
- In nav header: Playgrounds > Terminal

Or directly to

- <https://sandbox.bio/tutorials/playground>



# Some exercises

## ▪ Intro - demo

### ▪ Let's create a little directory structure

- `$ ls`
- `$ mkdir james`
- `$ ls`
- `$ cd james`
- `$ mkdir projects data Downloads`
- `$ ls`
- `$ cd projects`
- `$ mkdir RNA_seq fwo paper`
- `$ touch fwo/summary.txt`
- `$ ls`

### ▪ Tip: try to find directories and files by using tab

# Some useful basic commands

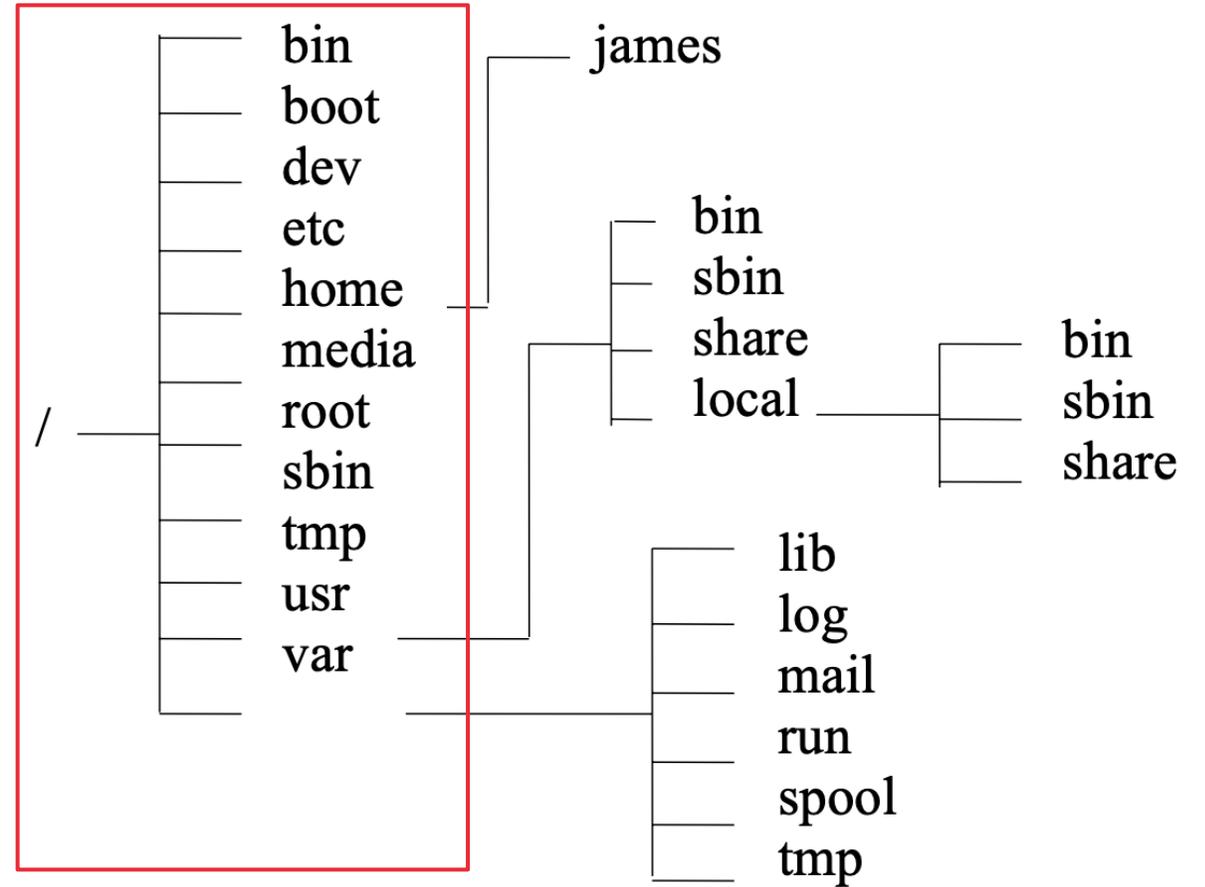
- **ls = list directory (folder) contents**
- **cd = change directories**
- **pwd = present work directory = current location**
- **touch = make a file**
- **mv = rename or move file/directory**
- **cp = copy file/directory**
- **rm = delete files/directories (!! Irreversible)**

# 3. Navigating the filesystem

# Linux Filesystem: root

## ■ Root

- Only the 'superuser' can perform cmds
  - = administrator space
  - In practice: put "sudo" in front of cmds
    - Only works in a space where you have administrator privileges
- Indicates administrator permission
- Need to type password (not visible!)

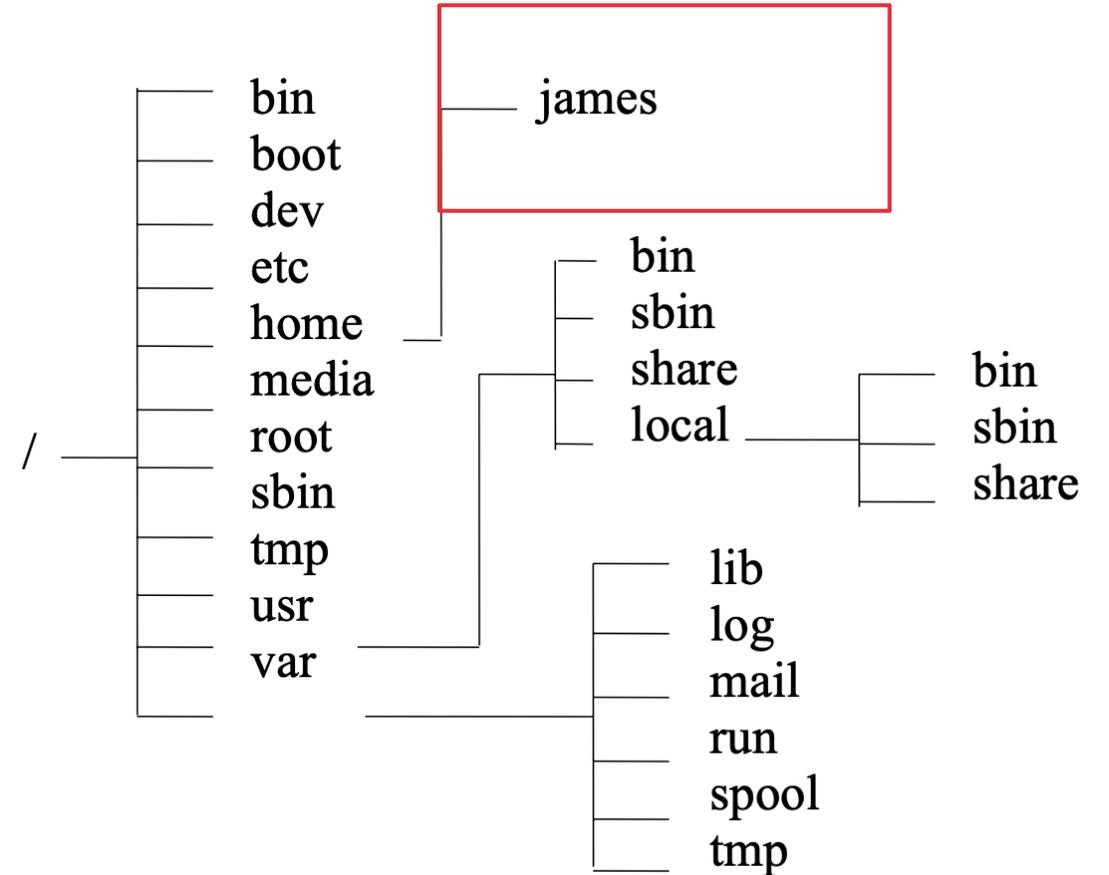


# Linux Filesystem: home

- **/home/user**

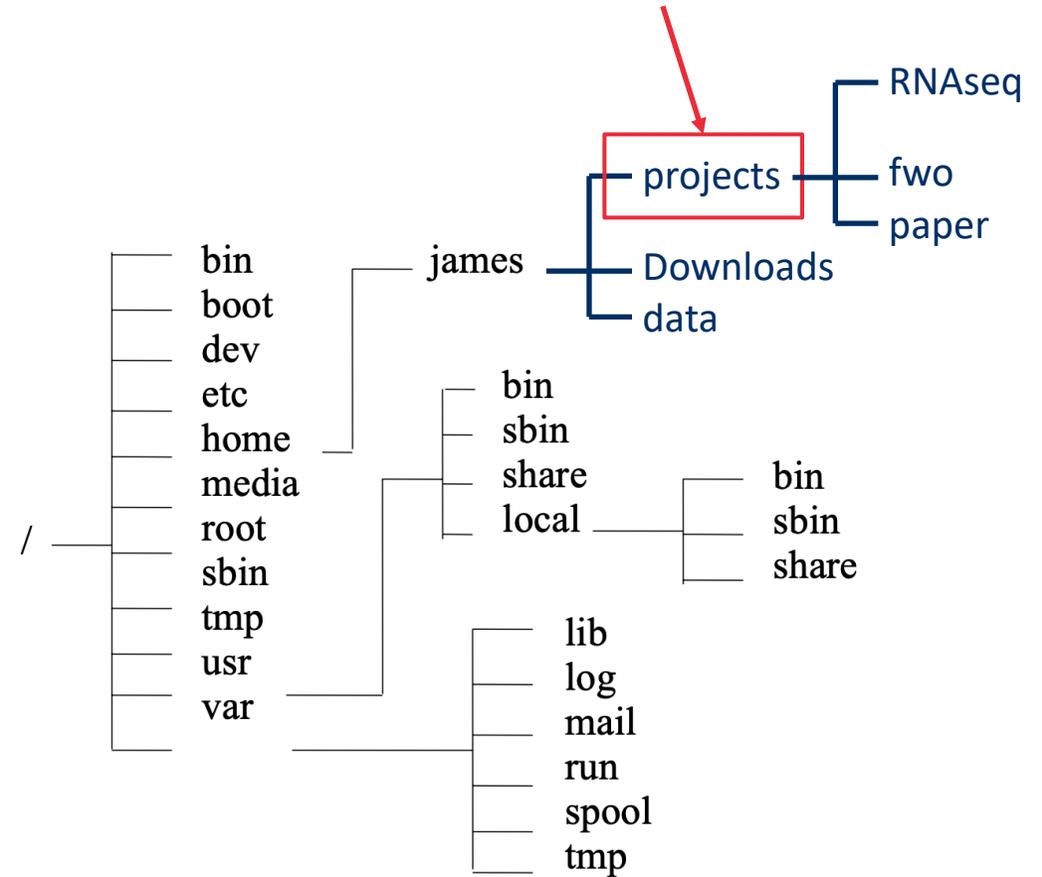
- Home contains directories with usernames
  - E.g. james, marie, guest, ...
- Similar to users in Windows
- In your own directory: free to execute cmds
- = space where you will perform your work
- Often abbreviated as ~

- **\$ cd without arguments will bring you always to /home/user**



# Navigating the file system

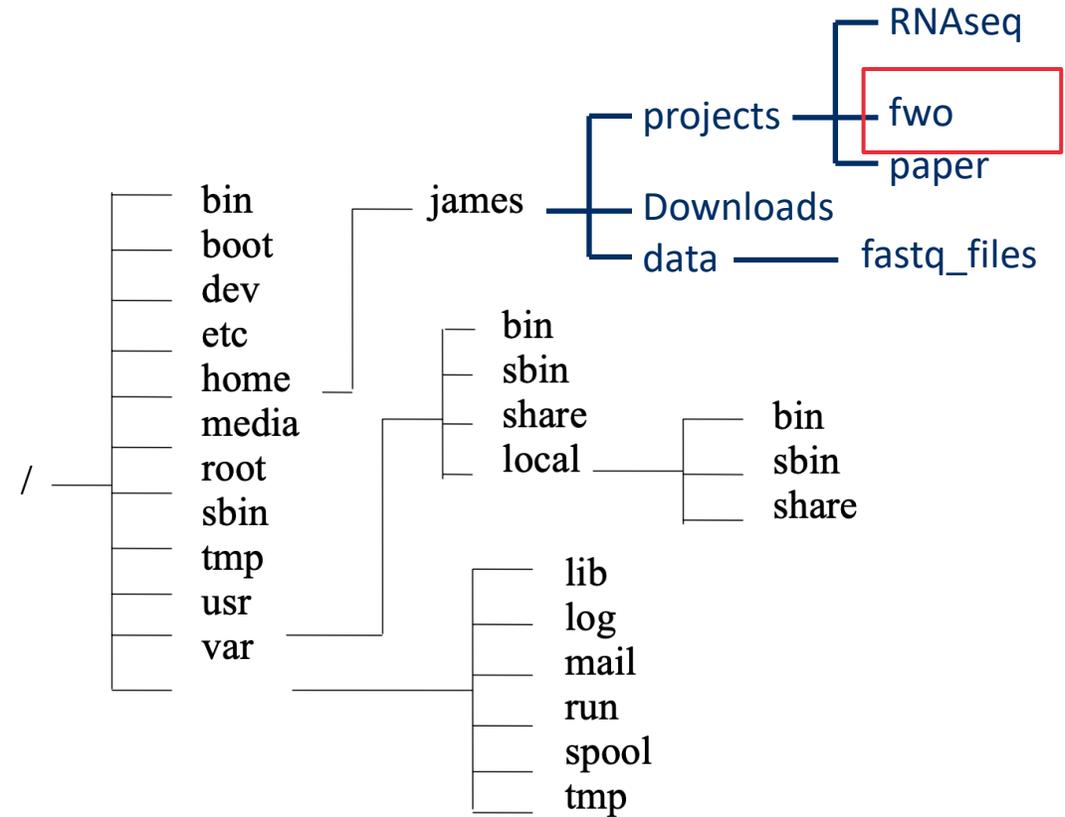
- **Command cd (change directory)**
- **Current location: projects**
- **Want to go to 'fwo'**
- **Absolute path**
  - Full path starting from root
  - `$ cd /home/james/fwo`
- **Relative path**
  - Respective from the current directory
  - `$ cd fwo`



# Navigating the file system

## ▪ Shortcuts for navigating

- Current directory: . (single dot)
  - `cd fwo`  $\equiv$  `cd ./fwo`
- 1 directory above: .. (double dot)
  - `cd ..`  $\Rightarrow$  projects
  - `cd ../../`  $\Rightarrow$  james
- Your home folder: ~ (tilde)
  - `cd /home/james`  $\equiv$  `cd ~` or `cd ~/`
  - `cd /home/james/fwo`  $\equiv$  `cd ~/fwo`
- Previous folder: - (hyphen)
  - `cd -`
- Long paths are possible
  - `cd ../data/fastq_files`



# Exercises part 3

# Some useful basic commands

- **ls = list directory (folder) contents**
- **cd = change directories**
- **pwd = present work directory = current location**
- **touch = make a file**
- **mv = rename or move file/directory**
- **cp = copy file/directory**
- **rm = delete files/directories (!! Irreversible)**
- **rmdir = remove (empty) directories**

# Some exercises

- **Exercise 1: start in project directory**

- What is the difference between
  - `$ touch file1.txt .`
  - `$ touch file2.txt`
  - `$ touch ../file3.txt`
  - `$ touch fwo/file4.txt`

The difference between the commands is that the two first commands will make the files `file1.txt` and `file2.txt` in the project directory (your current directory). The third command will make the `file3.txt` in the directory above, here the james directory. The last command will make the `file4.txt` in the `fwo` directory which is located in your current directory.

# Some exercises

## ▪ Exercise 2: Start in the james directory

1. Go to /sys/module making use of a relative path

```
cd ../../../sys/
```

1. Navigate to the kernel directory, located in sys. Make use of an absolute path. Then go back with `cd -`. Navigate again the that directory again with a relative path instead

```
cd /sys/kernel
```

```
cd -
```

```
cd kernel/
```

2. List the contents of the kernel directory

```
ls
```

3. Now without moving out of the kernel directory, list the contents of the power directory (it's a subdirectory of sys)

```
ls ../power
```

1. Make a new directory in the kernel directory. What happens?

```
mkdir: cannot create directory new_dir: Operation not permitted
```

This is because you don't have the right permissions to make a directory in this place, because this is a directory in your root system, on your own computer you can do this with adding "sudo" upfront.

# Some exercises

## ■ Exercise 3.1

- Use the man pages, the --help option, ChatGPT to learn about the following commands (how and why to use), compare the answers
  - `$ pwd`
    - Accessing the man pages: `man pwd`
    - Accessing the help page: `pwd --help`
  - `$ mv`
    - `man pwd`
    - `pwd --help`
  - `$ cp`
    - `man pwd`
    - `pwd --help`
- Use this information to complete the next exercises

# Some exercises

## ■ Exercise 3.2

1. Go to your home directory again, or the folders below if needed
2. Copy the summary.txt file from the fwo directory to the data directory  
`cp james/projects/fwo/summary.txt james/data/`
3. Rename the summary.txt file in the data directory to boring\_exercise.txt  
`cd james/data/`  
`mv summary.txt boring_exercise.txt`

# Some exercises

- **Exercise 4**
- **The same as exercise 3 but you must stay in the Downloads directory**
  1. Navigate to the Downloads directory  
`cd /root/tutorial/james/Downloads`
  1. Copy the summary.txt file from the fwo directory to the data directory  
`cp ../projects/fwo/summary.txt ../data/`
  1. Rename the summary.txt file in the data directory to boring\_exercise.txt  
`mv ../data/summary.txt ../data/boring_exercise.txt`

# Some exercises

## ▪ Exercise 5.1: ls

1. Create a file called `.secret_message.txt` in the data directory

`touch data/.secret_message.txt`

1. Can you find it using `ls`? Use info online to solve this

No, you can solve this by adding the `-a` option

1. Delete the `boring_exercise.txt` file

`rm data/boring_exercise.txt`

1. Remove the data directory

`rmdir data/` -> doesn't work because the directory isn't empty

`rm data/` -> doesn't work because it is a directory and not a file, this can be solved by adding the `-r` option

1. Problem/error? Try to google it!

# Some exercises

## ■ Exercise 5.2: `ls`

1. Try to find information about other `ls` options

`man ls`

`ls --help`

2. What do all these columns in `$ ls -l` mean?

```
root@localhost:~/tutorial/jamers# ls -l
total 2
drwxr-xr-x 2 root root  51 Nov 28 07:39 data
drwxr-xr-x 2 root root  51 Nov 28 07:39 Downloads
drwxr-xr-x 5 root root 138 Nov 28 07:39 projects
```

The first column is about permissions, the second one is the link count (the amount of links to the file or directory), then you have the owner's name and the group name. The fifth column is the size, then you have the date and hour of last modified. And as last the name of the file/directory

# Some exercises

## ■ Exercise 6: naming constrictions

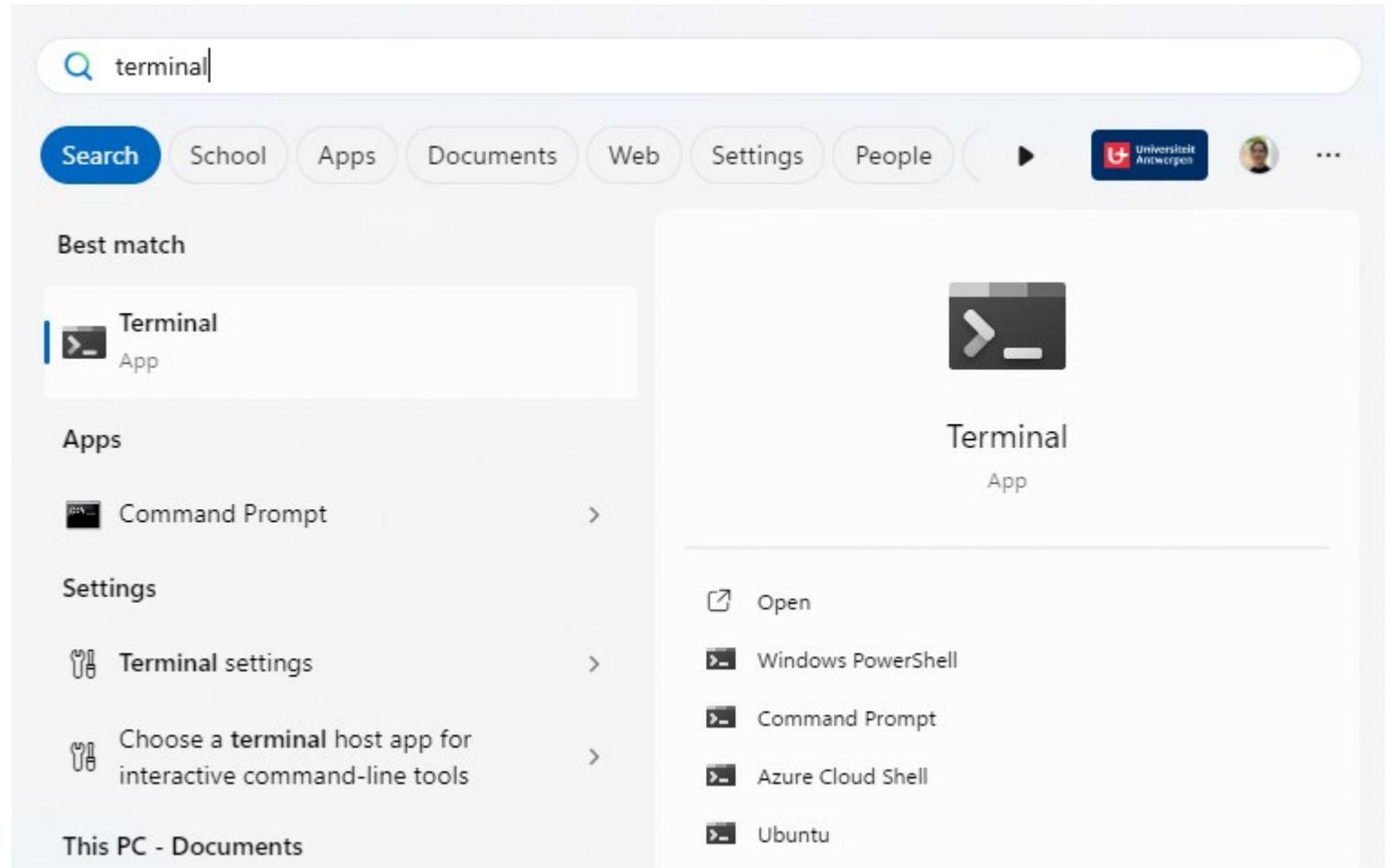
- Some characters are forbidden as a file or directory name ('/'), while some characters are considered bad practice ('\*', space, ':', '~', etc) and should be avoided.
- Why is '/' forbidden and why should '\*', space, ':' or '~' be avoided?
  - See comments for complete information about this
- If you do happen to have a file with '\*', space or '~' in the name, how could you quote it (i.e. refer to it) in the command line?

You can use “ ” or ‘ ’ or put an / upfront of the character you don't to have interpreted.

# Intermezzo: WSL intro

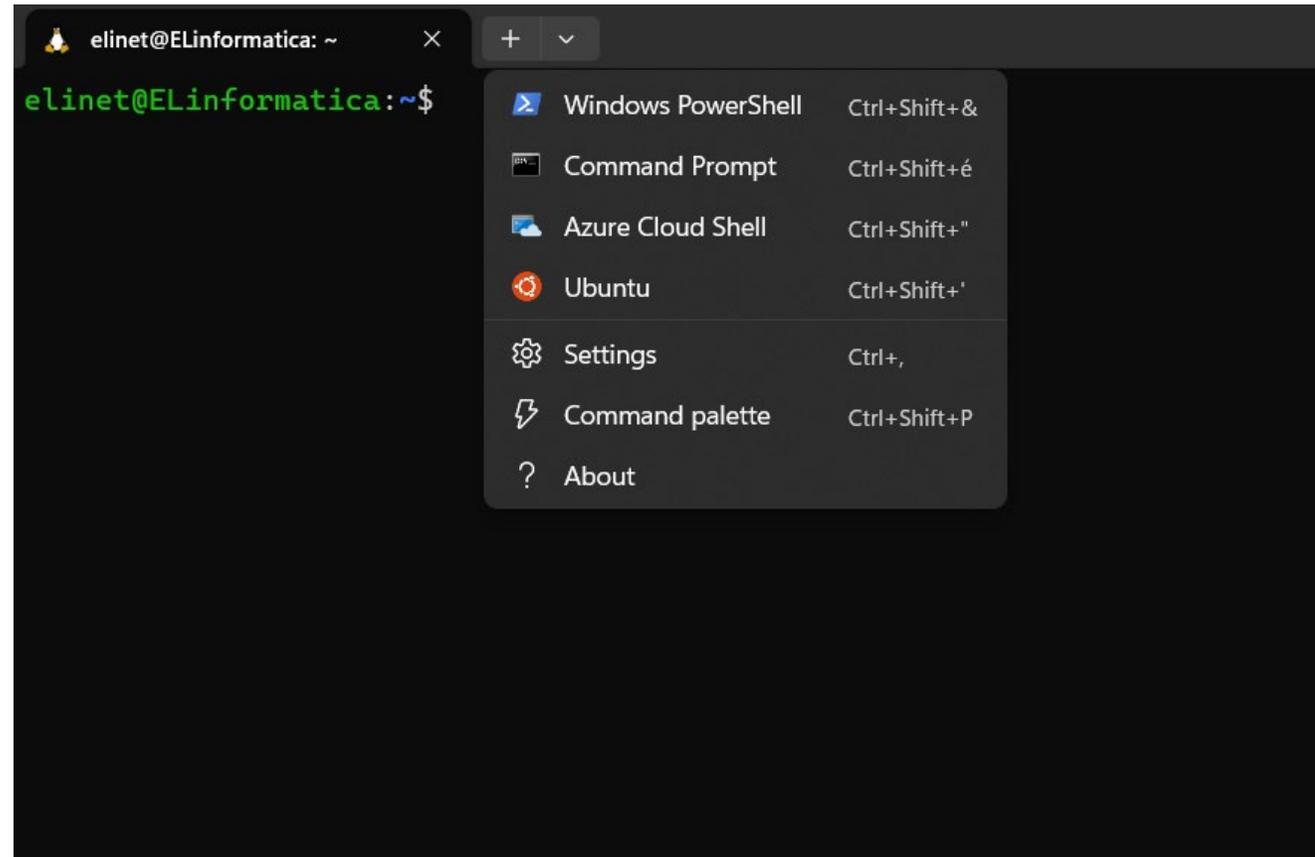
# WSL

- **Subsystem → Linux system inside of your Windows system**
- **Access via terminal**



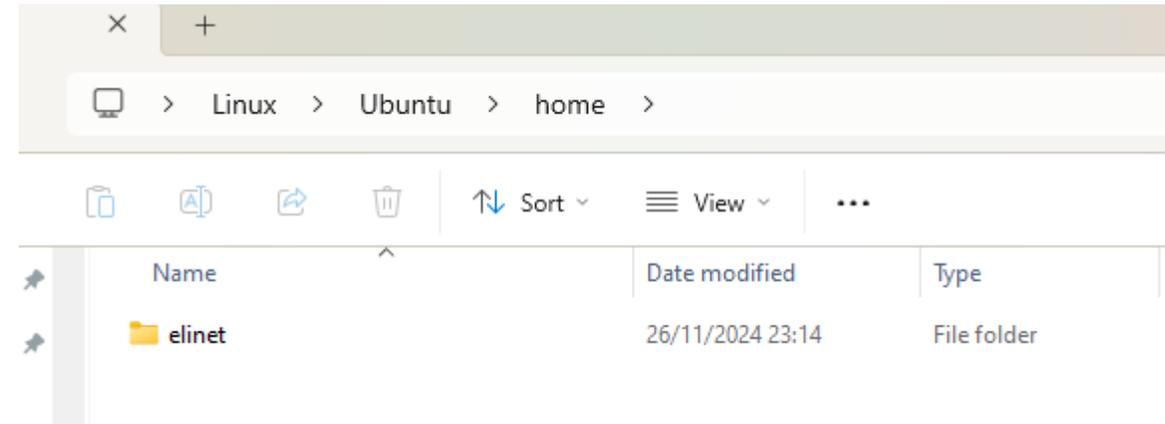
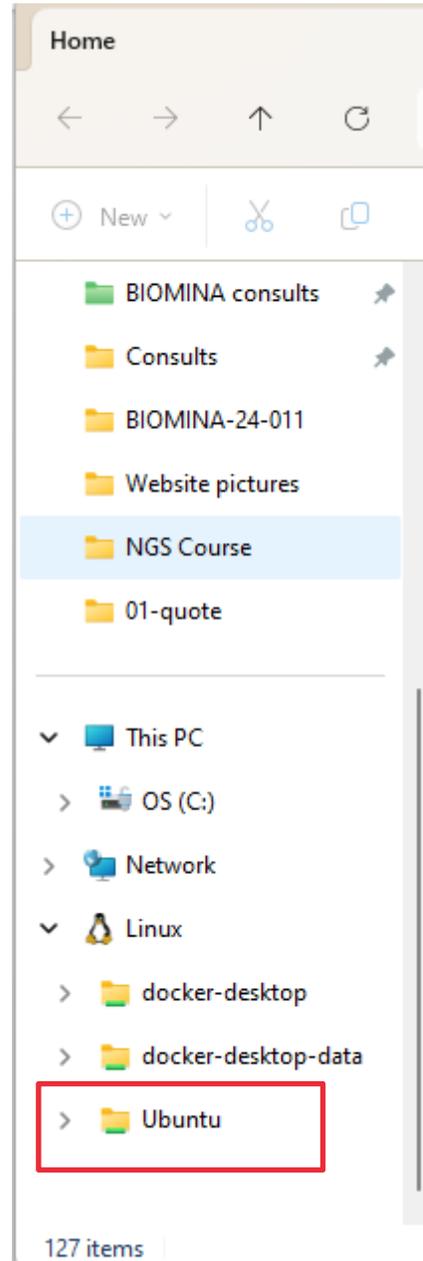
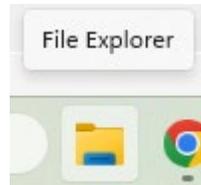
# WSL

- **Subsystem → Linux system inside of your Windows system**
- **Access via terminal**
- **This terminal will fulfill all your Linux hopes and dreams**



# WSL

- Access via file system



# WSL

- **Other way also possible: access “Windows” files via WSL terminal**
- **To go to your C drive: `$ cd /mnt/c`**
- **Navigate through your contents making use of Linux commands!**

# 4. Manipulating files and directories

# Reminders

- **Linux is case-sensitive**
  - `textfile.txt`  $\neq$  `Textfile.txt`
  - If both present in same directory: different files
- **Avoid using spaces in file/folder names**
  - Spaces separates different parts of your CLI command
  - Prone to make command mistakes
  - Preferably use underscores `_`, hyphens `-` or camelCase
    - `full_file_name.txt`
    - `fullFileName.txt`
  - *If you really want to use spaces, quote/escape your filename*
    - `"full file name.txt"`
    - `full\file\name.txt`

# Files & directories: useful commands

- **touch, mkdir, rm (-r)**
  - touch: make an empty file
  - mkdir: make a directory
  - rm: delete a file
    - -r : to delete a directory, add this option

# File management: compressed files

- **Linux variant of zip: .gzip**
  - Commands gzip/gunzip
  - `$ gzip textfile.txt > textfile.txt.gz`
  - `$ gunzip textfile.txt.gz > textfile.txt`
- **Read gzipped compressed files without unpacking**
  - `$ zcat`

```
localadmin@ngscoursevm:/home/training/data_files/fastq$ ls
Sample43_S91_L001_I1_001.fastq  Sample43_S91_L001_R1_001.fastq  Sample55_S55_L001_I1_001.fastq  Sample55_S55_L001_R1_001.fastq  Sample5_S55_L001_I1_001.fastq  Sample5_S55_L001_R1_001.fastq
Sample43_S91_L001_I1_001.fastq.gz  Sample43_S91_L001_R1_001.fastq.gz  Sample55_S55_L001_I1_001.fastq.gz  Sample55_S55_L001_R1_001.fastq.gz  Sample5_S55_L001_I1_001.fastq.gz  Sample5_S55_L001_R1_001.fastq.gz
Sample43_S91_L001_I2_001.fastq  Sample43_S91_L001_R2_001.fastq  Sample55_S55_L001_I2_001.fastq  Sample55_S55_L001_R2_001.fastq  Sample5_S55_L001_I2_001.fastq  Sample5_S55_L001_R2_001.fastq
Sample43_S91_L001_I2_001.fastq.gz  Sample43_S91_L001_R2_001.fastq.gz  Sample55_S55_L001_I2_001.fastq.gz  Sample55_S55_L001_R2_001.fastq.gz  Sample5_S55_L001_I2_001.fastq.gz  Sample5_S55_L001_R2_001.fastq.gz
localadmin@ngscoursevm:/home/training/data_files/fastq$ zcat Sample43_S91_L001_R1_001.fastq.gz | head -8
@M00984:63:000000000-BBJLL:1:1106:4432:22883 1:N:0:91
CGCACATTGCGCCCTCTGGTATTCAGAGGGCATGCCTGTTGAGCGTCATTTCTCTCAAACTTCGGGTTTGGTATTGAGTGATACCTTAGTGAAGTGGCGTTTCTTGAATGATATTGGCATGAGTGGTACTGGATAGTCTATGACTTTCAATGTATTAGGCTTCACTCACTCGTTGAATAGTTAATGGTATATTTT
AACAAAACGTTTTG
+
BBBBBBFFFBGGCGGGGGGGHGDGAFFGGFHHHHHHHGHGFFHGGGGHHHHHHHGFAD3FFFBFHFHGGGGF5FGHGF3@GFFEHFDFDHE??GGHHHFEFGFCG4GGFHGG4B4EHBFDFFC3?FG3BFHH23B2GFFBFFHHHGDG@G>2@?FF1@?<FHHHFD1?11GFB<0>G<1<1>FFF0=0=000<0G
FCFB7E..99;B;
@M00984:63:000000000-BBJLL:1:1107:8479:15588 1:N:0:91
GTAGCATATCAATAAGCGGAGGACCTGACTGACTTAGCAGCTATCTCGTATGCCGTCTCTGCTTGAATAAACTTAACTCTTTAGGTTCTGCCCTCTCATCTTATCATCTCCCTTTAACTACTCCCTCTCATCTTTTACTCCTTATTCTTCTTGTCTACTGCTTCCATCCATCACCCTCATCAACTCTCC
TTATCTCTATCTCT
+
11>1>-DFBFFFFADD1EE?EG0FGCH0GGFFHHBGGBFEGFFHHZABAGF2GEEHGHHHBGFGA110//011111B12211211111>2////08121222-12>>2B2BF0B1B1211@21221@01<00?1?7<1<?//11?1>=<11111=111>>1=1>000/<<./<00=..00=0:0./:0;0;/00:00
00;900900000//
localadmin@ngscoursevm:/home/training/data_files/fastq$
```

# File management: archiving files

- **tar**

- Makes a 'tarball' (.tar file): this is an archive file, holding multiple folders/files
- Optionally, compresses it (.tar.gz)
- Compress
  - `$ tar -zcf <COMPRESSED_FILENAME>.tar.gz <DIRECTORY>`
- Extract
  - `$ tar -zxf <COMPRESSED_FILENAME>.tar.gz <DIRECTORY>`
- -z: indicate the file is/will be zipped
- -c: compress mode
- -x: decompress mode
- -f: name of the source/destination (gzipped) tarball

# File management: archiving files

- Tar use?
  - To transfer data, easier to transfer 1 file instead of (deep) directory structure

```
localadmin@ngscoursevm:/home/training/data_files$ ls
Dummy_Data  ID_SNP  PYTHON_SCRIPTS  VARIANT_CALLING_FILES  bed_files  chr21.tar.gz  emails.txt  file.csv  fileA.txt  igv_bin  stdout.txt
Exome_Data  ID_SNP2 R_tutorial      adapters                chr21      dockers.txt  fastq       file.txt  fileB.txt  stderr.txt  test_file.txt
localadmin@ngscoursevm:/home/training/data_files$ tar czf ID_SNP2.tar.gz ID_SNP2
localadmin@ngscoursevm:/home/training/data_files$ ls
Dummy_Data  ID_SNP  ID_SNP2.tar.gz  R_tutorial      adapters  chr21      dockers.txt  fastq       file.txt  fileB.txt  stderr.txt  test_file.txt
Exome_Data  ID_SNP2 PYTHON_SCRIPTS  VARIANT_CALLING_FILES  bed_files  chr21.tar.gz  emails.txt  file.csv  fileA.txt  igv_bin  stdout.txt
localadmin@ngscoursevm:/home/training/data_files$ rm -rf ID_SNP2
localadmin@ngscoursevm:/home/training/data_files$ tar xzf ID_SNP2.tar.gz
localadmin@ngscoursevm:/home/training/data_files$ ls
Dummy_Data  ID_SNP  ID_SNP2.tar.gz  R_tutorial      adapters  chr21      dockers.txt  fastq       file.txt  fileB.txt  stderr.txt  test_file.txt
Exome_Data  ID_SNP2 PYTHON_SCRIPTS  VARIANT_CALLING_FILES  bed_files  chr21.tar.gz  emails.txt  file.csv  fileA.txt  igv_bin  stdout.txt
localadmin@ngscoursevm:/home/training/data_files$
```

# File management: archiving files

## ▪ Echo

- Example: `$ echo "This is a koffiekoek"`
- This command will “echo” the text that you give it
  - Here in the terminal `This is a koffiekoek` will appear
  - This is only outputted in the terminal and not saved anywhere

```
marie 10:31:30 11/28/24 🏠 ✓ echo "This is a koffiekoek"
This is a koffiekoek
marie 10:44:47 11/28/24 🏠 ✓
```

# Exercises part 4

# Very useful: wildcards

- **Wildcards = characters that can represent one or more other characters**
  - Useful to collect file/folders with a common name part
  - Pattern searching
  - \* : any number of any character
    - `$ ls *.fastq.gz` (Lists all the files ending in fastq.gz)
  - ? : any single character
    - `$ ls sample.chr?.bam`
    - Will match `sample.chr8.bam` but not `sample.chr10.bam`
  - `[0-9]/[a-z]/[A-Z]` : a range of digits or letters
    - `$ ls sample.chr1[0-9].bam`
    - Will match `sample.chr10.bam`, `sample.chr11.bam` ... `sample.chr19.bam`
    - To create files: use `{0..9}`
  - You can combine wildcards for more complex patterns

# Some exercises

## ▪ Intro

- Make a directory to work in, let's call it 'linuxcourse\_2024'
  - Put it in your personal (home) folder: `/home/<user>/linuxcourse_2024`
  - Go into this directory  
`mkdir /home/<user>/linuxcourse_2024`  
`cd /home/<user>/linuxcourse_2024`
- Try the following commands and see what they do
  - `$ touch file.txt file2.txt`
  - `$ tar -czvf file.tar.gz file.txt file2.txt`
  - `$ ls -l`
  - `$ rm file.txt file2.txt`
  - `$ ls -la`
  - `$ tar -xzvf file.tar.gz`
  - `$ ls -la`

## ▪ Intermediate

- Make a subdirectory called 'exercise1' `mkdir exercise1`
- Make a file called 'test' with the text "This is a test" (reminder: echo) `echo "This is a test" > test.txt`
- Move the file into the directory 'exercise1' `mv test.txt exercise1/`
- Show all the content of the directory in long list form and make file sizes human readable `ls -lh exercise1/`
- Make a tar file of the directory 'exercise1' `tar -cvf exercise1.tar exercise1/`

# Some exercises

- **Advanced**

- Make in one command 3 files with the following names file1A, file2A and file3B.

`Touch file{1A,2A,3B}`

- Make 10 files and they all are called file\_test1, file\_test2, ..., file\_test10

`Touch file_test{1..10}`

- Move all the file with an “A” in the 'exercise1' directory

`Mv file*A exercise1/`

- Write to the files with a “A” “this file is an A file”

- Reminder: what directory are you in right now?

`echo "this file is an A file" > file1A; echo "this file is an A file" > file2A`

- Remove the files file\_test1, file\_test2, ..., file\_test5

- Try it using a wild card **!!! WATCH OUT WITH \$ rm \***

`Rm file_test{1..5}`

- Make a copy of the exercise directory and give it the name “exercise2”

`Cp exercise1/ exercise2`

- Make an exercise3 directory

`Mkdir exercise3/`

- Move the rest of the test files (6-10) in 'exercise2'

`Mv file_test{6..10} exercise2`

- Finally, remove both these directories (2 and 3)

`rmdir exercise3/`

`Rm -r exercise2/`

-> This must be done differently because the exercise3 directory is empty and the exercise2 directory contains files.

# Some exercises

- **Prepare downloaded data**

- Data downloaded using intro (<https://tinyurl.com/linuxcourse2024>)
  - Is a .tar.gz file
- Move and extract the file from your Downloads folder to the working directory 'linuxcourse\_2024'

You can do this by the File Explorer on Windows

You can find your directory of your WSL by the following path

`\\wsl.localhost\Ubuntu\home\<<your_user_name>`

Or go in linux to your windows file system by the following path

`Cd /mnt/c/Users/<username_own_computer>/Downloads/`

- Will take a bit of time, just let it run

# 5. File contents

# Reading text files

- **cat ('concatenate')**
  - Prints one or more files to the screen

```
localhost@ngscoursevm:/home/training/data_files$ cat fileA.txt
rowA
rowB
localhost@ngscoursevm:/home/training/data_files$ cat fileB.txt
rowI
rowII
localhost@ngscoursevm:/home/training/data_files$ cat file[A-B].txt
rowA
rowB
rowI
rowII
localhost@ngscoursevm:/home/training/data_files$ █
```

# Reading text files

- **head/tail [-<number\_of\_lines>]**
  - Prints the first/last lines of a file. Optionally, you can give the number of lines as an option
  - Convenient for big files or when you only want the beginning/end of a file

```
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ head -3 file.csv
sample,raw.reads,aligned.reads,avg.bcov,p10x,p20x,Run
144928_s12,29660708,42679036,44.716,0.05498,0.16008,3
138578,133206094,98900104,96.990,0.02883,0.05631,1
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ tail -5 file.csv
120174,112289378,83869627,92.419,0.02732,0.05322,1
145314_s13,21980408,34251706,35.385,0.09070,0.26475,3
143664_s5,114484676,50864119,54.738,0.03909,0.10284,2
83477_s2,22047074,34737358,36.760,0.06369,0.20144,3
142443_s25,42113472,39925200,40.681,0.05439,0.17314,3
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$
```

# Reading text files

- **cut**

- Print specific columns of a file

- -f : number of column(s), can be a number, list or range
    - Default column separator/delimiter is TAB
      - -d : specify different delimiter
    - | head -4 : Linux pipe (ignore for now)

```
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ cut -f1 file.txt | head -4
sample
144928_s12
138578
135189
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ cut -f1,3 file.txt | head -4
sample aligned.reads
144928_s12      42679036
138578  98900104
135189  87900212
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ cut -f1-3 file.txt | head -4
sample raw.reads      aligned.reads
144928_s12      29660708      42679036
138578  133206094      98900104
135189  121362682      87900212
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$
```

```
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ cut -d, -f1,3 file.csv | head -4
sample,aligned.reads
144928_s12,42679036
138578,98900104
135189,87900212
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$
```

# Reading text files

- **sort**

- Print the sorted lines of a file

- Default sort: alphabetically
- -n : numeric sort
- -u : sort and removes duplicate lines
- -k: sort on 1 or more columns
- -V: version sort, use the numerical part to sort. Example: V1, V2, V3
- -t: change the column delimiter (when using -k). Default is TAB

```
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ # -k2,2n : numerical sort on only the second column. -k always needs a start and end column
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ sort -k 2,2n file.txt | head -4
sample raw.reads aligned.reads avg.bcov p10x p20x Run
126895 131176 64548 0.066 1.00000 1.00000 1
135038_s16 19646710 30228369 28.003 0.13701 0.38169 3
133915_s4 21500824 33383572 31.840 0.09058 0.28122 3
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ # -k2,2n -k1,1n : first sort on second, then on first column
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ sort -t, -k2,2n -k1,1n file.csv | head -4
sample,raw.reads,aligned.reads,avg.bcov,p10x,p20x,Run
126895,131176,64548,0.066,1.00000,1.00000,1
135038_s16,19646710,30228369,28.003,0.13701,0.38169,3
133915_s4,21500824,33383572,31.840,0.09058,0.28122,3
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$
```

# Reading text files

- **WC**
  - Get a word or line count for a file
    - Default returns 3 numbers: line count, word count, byte count
    - -l : only print line count (mostly used)
    - When given multiple files, it will give individual file and overall counts

```
localadmin@ngscoursevm:/home/training/data_files$ cat emails.txt
john.doe@example.com
alice_smith@email.co.uk
invalid-email
bob_jones123@gmail.com
charlie-brown@yahoo.com
eve@example.org
invalid.email@.com
localadmin@ngscoursevm:/home/training/data_files$ wc emails.txt
 7  7 141 emails.txt
localadmin@ngscoursevm:/home/training/data_files$ wc -l emails.txt
7 emails.txt
localadmin@ngscoursevm:/home/training/data_files$
```

# Text patterns search

- **grep**
  - Print the lines of a file that match a certain word/pattern
  - -i: case-insensitive search
  - -n: show the matching line number
  - -v: reverse, only show not matching lines
  - -P : give a regular expression to match

```
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ grep -n rowB test_file.txt
2:rowB
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ grep -ni rowb test_file.txt
2:rowB
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$ grep -vni rowb test_file.txt
1:rowA
3:rowC
4:rowD
5:rowS
training@ngscoursevm:~/linux_intro/philip_holmgren/data_files$
```

# Text patterns search

- **Regular expressions (regex for short):**
  - Same idea as wildcards but more powerful
  - Groups and ranges
    - . (dot) : any character
    - [0-9]: any digit
    - [a-zA-Z]: any letter (upper/lower case)
  - Quantifiers
    - \* : zero or more characters
    - + : 1 or more characters
    - ? : 0 or 1 character
    - {m}: m characters ; {m,} : m or more characters ; {m,n} : number characters ranges from m to n
  - Anchors
    - ^ : from beginning of the line
    - \$: at the end of the line
  - Very extensive:
    - <https://cheatography.com/davechild/cheat-sheets/regular-expressions/>

# Text patterns search: grep/regular expressions

- Grep examples

```
localadmin@ngcoursevm:/home/training/data_files$ cat emails.txt
john.doe@example.com
alice_smith@email.co.uk
invalid_email
bob_jones123@gmail.com
charlie-brown@yahoo.com
eve@example.org
invalid_email@
localadmin@ngcoursevm:/home/training/data_files$ grep -P '[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}' emails.txt
john.doe@example.com
alice_smith@email.co.uk
bob_jones123@gmail.com
charlie-brown@yahoo.com
eve@example.org
localadmin@ngcoursevm:/home/training/data_files$
```

```
localadmin@ngcoursevm:/home/training/data_files$ cat dockers.txt
REPOSITORY TAG IMAGE ID CREATED SIZE
391209680344.dkr.ecr.eu-west-1.amazonaws.com/aws_data_importer 1.4 1a9ec830c03b 22 hours ago 883MB
basespace_to_aws 1.4 1a9ec830c03b 22 hours ago 883MB
<none> <none> 558acc514cd8 22 hours ago 1.08GB
<none> <none> c1412423964d 22 hours ago 77.8MB
<none> <none> 4020ca978d13 22 hours ago 77.8MB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/upd-caller dev f62e3e6847af 4 weeks ago 4GB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/bedtools 2.29.2 73a68422e40a 5 weeks ago 112MB
bt dev 73a68422e40a 5 weeks ago 112MB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/swgs_cnv 1.1 fefd1210db1d 2 months ago 2.26GB
swgs_cnv 1.1 fefd1210db1d 2 months ago 2.26GB
swgs_cnv 1.0 c31fcef2cff4 4 months ago 2.26GB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/swgs_cnv 1.0 c31fcef2cff4 4 months ago 2.26GB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/swgs_qc 1.0 ea4643073be1 5 months ago 3.9GB
swgs_qc 1.0 ea4643073be1 5 months ago 3.9GB
ubuntu 18.04 f9a80a55f492 5 months ago 63.2MB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/wes_cnv 0.4.1 de651d8a354f 5 months ago 4.92GB
cromwell-dev latest 6405fb36d05b 7 months ago 801MB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/wes_cnv 201209680344.dkr.ecr.eu-west-1.amazonaws.com/swgs_cnv <none> 28db6f0e19e2 8 months ago 4.92GB
nv dev 79e9309d1a44 8 months ago 1.94GB
680344.dkr.ecr.eu-west-1.amazonaws.com/bowtie2 dev 79e9309d1a44 8 months ago 1.94GB
2 2.5.1 295eef4f7951 8 months ago 664MB
ta_importer 2 2.5.1 295eef4f7951 8 months ago 664MB
680344.dkr.ecr.eu-west-1.amazonaws.com/cutadapt 1.4 88ec7acaad97 8 months ago 967MB
pt 4.1 c58e97ada0b7 11 months ago 318MB
ace_to_aws 4.1 c58e97ada0b7 11 months ago 318MB
680344.dkr.ecr.eu-west-1.amazonaws.com/aws_data_importer 1.3 fada8dc57aef 16 months ago 802MB
22.04 27941809078c 17 months ago 77.8MB
latest 27941809078c 17 months ago 77.8MB
1.0 aaa7f75d5193 18 months ago 1.28GB
focal 2b4cba85892a 20 months ago 72.8MB
809758.dkr.ecr.eu-west-1.amazonaws.com/wings-client test 6a03c8e7e2be 24 months ago 12.9MB
latest 6a03c8e7e2be 24 months ago 12.9MB
o/hpc/gromacs 2021.3 5bdce2d86a60 2 years ago 854MB
ubuntu 20.04 f63181f19b2f 2 years ago 72.9MB
cmgantwerpen/varamplicnv 1.0.0 e40079facae1 3 years ago 1.37GB
cmgantwerpen/varamplicnv latest e40079facae1 3 years ago 1.37GB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/gatk 4.1.6.0 c8c598a2807a 3 years ago 2.74GB
localadmin@ngcoursevm:/home/training/data_files$ grep '<none>' dockers.txt
<none> <none> 558acc514cd8 22 hours ago 1.08GB
<none> <none> c1412423964d 22 hours ago 77.8MB
<none> <none> 4020ca978d13 22 hours ago 77.8MB
391209680344.dkr.ecr.eu-west-1.amazonaws.com/wes_cnv <none> 28db6f0e19e2 8 months ago 4.92GB
localadmin@ngcoursevm:/home/training/data_files$ grep -P '^<none>' dockers.txt
<none> <none> 558acc514cd8 22 hours ago 1.08GB
<none> <none> c1412423964d 22 hours ago 77.8MB
<none> <none> 4020ca978d13 22 hours ago 77.8MB
localadmin@ngcoursevm:/home/training/data_files$
```

# Before we do exercises, some tips

- **Inspect your input data before you begin**
  - Find patterns that you can use to parse or filter your wanted data
- **When using regexes**
  - Use the documentation or check online what the proper syntax is
  - You can copy-paste your regex in ChatGPT and just ask what's wrong with it
    - Or just ask it how to parse your data in the first place 😊

# Exercises part 5

# Exercises

- Go to `linuxcourse_2024/data_files/fastq`
  - Structure fastq file names
    - `<SAMPLE_NAME>_S<INCREMENT_NUMBER>_<SEQUENCING_LANE>_<READ>_001.fastq[.gz]`
  - List all files ending in "fastq" `ls -l *.fastq`
  - List all files with read 'R1' in file name `ls -l *R1*`
  - List all fastq files with Sample5 in file name `ls Sample5_*.fastq`
    - Try to avoid getting the files of sample Sample55
    - Now show the same files on screen using TAB completion
  - Using ls, can you find the largest file in this directory? `ls -lSh` (l = listform, S = sort, h = human readable)
    - Find out the ls option to sort on size
- Go to `linuxcourse_2024/data_files/ID_SNP`
  - List all VCF files with filename in the range of 200 and 399 `ls -l {200..399}.vcf`
    - Try to avoid .vcf.idx files
    - Use wildcards? Is it possible to get all files in 1 'range?' **yes**
      - If not, you can add additional items in your command (ls can take multiple arguments)
      - `ls <range with wildcards> ... ..`

# Exercises

- Go to `linuxcourse_2024/data_files/fastq`
  - How many lines does each **fastq** file of sample 5 contain? How much in total? `wc -l *fastq`
  - Get the first line from every .fastq file? `head -n 1 *.fastq`
  - Get all the lines (including line numbers) from all fastq files of sample 43 containing the string “@M00984”.  
`grep -n "@M00984" Sample43_S91_L001_*.fastq`
- Go to `linuxcourse_2024/data_files/ID_SNP`
  - *Show the lines in 095.vcf that do NOT start with a “#”.* `grep -v "^#" 095.vcf`
    - Reminder: reverse grep
  - In samples 1-9, find the variants with depth of coverage equal to 1 (DP=1) `grep "DP=1;" 00{1..9}.vcf`
    - Did you restrict to DP=1 and ignored values between DP=10 and DP=19?
    - Check the line structure on how to do this
  - For the same files, can you get depth of coverage between 10 and 99? `grep "DP=[1-9][0-9];" 00{1..9}.vcf`
    - Tip: amount of digits = 2

# 6. Input, output and pipelines

# Data handling

- **Up until now: print everything on the screen**
  - What if you want to write to a file?
- **Output streams**
  - Normal output (STDOUT)
  - Error output (STDERR)
- **Output redirection**
  - STDOUT: >
  - STDERR: 2>
- **Appending**
  - STDOUT: >>
  - STDERR: 2>>

```
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ grep -P "DP=\d{2,};" 00[1-9].vcf > /tmp/my_stdout.txt
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ head -5 /tmp/my_stdout.txt
001.vcf:chr1 67861520 rs2229546 C A 0 LowQual AC=0;AF=0.00;AN=2;DB;DP=39;Dels=0.00;FS=0.000;Haplotype
001.vcf:chr1 179520506 rs1410592 G A 1200.77 . AC=2;AF=1.00;AN=2;DB;DP=33;Dels=0.00;FS=0.000;Haplotype
001.vcf:chr2 169789016 rs497692 T C 296.77 . AC=1;AF=0.500;AN=2;BaseQRankSum=2.068;DB;DP=42;Dels=0
kSum=0.517 GT:AD:DP:GQ:PL 0/1:29,13:42:99:325,0,823
001.vcf:chr2 227896976 rs10203363 C T 671.77 . AC=1;AF=0.500;AN=2;BaseQRankSum=-0.009;DB;DP=51;Dels=0
RankSum=1.083 GT:AD:DP:GQ:PL 0/1:26,25:51:99:700,0,688
001.vcf:chr3 4403767 rs2819561 A G 488.77 . AC=2;AF=1.00;AN=2;DB;DP=19;Dels=0.00;FS=0.000;HaplotypeScore=0
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$
```

```
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ grep -P "DP=\d{2,};" 00[1-9].vf 2> /tmp/my_stderr.txt
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ head -5 /tmp/my_stderr.txt
grep: 00[1-9].vf: No such file or directory
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$
```

# Data handling

- **Separate streams but can be redirected to the same file**
  - Syntax: `> <FILE> 2>&1`
    - `grep something > /tmp/stdout.txt 2> /tmp/stderr.txt`

```
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ grep -P "DP=\d{2,};" 00[1-9].vf > /tmp/my_stdout.txt 2> /tmp/my_stderr.txt
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ grep -P "DP=\d{2,};" 00[1-9].vf >/tmp/my_bothstreams.txt 2>&1
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$ cat /tmp/my_bothstreams.txt
grep: 00[1-9].vf: No such file or directory
localadmin@ngscoursevm:/home/training/data_files/ID_SNP$
```

# Data handling

## ▪ Output of one command as input for another command

- Pipe |
- Write between the different commands
  - \$ grep something <FILE> | head -5
- Can be any number of commands
  - Can make big one-liner scripts with this
  - Really convenient to check something quickly



```
C1orf192
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | cut -d'|' -f2 | cut -d';' -f 1 | sort -u > /tmp/gene_list.txt
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | ^Ct -d'|' -f2 | cut -d';' -f 1 | head -5
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | head -5
gn|TF;ccds|CCDS3080;ens|ENSG00000091513;vegal|OTTHUMG00000150356
gn|RSPH4A;ccds|CCDS34521;ccds|CCDS55051;ens|ENSG00000111834;vegal|OTTHUMG00000015444
gn|NEK3;ccds|CCDS53871;ens|ENSG00000136098;vegal|OTTHUMG00000016958
gn|FTSJ3;ccds|CCDS11644;ens|ENSG00000108592
gn|C1orf192;ccds|CCDS30921;ens|ENSG00000188931;vegal|OTTHUMG00000034462
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | cut -d'|' -f2 | head -5
TF;ccds
RSPH4A;ccds
NEK3;ccds
FTSJ3;ccds
C1orf192;ccds
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | cut -d'|' -f2 | cut -d';' | head -5
cut: you must specify a list of bytes, characters, or fields
Try 'cut --help' for more information.
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | cut -d'|' -f2 | cut -d';' -f1 | head -5
TF
RSPH4A
NEK3
FTSJ3
C1orf192
localadmin@ngscoursevm:/home/training/data_files/bed_files$ cut -f4 large_gene_panel.bed | cut -d'|' -f2 | cut -d';' -f1 | sort -u > /tmp/gene_list.txt
localadmin@ngscoursevm:/home/training/data_files/bed_files$
```

# Before we do exercises, some tips

- **Again, investigate your input data**
  - Get a good feel of where you need to go
  - Then work top-down
    - Example: get the second tab-separated column, then get the fourth element separated with ";" , etc...
- **Check the intermediate output from each command**
  - Before you send it to another command
  - Best way to check which step has an issue if something goes wrong

# Exercises part 6

# Exercises

- Go to linuxcourse\_2024
  - Echo the text “Creating new file...” to the file "echo\_file.txt" `echo "new file .." > echo_file.txt`
    - Use the command 'echo' for this. First test out this command to see what it does
    - Check out the content of the file `cat echo_file.txt`
  - Append the text “Adding a second line to my text file” to the same file `echo “Adding a second line to my text file” >> echo_file.txt`
    - Check out the content of the file `cat echo_file.txt`
- Go to linuxcourse\_2024/data\_files/fastq
  - How many files are in this directory? Print the results on the screen `ls | wc -l`
  - For how many samples do we have \*.fastq data? `ls *.fastq | wc -l`
    - Use ls to first get the file names
    - Extract the sample from the file name somehow and remove duplicate sample names `ls | cut -d_ -f1`
  - Copy the file "Sample43\_S91\_L001\_I1\_001.fastq" to a file called "copied\_file.fastq"  
`cat Sample43_S91_L001_I1_001.fastq > copied_file.fastq`
    - Don't use the cp command but use output redirection to do the copy
    - Verify by checking the content `cat copied_file.fastq`

# Exercises

- Go to `linuxcourse_2024/data_files/VARIANT_CALLING_FILES`
  - Count the number of files
    - Write the result to the file "file\_count.txt" `ls | wc -l > ../file_count.txt`
    - Make it so that this file is made directly in your `linuxcourse_2024` folder
  - Get a size-sorted list of the files and print only the 3 largest files on the screen `ls -Slh | head -n 3`
    - Besides `ls`, can you find a Linux command that can calculate file sizes online? `du -h *`
- Go to `linuxcourse_2024/data_files/bed_files`
  - Get the last 1000 lines from the "large\_gene\_panel.bed" file and sort them on chromosome, and further sort on genomic start position `tail -n 1000 large_gene_panel.bed | sort -k1,1V -k2,2n`
    - Write the output to file "large\_file\_sorted.bed" in the same folder
    - `tail -n 1000 large_gene_panel.bed | sort -k1,1V -k2,2n > large_file_sorted.bed`
  - From "large\_gene\_panel.bed" select the exons corresponding to the AACS gene
  - `tail -n 1000 large_gene_panel.bed | sort -k1,1V -k2,2n > large_file_sorted.bed`
    - Write the genomic sorted output to file "AACS\_exons.txt" in the same directory
    - Are you sure you only selected AACS exons? If not, try to correct your output
    - On what chromosome is the AACS located? Write the solution to the screen on 1 line

# 7. Software installation & error handling

# Ways to install & use software

- **Imagine: experiment => fastq data**
- **You want to check the quality, don't know where to start**
  - Google

🔍 how to check quality of fastq data

- ChatGPT

What can I help with?

how can check the quality of my fastq data? it's in fastq.gz format, and I've never worked with it before.  
I have installed WSL on my windows PC



**Always double check!**

# Ways to install & use software

- **FastQC: A quality control tool for high throughput sequence data**
- **Download options:**
  - Package manager Linux (apt-get / dnf / yum ...)
  - Environment (Conda)
  - Container (Docker)
  - Manual (install binaries, add tool to \$PATH), least convenient but sometimes necessary
  - ...
- **! If you're using WSL: follow Linux installation instructions, not Windows**

# Ways to install & use software: apt get

- **Installation with Linux package manager**

- Installed on your “whole” system
- `$ sudo apt-get install fastqc`

- Why sudo?
- Apt-get can also be yum, dnf, ...

- **! Not possible on HPC**

→ Need sudo privileges

(administrator) which we (luckily) don't have 😊

```
elinet@ELinformatica:~$ sudo apt-get install fastqc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libaec0 libapache-pom-java libargs4j-java libcommons-compress-java libcommons-io-java
  libcommons-jexl2-java libcommons-lang3-java libcommons-logging-java
  libcommons-math3-java libcommons-parent-java libfindbin-libs-perl libhdf5-103-1
  libhtsjdk-java libjzip2-java libjson-simple-java libngs-java libngs-sdk-dev libngs-sdk2
  libsis-base-java libsis-base-jni libsis-jhdf5-java libsis-jhdf5-jni libsnappy-java
  libsnappy-jni libsnappy1v5 libsz2
Suggested packages:
  libxz-java libcommons-io-java-doc libcommons-lang3-java-doc libavalon-framework-java
  libcommons-logging-java-doc libexcalibur-logkit-java liblog4j1.2-java picard-tools
  libjson-simple-doc
The following NEW packages will be installed:
  fastqc libaec0 libapache-pom-java libargs4j-java libcommons-compress-java
  libcommons-io-java libcommons-jexl2-java libcommons-lang3-java libcommons-logging-java
  libcommons-math3-java libcommons-parent-java libfindbin-libs-perl libhdf5-103-1
  libhtsjdk-java libjzip2-java libjson-simple-java libngs-java libngs-sdk-dev libngs-sdk2
  libsis-base-java libsis-base-jni libsis-jhdf5-java libsis-jhdf5-jni libsnappy-java
  libsnappy-jni libsnappy1v5 libsz2
0 upgraded, 27 newly installed, 0 to remove and 93 not upgraded.
Need to get 9407 kB of archives.
After this operation, 16.2 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

# Ways to install & use software: conda

- **Conda allows for package and environment management**
  - Environment management: can create smaller environments within your system to download specific software in. Only when you're present in the environment, you can run the tools you downloaded.
  - Package management: easy to download specific versions of packages/software, needed for a specific project
  - Rule of thumb: for each new project, new conda environment
  - Use of environments is good practice, to keep your general space "clean": avoid unwanted dependency errors

# Ways to install & use software: conda

1. **Download conda (we're not doing this right now) only first time**
2. **Create environment**
3. **Activate environment**
4. **Download packages/tools**
  - Either you download the necessary tools one by one, by using  
`$ conda install <package name>`  
`$ conda install <package1> <package2>`
  - Or you give it a **file** to build an environment from (.yaml) → important for HPC
5. **Do your analysis**
6. **Deactivate environment**

# Ways to install & use software: conda (locally)

## ▪ Creating environment

```
elinet@ELinformatica:~$ conda create -n project_name
Retrieving notices: ...working... done
Channels:
 - conda-forge
 - bioconda
 - defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
  current version: 24.5.0
  latest version: 24.11.0

Please update conda by running

  $ conda update -n base -c conda-forge conda

## Package Plan ##

  environment location: /home/elinet/mambaforge/envs/project_name
```

```
Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate project_name
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

```
elinet@ELinformatica:~$ |
```



# Ways to install & use software: conda (locally)

- List existing envs, activate environment

```
elinet@ELinformatica:~$ conda env list
# conda environments:
#
base                /home/elinet/mambaforge
project_name        /home/elinet/mambaforge/envs/project_name
prokka              /home/elinet/mambaforge/envs/prokka
virtus              /home/elinet/mambaforge/envs/virtus
```

```
elinet@ELinformatica:~$ conda activate project_name
(project_name) elinet@ELinformatica:~$
```

# Ways to install & use software: conda (locally)

- Install the tool(s) you want to install
- **Channels:**
  - *Large repositories*
  - *Contain packages*
  - *E.g. bioconda repository, contains lots of interesting software for bioinformatics purposes*
- Check if command is available
- Done!

```
(project_name) elinet@ELinformatica:~$ conda install fastqc
Channels:
- conda-forge
- bioconda
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

==> WARNING: A newer version of conda exists. <==
current version: 24.5.0
latest version: 24.11.0

Please update conda by running

$ conda update -n base -c conda-forge conda

## Package Plan ##

environment location: /home/elinet/mambaforge/envs/project_name

added / updated specs:
- fastqc

The following packages will be downloaded:
```

package	build		
fastqc-0.12.1	hdfd78af_0	11.1 MB	bioconda
fontconfig-2.15.0	h7e30c49_1	259 KB	conda-forge

# Ways to install & use software: conda (HPC)

- **On HPC => conda discouraged, because it creates lots of small files**
- **Still need software?**
  - Check if it's already installed (in doubt, [contact CalcUA](#))
  - Not installed? Create a .yaml with your packages needed!

# Ways to install & use software: conda (HPC)

- **How to make a .yaml**
  - Create a new file
  - Type the following info:
    - Name of environment
    - Channels to look in (usually always these three)
    - Dependencies = packages = software you need
    - Can define version (see python)
  - Save file by naming it **name.yaml**

```
name: rnaseq_env
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
  - python=3.9 # Python version
  - fastqc # Quality control for raw sequencing reads
  - trimmomatic # Adapter trimming and quality filtering
  - hisat2 # RNA-seq alignment tool
  - samtools # Manipulation of SAM/BAM files
  - featurecounts # Gene-level quantification
  - star # Spliced aligner for RNA-seq data
  - salmon # Transcript-level quantification
  - r-base=4.2 # R for downstream analysis
  - r-essentials # R packages like ggplot2, dplyr, etc.
  - bioconductor-deseq2 # Differential expression analysis
  - multiqc # Aggregates QC reports
  - pandas # Python library for data analysis
  - matplotlib # Python library for plotting
  - numpy # Python library for numerical computing
```

# Ways to install & use software: conda (HPC)

- **How to make a .yaml**

- OR: build an environment with all your wanted software on your own PC

- **Then** export your environment to a yaml

```
$ conda env_name export > environment.yaml
```

- Then use this environment.yaml in the next step

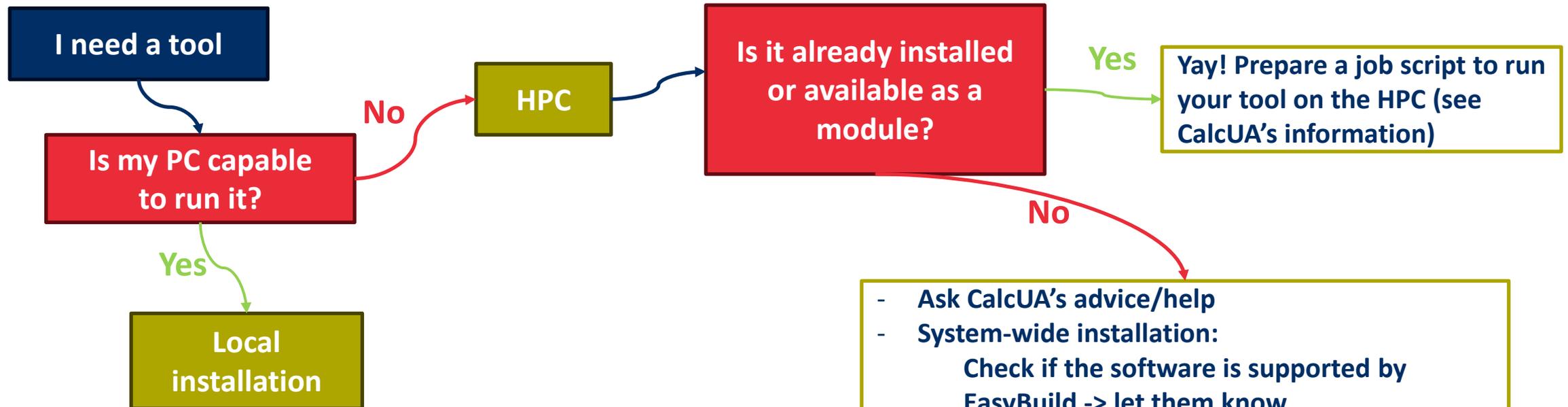
# Ways to install & use software: conda (HPC)

- This .yaml is then used to create a container using the hpc-wrapper-container command (see documentation CalcUA)
- **Container?!**
  - Even more separation of your software
  - Container = “box” where software is in “installed”
  - You tell your PC to use this “box” when you want to run the tool
  - No need to install it on the device: it’s installed in the box
  - You *pull* the container, then run the command
- **Conda\_env.yaml → container → then run your commands**
- **More HPC information: in CalcUA slides**

# Ways to install & use software: flowchart

- **How to download my needed software?**
  - FIRST: Google the tools you need, inspect the size of your data, ask AI, look for similar publications, compare alternatives, ...
  - Inspect the documentation of the tools
    - Popular tools: easily findable on Google, conda, docker (containers) etc.
    - Also often: GitHub pages → look for the README file: this should contain the information to install the tool you want to use
- **Determine what device you will use for your analysis**
  - Does your own PC suffice (enough memory, computational power, ...)?
  - Part of it on HPC or completely?

# Ways to install & use software: flowchart



## System wide:

```
$ sudo apt get install <name>
```

## In separated environment: conda

```
$ conda install <name>
```

```
$ conda env create -f environment.yaml
```

Use container (see documentation)

...

**Follow instructions in documentation of tools!**

# Manual installation (local)

- **Fastp (alternative to FastQC)**

- <https://github.com/OpenGene/fastp>
- Options to install:
  - Bioconda
  - Install binaries (?)

- Manual installation

```
# download the latest build  
wget http://opengene.org/fastp/fastp  
chmod a+x ./fastp
```

# Manual installation (local)

- **Fastp (alternative to FastQC)**

- Installation steps:

1. Download the tool (using wget)
2. Make the binary executable (meaning you give yourself the permission to run it)
3. Move the binary to a directory in your PATH

# Linux environment: special variables

## ▪ \$PATH

- List of locations where Linux search for programs
  - So when using cat, it knows where to search for this
- Is pre-defined but you can add paths to it
  - Locations in the list are separated with ":"
  - Order matters: first looks in first location, then second...

## ▪ You can change your \$PATH variable

- Just as another variable
- Use command export to set the path for the current session (else you overwrite it!)

```
localadmin@ngscoursevm:/home/training/data_files$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
localadmin@ngscoursevm:/home/training/data_files$ export PATH=$PATH:/home//training/
localadmin@ngscoursevm:/home/training/data_files$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home//training/
localadmin@ngscoursevm:/home/training/data_files$
```

# Manual installation (local)

- **Fastp (alternative to FastQC)**

- Installation steps:

1. Download the tool (using wget)
2. Make the binary executable (meaning you give yourself the permission to run it)
3. Move the binary to a directory in your PATH

Often included in the PATH:

- /usr/local/bin (preferred for user-installed binaries)

- /usr/bin (for system-wide binaries)

To move \$ `sudo mv ./fastp /usr/local/bin/`

4. Verify the installation (try to run the cmd)

# Error handling

- **Unfortunately, running programs is not always smooth**
- **Something goes wrong: error code**
  - Code 0 means no errors occurred
  - 1: general errors, minor problems
  - 2: severe errors
  - ... goes up until 255...
- **Get an error?**
  - Stay calm
  - **Read the error**: it often indicates a lot
  - Copy it and paste it into Google/AI

# Error handling

```
(base) elinet@ELinformatica:~/usr$ mkdir test  
mkdir: cannot create directory 'test': Permission denied
```

```
(base) elinet@ELinformatica:~$ rm project/  
rm: cannot remove 'project/': Is a directory
```

```
(base) elinet@ELinformatica:~/project/LeapSEQ_nf/data$ cd..  
cd..: command not found
```

# Error handling

```
(base) elinet@ELinformatica:~/project/LeapSEQ_nf/data$ fastqc *.pod5
Failed to process subset_PAK100.pod5
uk.ac.babraham.FastQC.Sequence.SequenceFormatException: ID line didn't start with '@'
    at uk.ac.babraham.FastQC.Sequence.FastQFile.readNext(FastQFile.java:158)
    at uk.ac.babraham.FastQC.Sequence.FastQFile.<init>(FastQFile.java:89)
    at uk.ac.babraham.FastQC.Sequence.SequenceFactory.getSequenceFile(SequenceFactory.java:106)
    at uk.ac.babraham.FastQC.Sequence.SequenceFactory.getSequenceFile(SequenceFactory.java:62)
    at uk.ac.babraham.FastQC.Analysis.OfflineRunner.processFile(OfflineRunner.java:159)
    at uk.ac.babraham.FastQC.Analysis.OfflineRunner.<init>(OfflineRunner.java:121)
    at uk.ac.babraham.FastQC.FastQCApplication.main(FastQCApplication.java:316)
Failed to process subset_plasmo100.pod5
uk.ac.babraham.FastQC.Sequence.SequenceFormatException: ID line didn't start with '@'
    at uk.ac.babraham.FastQC.Sequence.FastQFile.readNext(FastQFile.java:158)
    at uk.ac.babraham.FastQC.Sequence.FastQFile.<init>(FastQFile.java:89)
    at uk.ac.babraham.FastQC.Sequence.SequenceFactory.getSequenceFile(SequenceFactory.java:106)
    at uk.ac.babraham.FastQC.Sequence.SequenceFactory.getSequenceFile(SequenceFactory.java:62)
    at uk.ac.babraham.FastQC.Analysis.OfflineRunner.processFile(OfflineRunner.java:159)
    at uk.ac.babraham.FastQC.Analysis.OfflineRunner.<init>(OfflineRunner.java:121)
    at uk.ac.babraham.FastQC.FastQCApplication.main(FastQCApplication.java:316)
```

# Exercises part 7

# Running software

- **Download fastQC on your system (not with conda) (tip: apt-get)**
- **Run fastQC in data\_files/fastq, on all the fastq.gz files**

- Direct the output to a subdirectory called fastqc\_output
- Use the documentation of fastqc to find out how
- How do you select the .fastq.gz files only?
- What is your output? Is this useful?

```
mkdir fastqc_output
```

```
Fastqc -o fastqc_output/ data_files/fastq/*.fastq.gz
```

- **Download multiqc (not with conda)**

```
pip install multiqc
```

- **Run multiqc**

- Which input does it need? How do you run it? Why would you use this tool?
- `multiqc fastqc_output/.`

# 8. Bash scripting

# VSCoDe

- **To code/write programs effectively, we make use of code editors**
- **Enables a nice interface to edit scripts**
- **VSCoDe is widely used, convenient code editor**

# VSCode

- **Check installation VSCode**
- **Windows users: Connect to WSL**
- **Open data\_files folder**
- **Create new folder 'bash\_scripts'**

# Programming in bash

- **Bash = command processor**
  - Interprets all the commands you type
  - Linux terminal = program to interact with bash
  - So all this time, we've been using bash
- **Single commands versus ...**
  - One-liner scripts
    - Using pipes, seen previously
  - Bash scripts
    - For more advanced commands: sometime need to iterate a command
      - for loop

# Programming in bash

Bash script: ends with .sh

```
$ test.sh X
$ test.sh
1  #!/bin/bash
2
3  # Script metadata/comments
4  # Author: Your Name
5  # Date: YYYY-MM-DD
6  # Description: What the script does
7  # Usage: How to use the script
8
9  cd /home/elinet/linux_course/data_files
10 mkdir hello
11 cd hello
12 echo "This is a message" > message.txt
13
```

shebang

After the #: comment, not interpreted

Bunch of commands

Run the script by: `$ bash test.sh`

# Programming in bash

- **for loop**

- Do something (one or more commands) for a list of files/folders/variables

- Syntax when making a script:

```
for <AN_ITEM> in <A_LIST_OF_ITEMS>
```

```
do
```

```
    #do some stuff with the loop variable $<AN_ITEM>
```

```
done
```

- To use on command line, separating the different steps with semicolon ;

- In general: semicolon in bash works as a space: both separate commands

- **for** <AN\_ITEM> **in** <A\_LIST\_OF\_ITEMS>; **do** ...; **done**

```
training@ngscoursevm:~$ for NAME in Tycho Dale Philip; do echo "$NAME would like to help you learn Linux"; done
Tycho would like to help you learn Linux
Dale would like to help you learn Linux
Philip would like to help you learn Linux
training@ngscoursevm:~$
```

# Programming in bash: FOR loop

- `for <AN_ITEM> in <A_LIST_OF_ITEMS>; do ... ; done`
  - `<AN_ITEM>` = Loop variable
    - Can be any name. Name it to use in loop
  - `<A_LIST_OF_ITEMS>`
    - Can be a list of items
      - Tycho Dale Philip
      - Separated with a spaces
    - Can be a command
      - Need to enclose with backticks (`)



```
training@ngscoursevm:~$ for FILE in `ls /home/training/data_files/fastq`; do echo $FILE; done | head -5
Sample43_S91_L001_I1_001.fastq
Sample43_S91_L001_I1_001.fastq.gz
Sample43_S91_L001_I2_001.fastq
Sample43_S91_L001_I2_001.fastq.gz
Sample43_S91_L001_R1_001.fastq
training@ngscoursevm:~$
```

# Programming in bash: FOR loop

- **Loops or wildcards?**
  - Sometimes it's just a choice, previous example

```
localadmin@ngscoursevm:/home/training/data_files/fastq$ ls -1 | head -5
Sample43_S91_L001_I1_001.fastq
Sample43_S91_L001_I1_001.fastq.gz
Sample43_S91_L001_I2_001.fastq
Sample43_S91_L001_I2_001.fastq.gz
Sample43_S91_L001_R1_001.fastq
localadmin@ngscoursevm:/home/training/data_files/fastq$
```

- More advanced commands require loops

```
localadmin@ngscoursevm:/home/training/data_files$ cat sample_id.txt
002
004
008
045
localadmin@ngscoursevm:/home/training/data_files$ for SAMPLE in `cat sample_id.txt`; do find ID_SNP -name "${SAMPLE}.vcf"; done
ID_SNP/002.vcf
ID_SNP/004.vcf
ID_SNP/008.vcf
ID_SNP/045.vcf
localadmin@ngscoursevm:/home/training/data_files$
```

# Programming in bash: FOR loop

- **Loops or wildcards?**
  - Sometimes it's just a choice, previous example

```
localadmin@ngscoursevm:/home/training/data_files/fastq$ ls -1 | head -5
Sample43_S91_L001_I1_001.fastq
Sample43_S91_L001_I1_001.fastq.gz
Sample43_S91_L001_I2_001.fastq
Sample43_S91_L001_I2_001.fastq.gz
Sample43_S91_L001_R1_001.fastq
localadmin@ngscoursevm:/home/training/data_files/fastq$
```

- More advanced commands require loops

```
localadmin@ngscoursevm:/home/training/data_files$ cat sample_id.txt
002
004
008
045
localadmin@ngscoursevm:/home/training/data_files$ for SAMPLE in `cat sample_id.txt`; do find ID_SNP -name "${SAMPLE}.vcf"; done
ID_SNP/002.vcf
ID_SNP/004.vcf
ID_SNP/008.vcf
ID_SNP/045.vcf
localadmin@ngscoursevm:/home/training/data_files$
```

# Exercises part 8

# Exercises

## ▪ Exercise 1: introductory

You are given following bash script:

```
#!/bin/bash
echo "Welcome to Bash scripting!"
echo "Hello, ${USER}! Nice to meet you."
cd ~
echo "You have reached your destination: $PWD"
```

1. what does it do? It will print “welkom to bash scripting” in the terminal, then print “Hello, <username>! Nice to meet you. Change the directory to the /home directory and then print again in the terminal “You have reached your destination: met het absolute path van de home directory”
2. create a file and copy the script in there: `$ nano bash_script1.sh`
3. set the correct permissions: `$ chmod +x bash_script1.sh`
4. run the script: `$ ./bash_script1.sh`

# Exercises

## ▪ Exercise 2: intermediate

1. Write a bash script that creates a file with following contents:  
the user who created the file,  
the date when the file was created, and  
the directory where the file is created.

Format this information however you like. Give the file a unique name, so that it will not overwrite any of your own files ;)

2. Run you script
3. Add your current working directory (where your script is located! cd to the directory of your script if needed!) to the path: `$ export PATH=$PWD:$PATH`  
cd to another directory (e.g. your Desktop, somewhere where your script is NOT located)  
now you should be able to run your script by only typing the filename, not its entire filepath (eg: `bash_script2.sh`)

```
#!/bin/bash
# Get the username of the user running the
scriptuser=$USER
# Get the current working directory
current_dir=$PWD
# Get the current date and time
date_created=$(date)
# set a file name:
file_name=intro_linux_ex7.2.txt
# ALTERNATIVE, but more advanced: generate a unique
file name: file_name=""info_file_$(date +%s).txt""
# Write the information to the file
echo ""Created by: $user"" > ""$file_name""
echo ""Date created: $date_created"" >> ""$file_name""
echo ""Directory: $current_dir"" >> ""$file_name""
```

# Exercises

- **Exercise 3: advanced**

What is the meaning of \$1 in following bash script?

```
#!/bin/bash
```

```
# A script to count the number of characters in a command-line argument  
using wc
```

```
# Use wc to count the number of characters
```

```
char_count=$(echo -n "$1" | wc -c)
```

```
# Print the result
```

```
echo "The input '$1' has $char_count characters."
```

This is the first argument that you give while executing your script for example: \$ script argument, then argument will be used in the script

Can you think of some use cases for \$i?

# Exercises

- **Exercise 4: advanced**

What does this script do?

```
#!/bin/bash
while echo -n "enter number: "; read NUM
do
  if [ $NUM -eq $NUM ] 2>/dev/null; then
    :
  else
    echo " $NUM is not a number"
    continue
  fi
  if [[ $(( $NUM % 2 )) -eq 0 ]]; then
    echo " $NUM is an even number"
    continue
  fi
  echo " $NUM is an odd number"
  break
done
```

# Exercises on FOR loops

## ▪ Exercise 1

1. Write a Bash script that creates 10 empty files named file1.txt, file2.txt, ..., file10.txt in the current directory
2. Print a message to the terminal for each file created, confirming its creation.

Expected output

```
File file1.txt has been created.  
File file2.txt has been created.  
File file3.txt has been created.  
File file4.txt has been created.  
File file5.txt has been created.  
File file6.txt has been created.  
File file7.txt has been created.  
File file8.txt has been created.  
File file9.txt has been created.  
File file10.txt has been created.
```

# Exercises on FOR loops

## ▪ Exercise 2: Advanced

1. Create a directory called project.
2. Navigate into the project directory.
3. Create 5 files named file1.txt, file2.txt, ..., file5.txt.
4. Write the current date and time into each of the 5 files.
5. Copy all files with names starting with file into a new subdirectory called backup.
6. Rename all files in the backup directory by adding a .bak extension (e.g., file1.txt → file1.txt.bak).
7. Delete the original files (file1.txt to file5.txt) from the project directory.
8. Print a message indicating that the script completed successfully.



University of Antwerp  
Human Molecular Genetics  
Centre of Medical Genetics

**Thank you for your attention!**

**biomina**