# Introduction to Linux

**Armando Djiyou**

PhD candidate, The University of Douala

Local Trainer, H3ABionet

WORKSHOP ON BIOINFORMATICS APPLICATIONS IN GENOMICS SURVEILLANCE OF BACTERIAL ANTIMICROBIAL RESISTANCE
22-24 February 2023, Yaoundé, Cameroon

## Learning outcomes

- Understand the Linux file structure
- Understand the command line structure and learn basic commands
- Learn how to create, access files and directories and navigate through them
- Learn how to read files content and extract information from them
- Learn how to combine commands and redirect outputs
- Learn how to manage files permissions

# Course structure

Part I: Introduction to Linux and Unix and the command line

Part II: Manipulating and extracting information from files

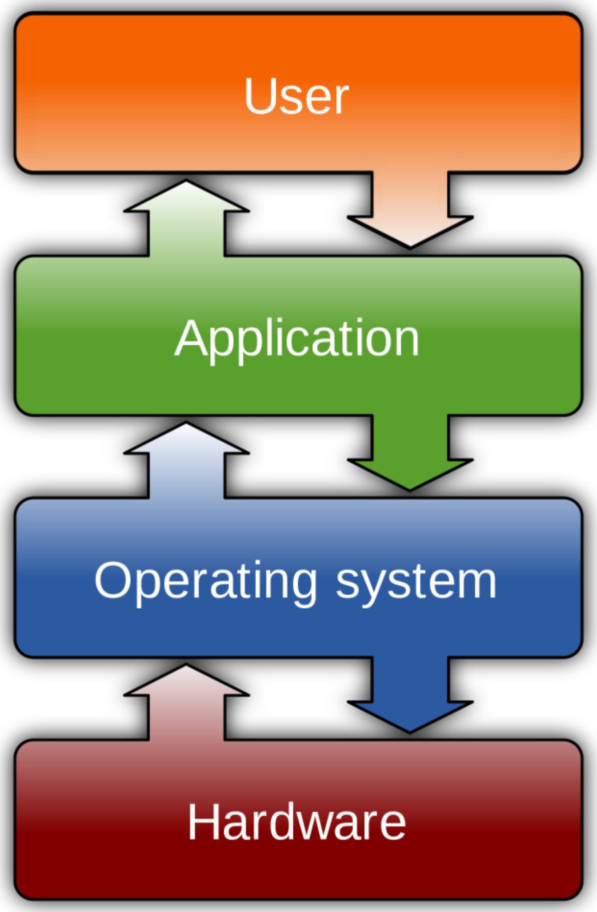
Part III: File permissions

Part IV: Practicals

## Part I:
## Introduction to Linux and Unix and the command line

# What is Linux?

- UNIX is an Operating System (OS) initially developed in the 1960
- OS: software that supports the computer's basic functions
- There are many different versions of UNIX, that share many similarities
- The most popular varieties of UNIX are Linux and MacOS
- UNIX systems have a graphical user interface (GUI) making easier the environment

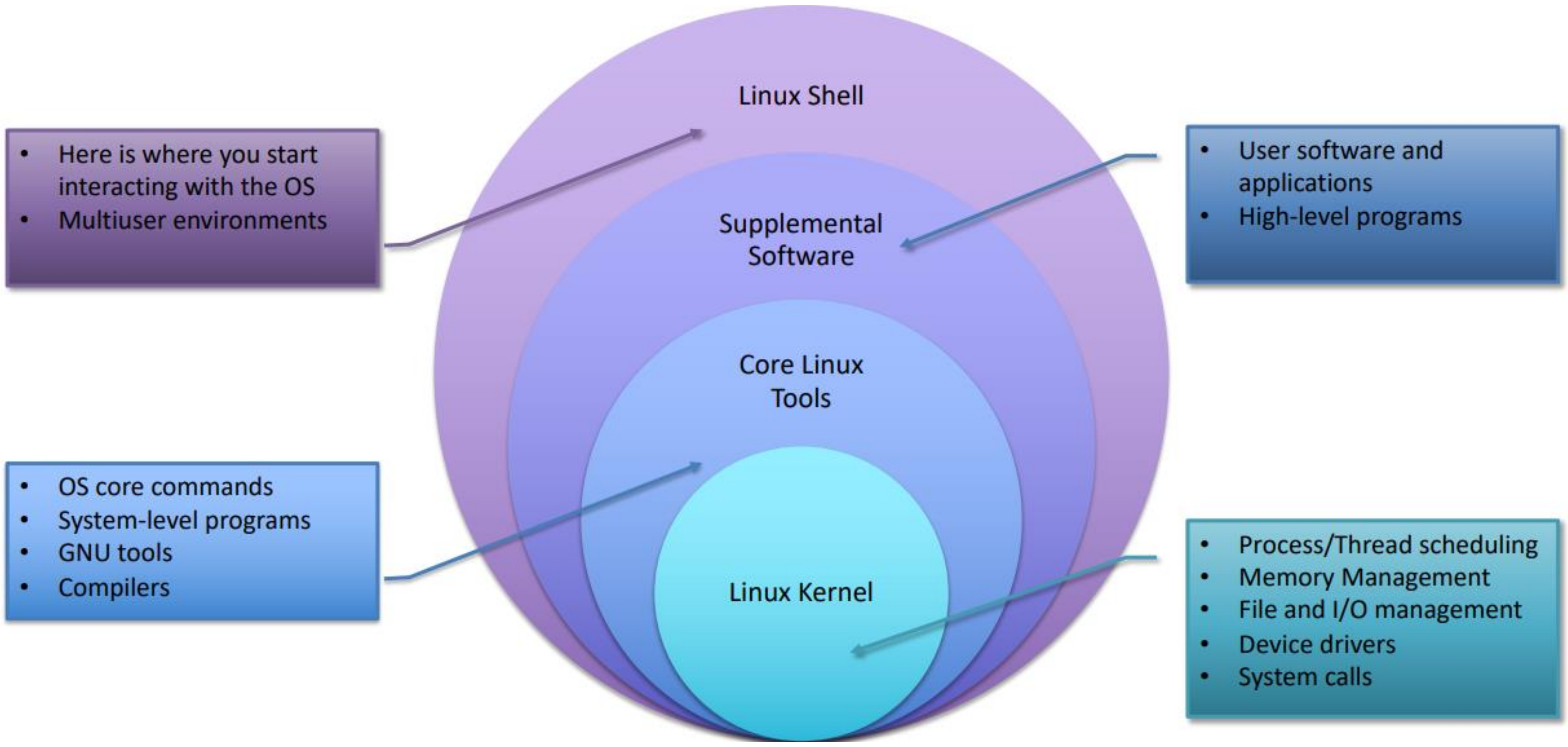


https://en.wikipedia.org/wiki/Operating_system

# Why Linux?

- Unix is particularly suitable for working with big files and has several powerful and flexible commands that can be used to process and analyse this data.
- One advantage of learning Unix is that many of the commands can be combined in an almost unlimited fashion
- Unix is the standard operating system on most large computer systems in scientific research, in the same way that Microsoft Windows is the dominant operating system on desktop PCs
- Linux is free and the most popular distributions are Ubuntu, Fedora/Red Hat, Mandriva, etc.
- Very stable, secure and fast developing OS (many developers)
- Best multi-user and multi tasking OS, this is why it is the preferred operating system for large-scale scientific computing.

## Structure of Linux



## Linux Shell and terminal

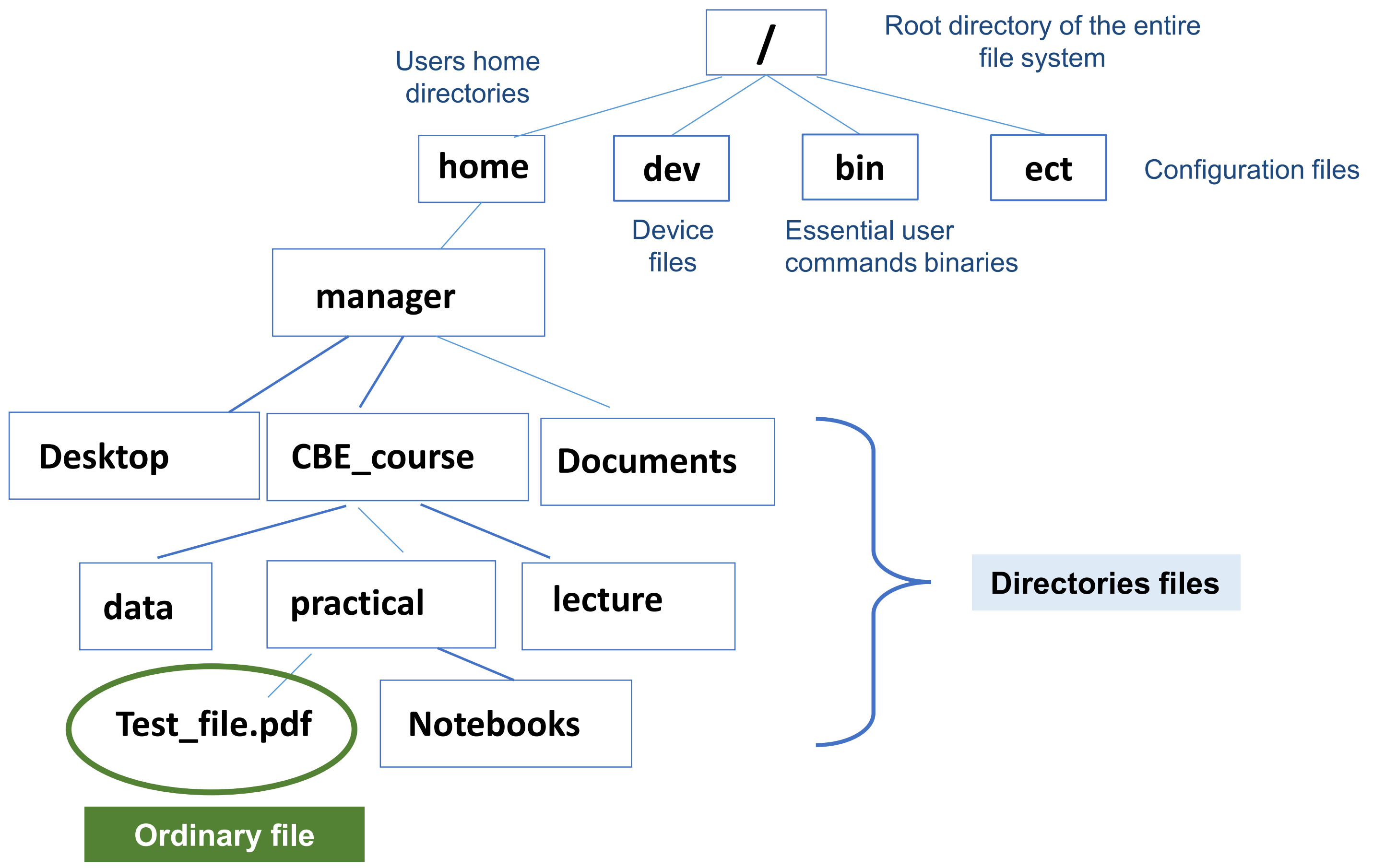
- **The shell** is a program that takes commands from the user’s keyboard and passes them to the operating system to execute
  - There are many different Unix shells
  - The most popular shell for interactive use include Bash: the default on most Linux installations

- **A terminal** refers to a wrapper program (the “black window”) that opens in a window and lets users interact with the shell.

- **The shell prompt** (or command line) is where one types commands
  - User name
  - System name
  - Name of the current directory (~)

# Working with files and directories

## Linux file structure



### Home directory

When you first log in on a UNIX system, the working directory is your **home directory**.

### Working or current directory

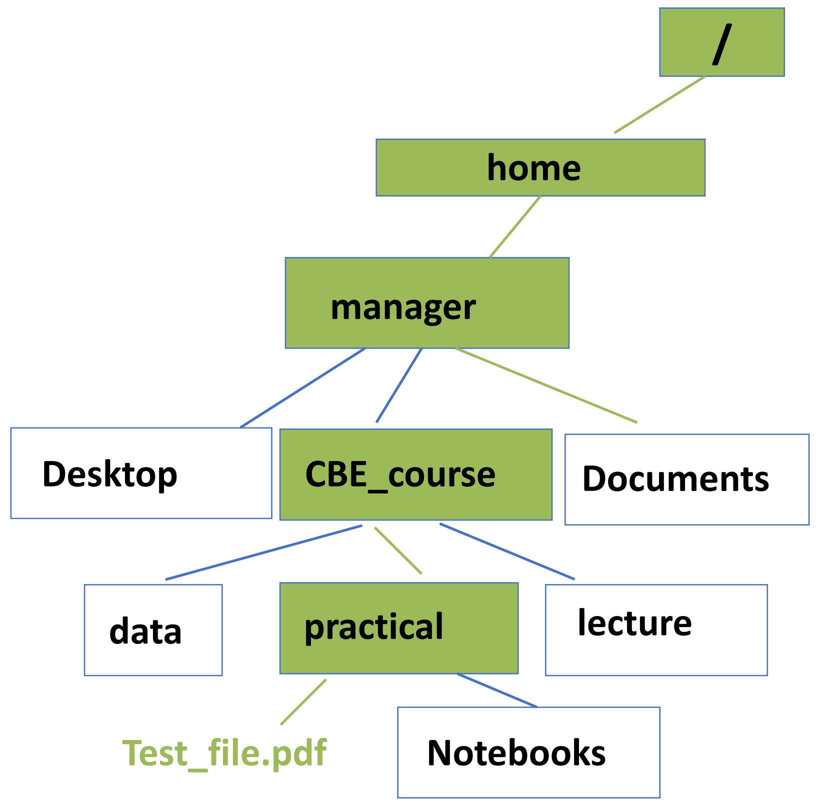
While working you will be associated to one directory called the **working directory or the current directory**

An abbreviation of the working directory is displayed as part of the prompt on your terminal

# Understanding path and pathname

## Absolute path:

File hierarchy starting from the root



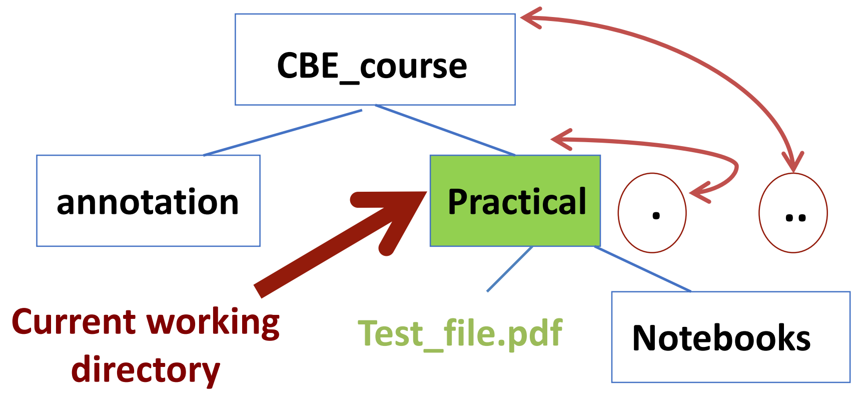
The full path to test_file.pdf is:

/home/manager/CBE_course/practical/Test_file.pdf

## Relative path:

File hierarchy starting from the current working directory

- . (dot): the current directory
- .. (dot-dot): the parent directory



The relative path to test_file.pdf is:

./Test_file.pdf

## Part II:
## Manipulating and extracting information from files in Linux

# Commands basic structure

## Command

A **command** is a program that you are running

## [–options]

**Option** tells a command how to operate and starts with a dash

## [arguments]

**The argument** tells a where to operate the command

| | | |
|---|---|---|
| ls | -lh | /home/manager/CBE_course/ |
| grep | -i “gene” | /home/manager/CBE_course/practical/Test_file.pdf |
| tail | -n | /home/manager/CBE_course/practical/Test_file.pdf |
| rm | -r | /home/manager/CBE_course/practical/ |

**Example:**

```
ls –lh /home/manager/CBE_course/
grep –i “gene” /home/manager/CBE_course/practical/Test_file.pdf
tail -5 /home/manager/CBE_course/practical/Test_file.pdf
```

# Key commands for handling directories

| Command | Description |
|---|---|
| pwd | Prints or displays the absolute path Current Working Directory |
| | Command structure: pwd |
| cd | Change directory: allows moving from one directory to another |
| | Command structure: cd <path><br>The path name of the directory you want to move to should be specified<br>You can specify either the absolute path or the relative path<br>Useful tips: cd without specifying any path move back to the home directory |
| ls | Lists a directory content |
| | Command structure: ls [OPTION] [dirname]<br>Some useful options:<br>• -l: shows sizes, modified date and time, file or folder name and owner of file and permissions<br>• -a: List all files including hidden file starting with '.'<br>• -lh: shows sizes in easier readable format<br>• -lS: sorting by file sizes |

# Key commands for handling directories

| Command | Description |
|---|---|
| mkdir | Make directory: creates a new directory |
| | Command structure: mkdir dirname [path]<br>mkdir dirname: would create a directory with the specified dirname<br>The newly created directory will be created in your current working directory<br>If you want to create it elsewhere, you have to specify the path: mkdir dirname path |
| rmdir/rm -r | Removes a directory |
| | Command structure: rmdir dirname [path]<br>It would remove the dirname directory only **if empty**<br>If the directory is in your current working directory, it is not necessary to specify the path<br>If the directory is not empty, you can use rm with option -r, which stands for recursive, that will recursively remove a directory and its contents |
| history | To see the command you have typed so far |
| | It allows you to see the command you have typed and save it in a separate file if needed |

## Basic manipulating file commands

| Command | Description |
|---|---|
| nano | It is a simple and easy-to-use text editor |
| | Command structure: nano filename [path]<br>Type the content of your file and once you finish typing, hit Ctrl+x to save and exit |
| cat | Concatenates and prints the contents of a short file |
| | Command structure: cat filename [path] |
| more | View the content of a long file and navigate through it |
| | Command structure: more filename [path] |
| less | View the content of a long file, by portions |
| | Command structure: less filename [path]<br>Move a page down: either use the page down key or space<br>To exit less, type q  and type g to go to the end of the text file |
| head/tail | View the first or the last lines of a long file |
| | Command structure: head <option> <filename><br>By default: 10 last lines at a time (use –n to change the number of lines you want to display) |

# Basic manipulating file commands

| Command | Description |
|---|---|
| cp | Copy files and directories |
| | Command structure: cp <pathfrom> <path to><br>Examples of application:<br>cp file1 file2 (Copy the contents of file1 into file2. If file2 does not exist, it is created)<br>cp filename dirpath (Make a copy of the file or directory into the specified destination directory) |
| mv | Moves or renames files and directories depending on how it is used |
| | Examples of applications:<br>mv -i filename1 filename2 (to rename a file; the -i will avoid overwriting file2 if it does exist)<br>mv filename dirpath (to move a file or directory to another directory)<br>mv file1 file2 file3 dirpath (to move different files or directories to another directory) |
| wc | Word counts: Prints new line, word, and byte counts for each file |
| | Command structure: wc <option> <filename><br>Some useful options:<br>-c: print the byte counts<br>-m: print the character counts<br>-l: print the newline counts |

# Extracting data from files

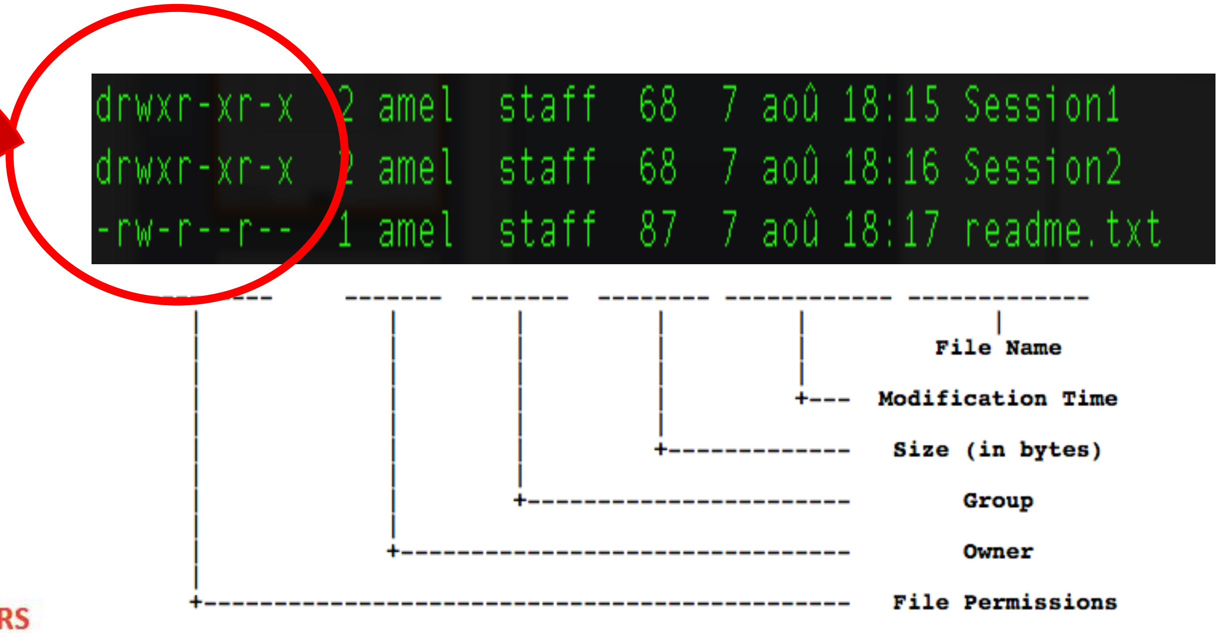
| Command | Description |
|---|---|
| grep | **G**lobal **R**egular **E**xpression **P**rofile" is used to search for the occurrence of a specific pattern (regular expression…) in a file |
| | grep outputs the whole line containing that pattern<br>Examples of application:<br>grep gene <filename> (Extract lines containing the term gene from a specific file)<br>grep -v gene <filename> (Extract lines that do not contain the pattern gene from a specific file)<br>grep -i gene <filename>  (Ignore case distinctions in both the PATTERN and the input files) |
| cut | Used to extract specific fields from a file |
| | Command structure: cut <option> <filename><br>Important options are<br>-d (field delimiter)<br>-f (field specifier)<br>Example:<br>cut -d' ' -f2,3 <filename> (extract fields 2 and 3 from a file having 'space' as a separator) |

## Part III:
## File permission in Linux

# File System Ownership and Permissions

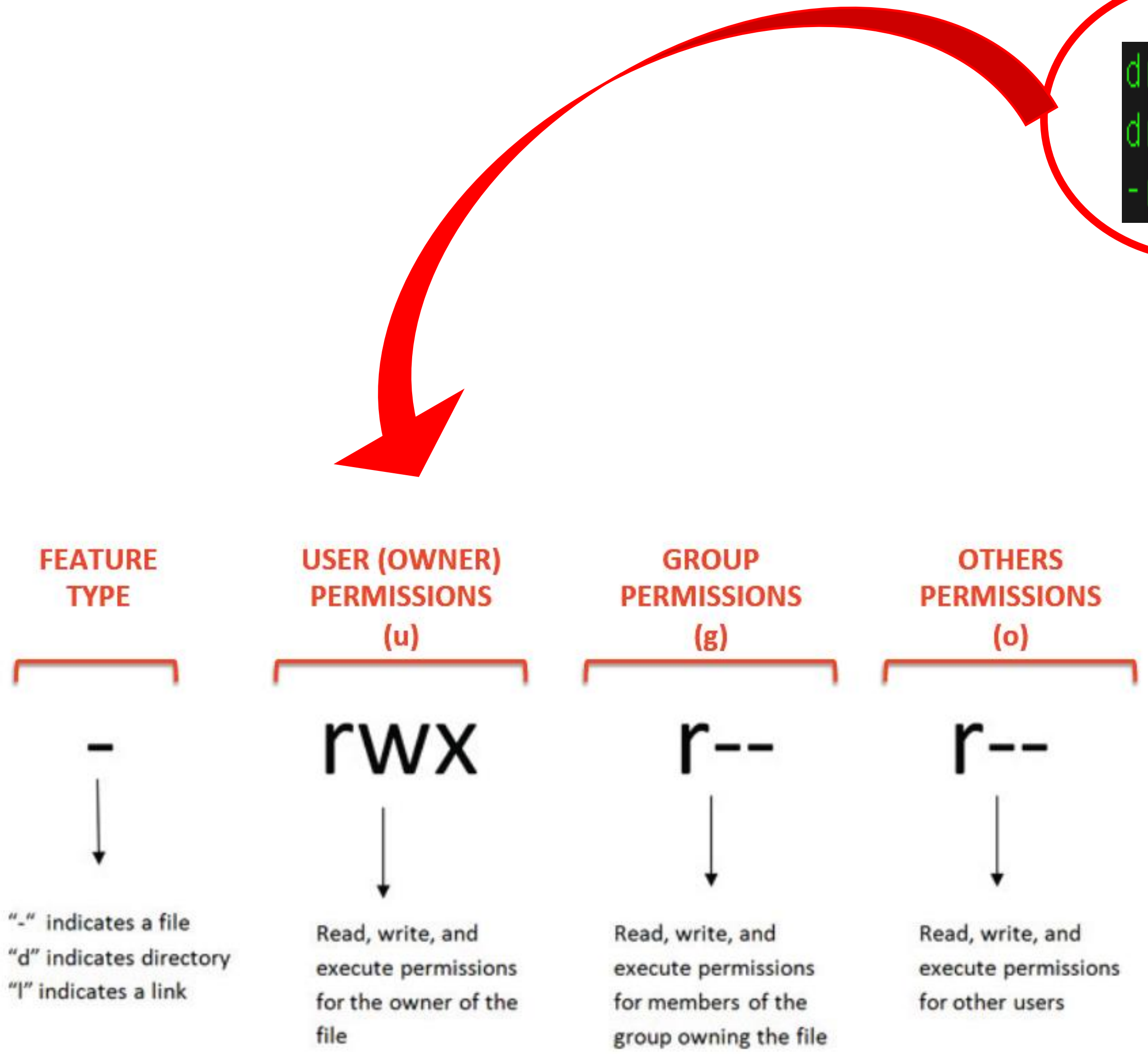
- All files and directories have individual and group *ownership*.
- All files and directories have **read** (**r**), **write** (**w**), and **execute** (**x**) *permissions* assigned as octets to the **individual** owner (u), the **group** (g) owner and all **others** (o) that are logged into the system.
- You can change permissions if you are the individual owner or a member of the group.
- Only the root user can change ownership.

# Description of file permissions



```
drwxr-xr-x  2 amel  staff  68  7 aoû 18:15 Session1
drwxr-xr-x  2 amel  staff  68  7 aoû 18:16 Session2
-rw-r--r--  1 amel  staff  87  7 aoû 18:17 readme.txt
```

Description of files and directories using ls -l


FEATURE TYPE
USER (OWNER) PERMISSIONS (u)
GROUP PERMISSIONS (g)
OTHERS PERMISSIONS (o)
-
rwx
r--
r--
"-" indicates a file
"d" indicates directory
"l" indicates a link
Read, write, and execute permissions for the owner of the file
Read, write, and execute permissions for members of the group owning the file
Read, write, and execute permissions for other users

Permissions are broken into 4 sections

## Access permissions on files

- r indicates read permission: the permission to read and and copy the file
- w indicates write permission: the permission to change a file
- x  indicates execution permission: the permission to execute a file, where appropriate

## Access permissions on directories

- r indicates the permissions to list files in the directory

- w indicates that users may delete files from the directory or move files into it

- x  indicates means the right to access files in the directory. This implies that you may read files in the directory provided you have read permission on the individual files

# chmod command

It is used to change the permissions of a file or a directory.

Syntax: chmod options permissions filename

| Symbol | Meaning |
|---|---|
| u | user |
| g | group |
| o | other |
| a | all |
| r | read |
| w | write (and delete) |
| x | execute (and access directory) |
| + | add permission |
| - | take away permission |

Example: chmod u=rwx,g=rx,o=r filename

Users can read, write and execute, the group to which the user belong can read and execute and the others can only read

## Few tips

- Use tab completion - it will save you time!
- Build commands slowly!
- man the_name_of_a_command often gives you help
- Always have a quick look at files with less or head to double check their format
- Watch out for data in headers and that you don’t accidentally grep some if you don’t want them
- If you did something smart but can’t remember what it was, try typing history
- Google is normally better at giving examples (prioritise stackoverflow.com results, they’re normally good)

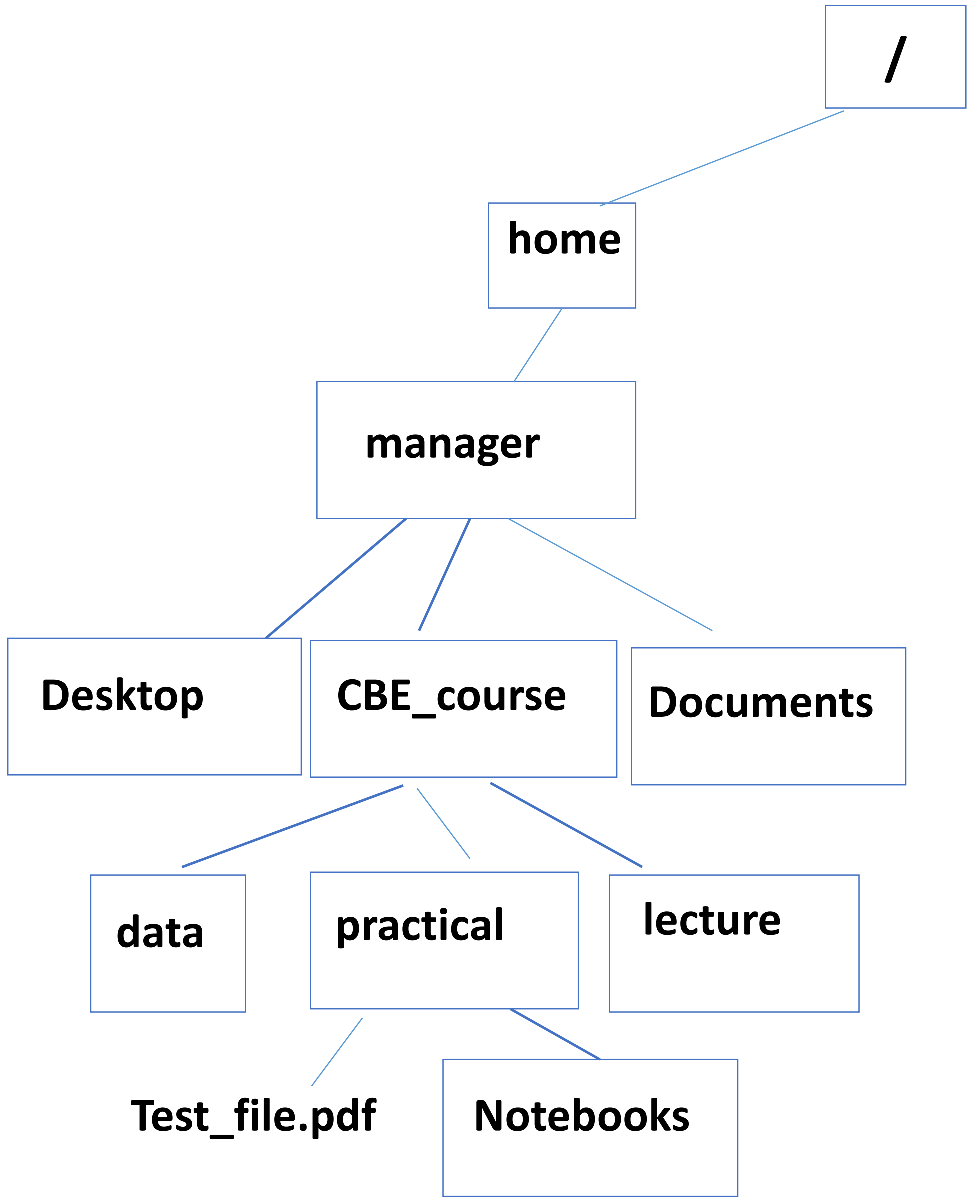
## Part IV:

## Practical

# Practical #1

1. Reproduce the file structure below on your computer
2. Return to your home directory
3. Create a new directory in **“practical”** and name it “exercise1”
4. Go into the “exercise1” directory
5. Create a new text file and name it “test1.txt”. Then add the following lines to the file:

   ***Exercise #1:***
   ***My name is xxxx (your name)***
   ***I am a xxxx (fonction) at (institution)***
   ***I am attending my first Linux course***
   ***Thank you to the CBE course team  for giving me this wonderful opportunity***
6. Now create a new directory under the “exercise1” directory and name it “subdir1”.
7. Go into the “subdir1” and try to print the content of “test1.txt” on your terminal.
8. Return to the “exercise1” directory and count the number of lines in “test1.txt”.



# Practical #2

1. Return to your home directory
2. Create a new directory in **“practical”** and name it “exercise2”
3. Go to the “exercise2” directory
4. Copy the “test1.txt” file from the “exercise1” directory to the “exercise2” directory
5. Make a new directory under “exercise2” and name it “subdir2”
6. Rename the file “test1.txt” to “test2.txt”
7. Move the “test2.txt” into the “subdir2” directory
8. Extract lines containing the pattern “**Linux**”. How many lines are outputted?
9. Extract lines containing the pattern **“I am”** and redirect the outpout to “grep.out” file
   - ***Tips: use the greater than “>” sign***

9. Count the number of lines containing the pattern “I am”. Try doing it in a single command
   - ***Tips: use the pipe “|” command to combine two commands in one (use the output of one command as the input of the next command)***

9. Remove everything under the “exercise2” directory

# Practical #3

1. Set the “exercise1” directory as your working directory
2. Try the following command in your virtual machine

```
ls –lh exercise1

chmod g+w test1.txt

ls –lh test1.txt

chmod ugo+x test1.txt

ls –lh test1.txt

chmod ugo+w subdir1

ls –ld subdir1

chmod a-w
```

3. What did you notice at each step?