

# Tecniche di programmazione in chimica computazionale

## Introduction to Linux

Emanuele Coccia

Dipartimento di Scienze Chimiche e Farmaceutiche

# Our goal

- Practical introduction to the use of [Linux](#)

# Our goal

- Practical introduction to the use of [Linux](#)
- Working using a [terminal](#)

# Our goal

- Practical introduction to the use of [Linux](#)
- Working using a [terminal](#)
- Managing [files](#) and [directories](#)

# Our goal

- Practical introduction to the use of [Linux](#)
- Working using a [terminal](#)
- Managing [files](#) and [directories](#)
- [Environment](#) variables

- Two physical/logical **layers**:
  - **hardware**: physical components
  - **software**: instructions (on a set of data) for a computer
- Originally, instructions written in the **physical format**
- **Idea**: write instructions according to rules close to the programmer

# Historical overview

- **50/60s**: the operating system (OS) written in assembly code, each computer has its own OS

# Historical overview

- **50/60s**: the operating system (OS) written in assembly code, each computer has its own OS
- **No communication** between computers



# Historical overview

- **50/60s**: the operating system (OS) written in assembly code, each computer has its own OS
- **No communication** between computers
- Dennis Ritchie and Ken Thompson invented the B and C language at the Bell Labs

# Historical overview

- **50/60s**: the operating system (OS) written in assembly code, each computer has its own OS
- **No communication** between computers
- Dennis Ritchie and Ken Thompson invented the B and C language at the Bell Labs
- High-level languages, **independent** on the hardware



# Historical overview

- They created a C-based OS
- **UNIX** was the name of the project

- They created a C-based OS
- **UNIX** was the name of the project
- Computers **communicate** each other because they share the same “language” (a part the kernel, see next slides)
- **70s and 80s**: UNIX not usable on microcomputers (like PCs), too slow

- Late 80s, early 90s: large diffusion of home computers

# Historical overview

- Late 80s, early 90s: large diffusion of home computers
- 1991: Linus Torvalds, student at the University of Helsinki, invented [Linux \(Unix-based OS\)](#)
- The Linux project has grown into a mature and powerful OS
- Linux is developed and distributed under the [GNU General Public License \(GPL\)](#)

- Late 80s, early 90s: large diffusion of home computers
- 1991: Linus Torvalds, student at the University of Helsinki, invented [Linux \(Unix-based OS\)](#)
- The Linux project has grown into a mature and powerful OS
- Linux is developed and distributed under the [GNU General Public License \(GPL\)](#)
- The GNU project ([www.gnu.org](http://www.gnu.org)) was founded in 1984 with the goal of developing high quality, free software
- GNU is a recursive acronym for "[GNU's Not UNIX](#)"

# Why using Linux?

Because Linux is:

- **stable**: high-level system



# Why using Linux?

Because Linux is:

- **stable**: high-level system
- **multiuser**: many users can work at the same time

# Why using Linux?

Because Linux is:

- **stable**: high-level system
- **multiuser**: many users can work at the same time
- **multitasking**: many processes contemporarily

# Why using Linux?

Because Linux is:

- **stable**: high-level system
- **multiuser**: many users can work at the same time
- **multitasking**: many processes contemporarily
- **free**: open source license (you can modify, copy, distribute, personalize the source code)

# Why using Linux?

Because Linux is:

- **stable**: high-level system
- **multiuser**: many users can work at the same time
- **multitasking**: many processes contemporarily
- **free**: open source license (you can modify, copy, distribute, personalize the source code)
- **portable**: applied on very different architectures

# Why using Linux?

Because Linux is:

- **stable**: high-level system
- **multiuser**: many users can work at the same time
- **multitasking**: many processes contemporarily
- **free**: open source license (you can modify, copy, distribute, personalize the source code)
- **portable**: applied on very different architectures
- **powerful** and **secure**

- Free software (licensed under GPL)

- Free software (licensed under GPL)
- You can download, modify, distribute software as a simple user
- You can fix bugs!

- Free software (licensed under GPL)
- You can **download, modify, distribute** software as a simple user
- You can **fix** bugs!
- The GNU project is an example of free software
- “If programmers **deserve to be rewarded** for creating innovative programs, by the same token they **deserve to be punished** if they restrict the use of these programs” (Richard Stallman)



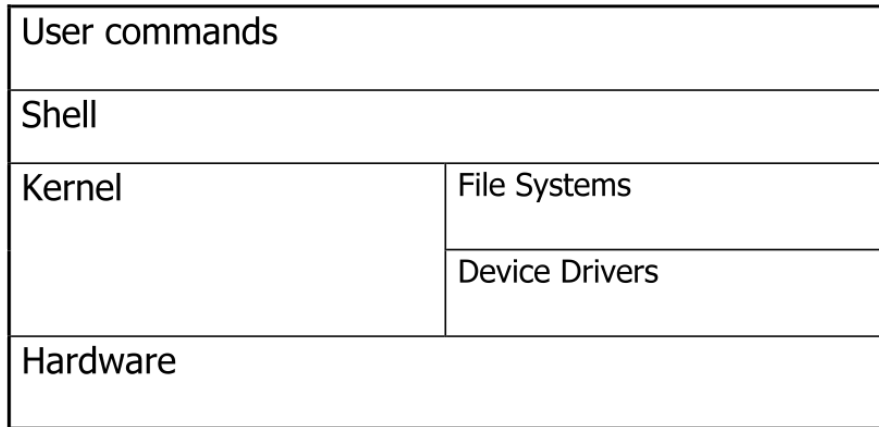
- Free software (licensed under GPL)
- You can **download, modify, distribute** software as a simple user
- You can **fix** bugs!
- The GNU project is an example of free software
- “If programmers **deserve to be rewarded** for creating innovative programs, by the same token they **deserve to be punished** if they restrict the use of these programs” (Richard Stallman)
- You can choose several distributions

- [Download](#) Linux from the Internet

- Download Linux from the Internet
- Every Linux distribution contains the basic packages

- Download Linux from the Internet
- Every Linux distribution contains the basic packages
- They all use the Linux kernel:
  - Ubuntu
  - Fedora
  - SuSe Linux
  - Debian
  - CentOS
  - ...
- Compatible each other

# The Linux system



- Heart of the system

# The kernel

- Heart of the system
- The kernel manages the communication between the hardware and the peripherals

- Heart of the system
- The kernel manages the communication between the hardware and the peripherals
- The kernel makes sure that processes and daemons (server processes) are started and stopped correctly



- Heart of the system
- The kernel manages the communication between the hardware and the peripherals
- The kernel makes sure that processes and daemons (server processes) are started and stopped correctly
- The kernel manages the hardware resources for the rest of the system

- The shell **interprets** user commands
- **Interface** between the OS and the user

- The shell **interprets** user commands
- **Interface** between the OS and the user
- Responsible for finding the commands and starting their execution

- The shell **interprets** user commands
- **Interface** between the OS and the user
- Responsible for finding the commands and starting their execution
- Several different shells are available

- The shell **interprets** user commands
- **Interface** between the OS and the user
- Responsible for finding the commands and starting their execution
- Several different shells are available
- **Bash** is popular and easy to use

- Devices are the way Linux talks to the world

- Devices are the way Linux talks to the world
- Devices are special files in the `/dev directory`

- Devices are the way Linux talks to the world
- Devices are special files in the `/dev` directory
- They manage the `peripherals` (keyboard, hard disk, mouse etc.)



- Devices are the way Linux talks to the world
- Devices are special files in the `/dev` directory
- They manage the `peripherals` (keyboard, hard disk, mouse etc.)
- They allow software to interact with a `device driver`

# The filesystem (I)

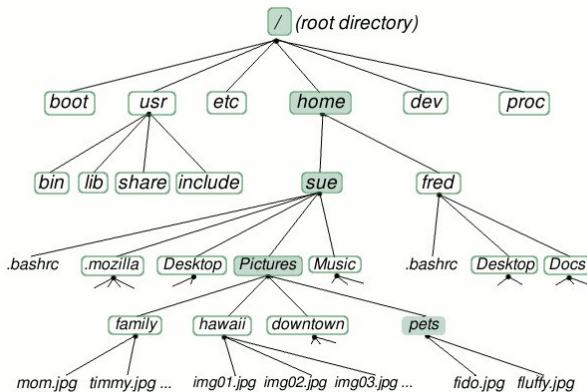
- “On a UNIX system, everything is a file; if something is not a file, it is a process”

# The filesystem (I)

- “On a UNIX system, everything is a file; if something is not a file, it is a process”
- Hierarchical filesystem

# The filesystem (I)

- “On a UNIX system, everything is a file; if something is not a file, it is a process”
- Hierarchical filesystem
- Only one root (/)



- Data files are stored in **directories** (folders)

# The filesystem (II)

- Data files are stored in **directories** (folders)
- Directories may be nested as deep as needed

# The filesystem (II)

- Data files are stored in **directories** (folders)
- Directories may be nested as deep as needed
- Files are named by naming each containing directory, starting at the root (**pathname**)

# The filesystem (II)

- Data files are stored in **directories** (folders)
- Directories may be nested as deep as needed
- Files are named by naming each containing directory, starting at the root (**pathname**)
- Directories are simply files containing other files



- `/home` → user directories

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software
- `/usr/bin` (or `/bin`) → system binaries, including the command shell

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software
- `/usr/bin` (or `/bin`) → system binaries, including the command shell
- `/usr/lib` (or `/lib`) → various libraries for processes

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software
- `/usr/bin` (or `/bin`) → system binaries, including the command shell
- `/usr/lib` (or `/lib`) → various libraries for processes
- `/etc` → system configuration files

# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software
- `/usr/bin` (or `/bin`) → system binaries, including the command shell
- `/usr/lib` (or `/lib`) → various libraries for processes
- `/etc` → system configuration files
- `/dev` → device files for all your peripherals



# The filesystem (III)

- `/home` → user directories
- `/proc` → kernel-processes pseudo file-system
- `/boot` → boot-up routines and the kernel
- `/usr` → user-oriented software
- `/usr/bin` (or `/bin`) → system binaries, including the command shell
- `/usr/lib` (or `/lib`) → various libraries for processes
- `/etc` → system configuration files
- `/dev` → device files for all your peripherals
- `/var` → various other files: mail, web server etc.
- `/opt` → extra software

- Your personal account for login

- Your personal account for login
- Use [Bitvise SSH client](#) to login into a Linux machine
  - Host: dscfalpha6.units.it
  - Username: your username

- Your personal account for login
- Use [Bitvise SSH client](#) to login into a Linux machine
  - Host: dscfalpha6.units.it
  - Username: your username
- To copy files locally, use [Bitvise SFTP](#)

# How to launch a Linux command

- Synopsis of any Linux command:  
command name **options (flags)** **arguments**

Everything is a file, we need to learn how to manage files  
To create and edit text files, various editors are available:

- nedit

Everything is a file, we need to learn how to manage files  
To create and edit text files, various editors are available:

- nedit
- gedit

Everything is a file, we need to learn how to manage files  
To create and edit text files, various editors are available:

- nedit
- gedit
- emacs



Everything is a file, we need to learn how to manage files  
To create and edit text files, various editors are available:

- nedit
- gedit
- emacs
- pico

Everything is a file, we need to learn how to manage files  
To create and edit text files, various editors are available:

- nedit
- gedit
- emacs
- pico
- vi (vim) → [vimtutor](#)  
Open a terminal and type [vimtutor](#)

- `pwd`: display the current working directory

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories
- `rm (-r)`: delete files and directories



# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories
- `rm (-r)`: delete files and directories
- `mv`: rename or move file and directories

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories
- `rm (-r)`: delete files and directories
- `mv`: rename or move file and directories
- `history`: chronology of commands

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories
- `rm (-r)`: delete files and directories
- `mv`: rename or move file and directories
- `history`: chronology of commands
- `exit (logout)`: close the actual session

# Basic commands

- `pwd`: display the current working directory
- `cd`: change directory
- `ls`: list of files and directories
- `mkdir`: create directories
- `cp (-r)`: copy files and directories
- `rm (-r)`: delete files and directories
- `mv`: rename or move file and directories
- `history`: chronology of commands
- `exit (logout)`: close the actual session
- `man`: manual and info of commands

- / → The root directory

# Special file names

- / → The root directory
- . → The current directory
- .. → The parent (previous) directory
- ~ → My home directory

- `more`: file perusal filter

- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement



- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement
- **head**: display first lines of a file

- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement
- **head**: display first lines of a file
- **tail**: display the last part of a file

- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement
- **head**: display first lines of a file
- **tail**: display the last part of a file
- **cat**: concatenate and print files

- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement
- **head**: display first lines of a file
- **tail**: display the last part of a file
- **cat**: concatenate and print files
- **paste**: merge corresponding or subsequent lines of files

# Managing files

- **more**: file perusal filter
- **less**: similar to more, but it allows backward movement in the file as well as forward movement
- **head**: display first lines of a file
- **tail**: display the last part of a file
- **cat**: concatenate and print files
- **paste**: merge corresponding or subsequent lines of files
- **grep**: find and display a string in a given file

- Redirect input <

# Special characters and key combinations

- Redirect input <
- Redirect output >

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character



# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character
- The Tab key: command or filename completion

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character
- The Tab key: command or filename completion
- Ctrl+C: kill a running process

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character
- The Tab key: command or filename completion
- Ctrl+C: kill a running process
- Ctrl+Z: stop running process

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character
- The Tab key: command or filename completion
- Ctrl+C: kill a running process
- Ctrl+Z: stop running process
- Ctrl+A: move the cursor to the beginning of the command line

# Special characters and key combinations

- Redirect input <
- Redirect output >
- The \* character
- The ? character
- The Tab key: command or filename completion
- Ctrl+C: kill a running process
- Ctrl+Z: stop running process
- Ctrl+A: move the cursor to the beginning of the command line
- Ctrl+E: move the cursor to the end of the command line

- `diff`: compares files line by line

# More commands I

- `diff`: compares files line by line
- `sdiff`: side-by-side merge of file differences



# More commands I

- **diff**: compares files line by line
- **sdiff**: side-by-side merge of file differences
- **zip** and **gzip**: packages and compresses (archive) files
  - zip name.zip file1 file2 etc.
  - unzip name.zip

# More commands I

- **diff**: compares files line by line
- **sdiff**: side-by-side merge of file differences
- **zip** and **gzip**: packages and compresses (archive) files
  - `zip name.zip file1 file2 etc.`
  - `unzip name.zip`
- **tar**: stores and extracts files from a tape or disk archive
  - `tar -czvf archive.tgz namedir` (`tar -cjvf archive.bz2 namedir`)
  - `tar -xvf archive.tgz` (`tar -xjvf archive.bz2`)

# More commands I

- **diff**: compares files line by line
- **sdiff**: side-by-side merge of file differences
- **zip** and **gzip**: packages and compresses (archive) files
  - `zip name.zip file1 file2 etc.`
  - `unzip name.zip`
- **tar**: stores and extracts files from a tape or disk archive
  - `tar -czvf archive.tgz namedir` (`tar -cjvf archive.bz2 namedir`)
  - `tar -xvf archive.tgz` (`tar -xjvf archive.bz2`)
- **find**: searches for files in a directory hierarchy
- **bc -l**: calculator

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file

# More commands II

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- `top`: displays Linux tasks

## More commands II

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- `top`: displays Linux tasks
- `ps`: lists your processes on the system
- `ps aux`: lists all the processes on the system

## More commands II

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- `top`: displays Linux tasks
- `ps`: lists your processes on the system
- `ps aux`: lists all the processes on the system
- `kill`: sends a signal to processes (only your own processes unless you are root)

## More commands II

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- `top`: displays Linux tasks
- `ps`: lists your processes on the system
- `ps aux`: lists all the processes on the system
- `kill`: sends a signal to processes (only your own processes unless you are root)
- `fg` and `bg`: foreground and background processes



## More commands II

- `wc`: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- `top`: displays Linux tasks
- `ps`: lists your processes on the system
- `ps aux`: lists all the processes on the system
- `kill`: sends a signal to processes (only your own processes unless you are root)
- `fg` and `bg`: foreground and background processes
- `whoami`: reports what user you are logged on

# More commands II

- **wc**: counts the number of bytes, characters, whitespace-separated words, and newlines in each given file
- **top**: displays Linux tasks
- **ps**: lists your processes on the system
- **ps aux**: lists all the processes on the system
- **kill**: sends a signal to processes (only your own processes unless you are root)
- **fg** and **bg**: foreground and background processes
- **whoami**: reports what user you are logged on
- **which**: locates a command

- The superuser `root` (the `su` command)

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities
- A user can belong to multiple groups

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities
- A user can belong to multiple groups
- Create users and groups (`useradd` and `groupadd`)

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities
- A user can belong to multiple groups
- Create users and groups (`useradd` and `groupadd`)
- `passwd`: change password



# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities
- A user can belong to multiple groups
- Create users and groups (`useradd` and `groupadd`)
- `passwd`: change password
- Files can be `read`, `written` and `executed`

# Users, groups and permissions

- The superuser `root` (the `su` command)
- Users (UIDs) are placed in groups, identified by group identifications (GIDs)
- Groups define functional areas/responsibilities
- A user can belong to multiple groups
- Create users and groups (`useradd` and `groupadd`)
- `passwd`: change password
- Files can be `read`, `written` and `executed`
- `chmod`, `chown` and `chgrp`

- **Environment variables**: global settings that control the function of the shell and other programs

# Environment variables

- **Environment variables**: global settings that control the function of the shell and other programs
- Variables as **PATH**, **HOME**, **SHELL** etc.
- **PATH**: lists directories of the shell search for the commands the user may type without having to provide the full path
- **HOME**: indicates where a user's home directory is located in the filesystem
- **SHELL**: indicates shell type

# Environment variables

- **Environment variables**: global settings that control the function of the shell and other programs
- Variables as **PATH**, **HOME**, **SHELL** etc.
- **PATH**: lists directories of the shell search for the commands the user may type without having to provide the full path
- **HOME**: indicates where a user's home directory is located in the filesystem
- **SHELL**: indicates shell type
- **Echo**: displays a line of text