2025/2026

Course: Open Source Software (OSS)

- Dr A. HADJI

Module Objectives:
To introduce

Introduce to the principles, tools, and practices of Free and Open Source Software (FOSS) to enable them to use, evaluate, and contribute effectively.

Content of the Module :

Chapter 2: Open Source Tools

2.1 Introduction (history, advantages / disadvantages, and licensing)

2.2 Development Environment (Introduction to Linux, Introduction to code editors)

2.3 Office Suite (Libre Office suite)

2.4 Collaboration (Storage and sharing)

2.5 Contributing to an open source project

Chapter 2: Open Source Tools

2.1. Introduction

- A free software is a computer program whose **source code** is accessible, modifiable, and freely redistributable.
  The term *free* comes from ***"freedom"*** rather than ***"free of charge"***.
- The **Free Software Foundation (FSF)**, founded by Richard Stallman in 1985, defines free software according to **four essential freedoms**:
  - **Freedom to run the program for any purpose.**
  - **Freedom to study how the program works (access to the source code is required).**
  - **Freedom to modify the program to suit one's needs.**
  - **Freedom to redistribute copies, with or without modifications.**
- Free software thus promotes **transparency, collaboration, and sharing**.

Chapter 2: Open Source Tools

2.1. Introduction

- The term **open source software** refers to a similar concept but emphasizes **collaborative software development and quality** rather than ethical or political freedom.

- The **Open Source Initiative (OSI)**, created in 1998, defined a set of **criteria** that software must meet to be considered open source — including access to the source code, free redistribution, and non-discrimination among users.

- Free software thus promotes **transparency, collaboration, and sharing.**

Chapter 2: Open Source Tools

2.1. Introduction

Avant le logiciel libre

- In the early days of computing, computers were extremely **expensive** and were distributed with software specifically designed for each machine, whose value was considered secondary.
- Programs were often modified, developed, and shared freely by their users, without any particular rules or restrictions.
- Eventually, software began to be recognized as having intrinsic value — it could be sold and was protected by **copyright law**.
- Microsoft rose to prominence through an agreement with **IBM**: IBM sold **personal computers (PCs)** equipped with Microsoft's **MS-DOS** operating system, over which Microsoft retained the rights.
- This marked the birth of the **software industry**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- There are also **obligations**, which vary depending on the license: GPL, BSD, Mozilla, X, etc.

- **not free**, **free of charge:**
  Internet Explorer, MacTCP, Acrobat Reader, freeware, etc.
- **not free**, **not free of charge:**
  no comment...
- **free (open source),** **free of charge:**
  Firefox, Linux, FreeBSD, OpenBSD, Android, OCaml, R, Scilab, VLC, sendmail, perl, etc.
- **free (open source),** **not free of charge:**
  commercial distributions of Linux, etc.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

The beginnings (1960s-1970s)

- In the early days of computing, software was freely shared among researchers and universities. Programs were considered **scientific collaborative tools**, not commercial products.
- **Unix :**
  Launched in **1969** at **AT&T's (Bell Labs)**, Unix was distributed to numerous industries and universities, which developed their own variants.
  Today, the descendants of Unix include:
- The **BSD family (Berkeley Software Distribution)**: **FreeBSD**, **OpenBSD**, **NetBSD**, and **Darwin**, the (open source) kernel of **macOS**.
  BSD also gave rise to the **BSD family of licenses.**
- **GNU/Linux** (*GNU = GNU's Not Unix*)

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

The closure of source code (1980s)

- As companies like Microsoft and IBM emerged, software became **proprietary**: source code was closed and distributed under restrictive licenses.
  This shift limited user and developer freedom.
- **The GNU Project**

By Aurelio A. Heckert, CC BY-SA 2.0

- Launched in **1983** by **Richard Stallman**, the GNU Project aimed to create a **completely free Unix-like operating system**.
-

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **The closure of source code (1980s)**
- **The GNU Project**

Definition

- The **GNU Project** is a **free software initiative** launched in **1983** by **Richard Stallman** at the Massachusetts Institute of Technology (MIT).
  The name **GNU** is a **recursive acronym** that stands for **"GNU's Not Unix"** — meaning that GNU was designed to be similar to Unix but entirely **free and open**.
- The main goal of the project was to develop a **complete, Unix-compatible operating system** made entirely of **free software**, allowing users the freedom to **use, study, modify, and share** all its components.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

2.2.1 The closure of source code (1980s)

The GNU Project

Objectives of the GNU Project

The GNU Project was created to:

- **Ensure users' freedom** to control their computers and software.
- **Replace proprietary Unix systems** with a completely free alternative.
- **Build a community** of developers and users who collaborate transparently.
- **Encourage knowledge sharing** and discourage software monopolies.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

2.2.1 The closure of source code (1980s)

X Window System (X11)

By Sven, CC BY-SA 3.0

- Launched in **1984** at the **Massachusetts Institute of Technology (MIT)**, the **X Window System**, also known as **X11**, is a windowing system widely used on **Unix** and **Unix-like** operating systems. It was distributed under the **X11 license**, which later inspired the well-known **MIT License**. Since **2004**, the system has been managed by the **X.Org Foundation**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **The Linux era (1990s)**
- In 1991, **Linus Torvalds**, a Finnish student, developed the **Linux kernel**, releasing it under the **GPL (General Public License)**.
- The combination of GNU software and the Linux kernel produced **GNU/Linux**, now powering millions of servers, computers, and mobile devices.
-

By Larry Ewing, Simon Budig, Garrett LeSage, CC0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

The Linux era (1990s)

- Launched in **1991** by **Linus Torvalds**, the **Linux kernel** was the **first free software project** to be developed collaboratively over the **internet** — initially through the exchange of patches via **newsgroups**, and later by **email**. It has been distributed under the **GNU General Public License (GPL) version 2.0** since **1992**.
- The development model of Linux inspired **Eric Raymond's** famous essay, *"The Cathedral and the Bazaar"*, which analyzed the benefits of open and decentralized software development.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**

- Projects like **Firefox, Apache, LibreOffice, Android, Git, and Python** demonstrated that open source software can be **robust, efficient, and competitive** with commercial solutions.
- Governments, universities, and companies increasingly adopt open source tools for **economic, technical, and strategic** reasons.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**
- **NCSA HTTPd and the Birth of the Apache Web Server**
- **NCSA HTTPd** was one of the **first web servers** and the **most widely used** in the world between **1993 and 1995**. It was developed at the **University of Illinois at Urbana-Champaign** by **Robert McCool**.

By The Apache Software Foundation and Vulphere, Apache License 2.0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**
- When the project was no longer maintained, its users decided to create a **fork** by combining the various patches that had been circulating in the community.
- The new project was humorously named **“Apache”**, a play on words from **“a patchy server.”**
- The **Apache Web Server** went on to become the **most widely used web server in the world** from **1995 to 2016.**
- This success led to the creation of the **Apache Software Foundation** and the establishment of the **Apache License 2.0.**

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**
- **Netscape and the Birth of Mozilla Firefox**
- **Netscape** was the **dominant web browser** between **1995 and 1997**, until **Internet Explorer** took over the market by being **bundled with Microsoft Windows.**
- In **1999**, the **Netscape company** released the **source code** of its browser, which became the foundation of the **Mozilla project** — later evolving into **Mozilla Firefox.**

By Mozilla Corporation, MPL 2.0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**

- **Firefox** became a **serious competitor** to Internet Explorer until **Google Chrome** entered the market and eventually dominated.
- The **Mozilla Foundation** was established in **2003**, and today it is also known for other major projects, including the **Rust programming language**.
- The project also gave rise to the **Mozilla Public License (MPL 2.0)**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**

From StarOffice to LibreOffice

- **Star Division**, a **German company** that developed the **StarOffice suite**, was **acquired in 1999** by **Sun Microsystems**.
- In **2000**, **Sun Microsystems** released the **source code** of the suite under an open-source license, giving rise to **OpenOffice**.

By Christoph Noack, CC BY-SA 3.0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**

From StarOffice to LibreOffice

- In **2005**, the **OpenDocument** format was established as an **international standard** for office documents.
- When **Oracle** acquired **Sun Microsystems** in **2009**, concerns about the project's future led the community to create a **fork** in **2010**: the **Document Foundation**, which launched **LibreOffice** — a fully community-driven office suite.

By Christoph Noack, CC BY-SA 3.0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**
- **Android: The Open Source Mobile Operating System**
- **Android** is an **open-source operating system** based on the **Linux kernel**, developed by **Google** and first released in **2007**.
- It was designed to **compete with Apple's iPhone**, and today it powers **over 70% of the global smartphone market**.

By Google, CC BY-SA 3.0

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**
- **Android: The Open Source Mobile Operating System**
- Although Android is **not developed in a fully collaborative or open manner**, its **license allows smartphone manufacturers** to create and distribute their **own modified versions** of the system.
- In practice, however, **most Android smartphones are not truly free**, since they include **Google Mobile Services (such as Google Play)**, which are **proprietary components**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**
- **SourceForge: One of the First Open Source Forges**
- Launched in **1999**, **SourceForge** was one of the **first platforms (forges)** dedicated to **hosting open-source software projects.**
- It provided developers with a range of essential tools such as **code repositories**, **bug trackers**, **wikis**, **mailing lists or forums**, and **download pages.**

By Slashdot Media LLC, Public Domain

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

Expansion to the public (2000s-2020s)

SourceForge: One of the First Open Source Forges

- SourceForge greatly **simplified the sharing and maintenance of small open-source projects**, including **minor forks of unmaintained software.**
  Over time, the platform hosted **more than 500,000 projects.**
- However, in **2013**, the site **lost credibility** due to **questionable practices**, leading many developers to **migrate their projects to GitHub.**

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**
- **git**
- The **Linux project** initially adopted a **decentralized but proprietary** version control system called **BitKeeper**. Although its license allowed free use for open-source development, this decision sparked **controversy** within the community, and the license was eventually **revoked**.
- In response, **Linus Torvalds** created **Git** in **2005** as a **free and open-source alternative** to BitKeeper — one that would fully support **distributed, collaborative software development**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s-2020s)**
- **git**
- Traditional version control systems such as **CVS** and **SVN** were **centralized**, meaning they required a **central repository**, direct **access to that repository**, and sometimes even **file locking mechanisms**.
  This structure was **not well suited** to the **open and decentralized collaboration model** typical of **open-source development**.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Expansion to the public (2000s–2020s)**
- **GitHub**
- **GitHub** was launched in **2008** by **four developers in California** as a **Git-based forge** for hosting and managing software projects.
- GitHub quickly became the **leading platform for open-source projects**, while also being widely adopted by **companies** to host and manage their **internal repositories**. Today, the platform hosts **over 200 million projects**, making it one of the **largest development ecosystems in the world**.

By GitHub, Public Domain

Chapter 2: Open Source Tools

2.1. Introduction

Advantages of Open-Source Software

Key Advantages of Open Source Software

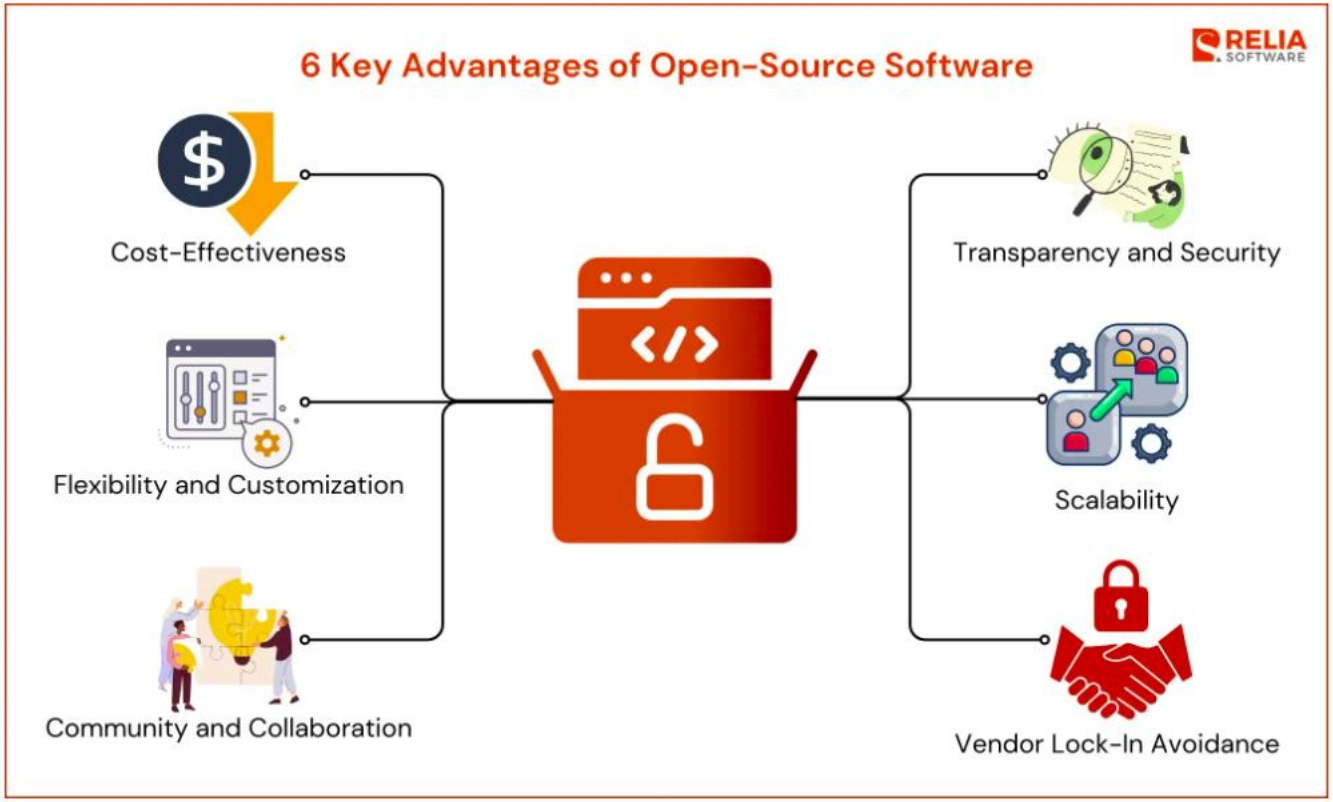
- **Cost-effectiveness**: eliminates licensing fees and reduces maintenance expenses through community support.
- **Flexibility**: full customization of the source code to meet specific organizational needs.
- **Community collaboration**: continuous improvement driven by global developer contributions.
- **Transparency and security**: open code enables faster detection and correction of vulnerabilities.
- **Scalability**: easily adapts to growth and changing requirements without additional costs.
- **Independence**: prevents vendor lock-in and ensures long-term technological freedom.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Advantages of Open-Source Software**



Chapter 2: Open Source Tools

2.1. Introduction

2.1.1 Historical Background of Free Software

- **Diadvantages of Open-Source Software**
- **Lack of dedicated support**: open source projects often lack professional, on-demand support; community help may be slow or unreliable during critical issues.
- **Hidden costs**: despite being free to download, OSS may require expenses for training, customization, maintenance, or hiring external developers.
- **Compatibility and integration issues**: some open source tools may not integrate smoothly with proprietary systems, leading to additional development and implementation costs.

Chapter 2: Open Source Tools

2.1. Introduction

Diadvantages of Open-Source Software

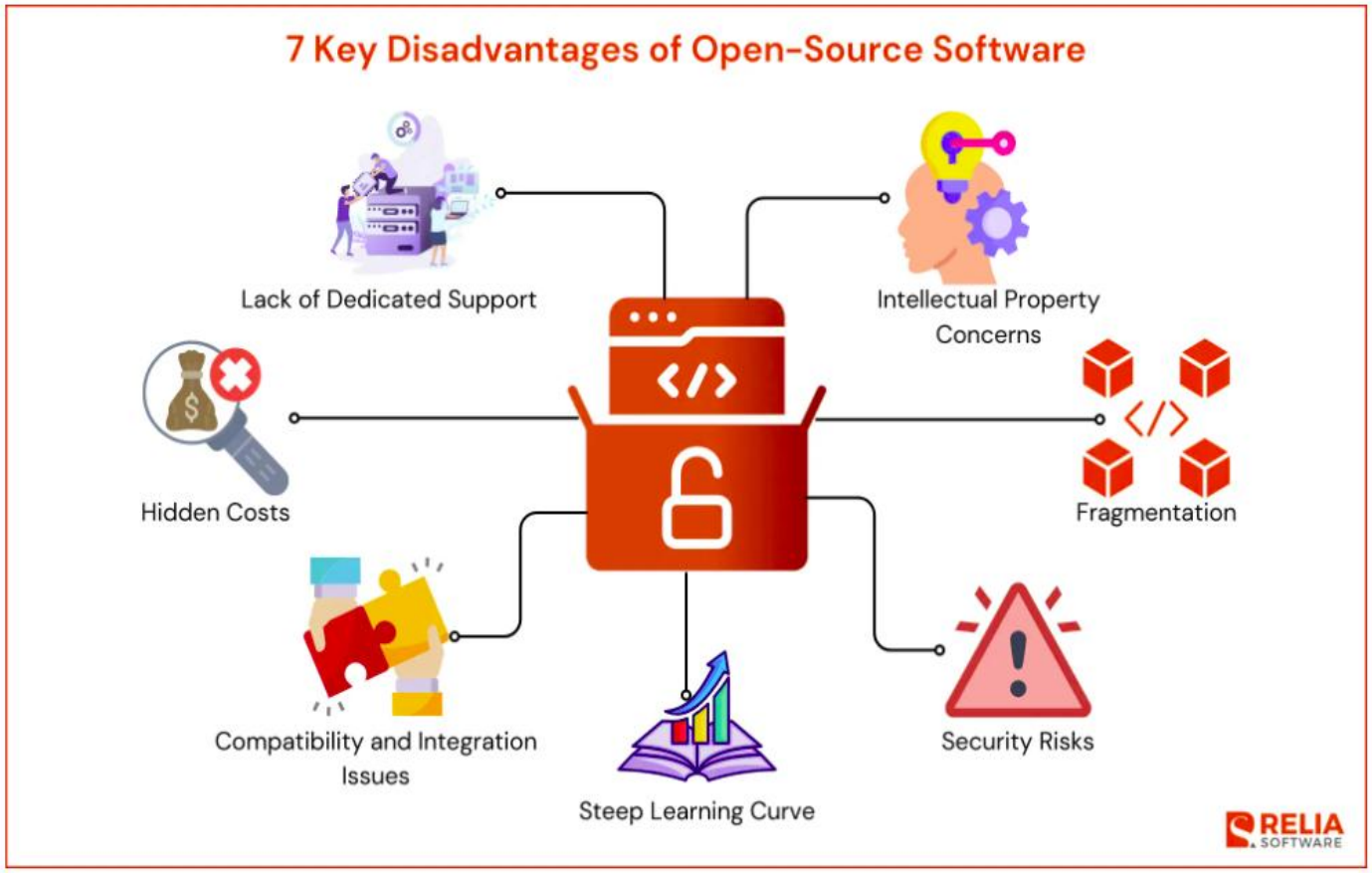
- **Steep learning curve**: limited documentation and less user-friendly interfaces can make adoption and training more difficult for non-technical users.
- **Security risks**: open access to source code can expose vulnerabilities to malicious actors if not patched promptly; community-based security is not always guaranteed.
- **Fragmentation**: decentralized development may result in multiple versions or forks, causing inconsistencies in support, documentation, and long-term stability.
- **Intellectual property concerns**: certain licenses (e.g., GNU GPL) require derivative works to remain open source, which can conflict with proprietary business models.

❖ Chapter 2: Open Source Tools

❖ 2.1. Introduction

❖ 2.1.1 Historical Background of Free Software

- **Diadvantages of Open-Source Software**



Chapter 2: Open Source Tools

2.1. Introduction

2.1.2 Licensing

- In the world of software, a **license** is a legal document that defines how a program can be used, modified, and shared.
- Open Source and Free Software licenses are designed to **protect user freedoms**, while **proprietary licenses** focus on **restricting usage and redistribution**.
- Free Software = Freedom of **use, study, modification, and sharing**.
  These freedoms are protected through specific **licenses** like the GPL.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.2 Licensing

- **Main Types of Open Source Licenses**
- There are several families of licenses in the open-source ecosystem, each with its philosophy and level of restriction.
- **1. The GNU General Public License (GPL)**
- Created by **Richard Stallman** for the **GNU Project**.
- The GPL ensures that all **derivative works** remain **free and open source**.
- If you modify and redistribute a GPL program, you must **publish your modifications** under the **same GPL license**.
- This is called the **"copyleft" principle** — it preserves freedom by making it *viral*.
- **Example:**
  Linux, GCC, and GIMP are distributed under GPL.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.2 Licensing

- **2. The Lesser General Public License (LGPL)**
- A **weaker** version of the GPL, created for **software libraries.**
- It allows proprietary software to **link** to LGPL libraries **without being forced** to become open source.
- Balances openness and compatibility with commercial software. **Example:** The GNU C Library (*glibc*) uses the LGPL.
- **3. The Apache License**
- Created by the **Apache Software Foundation**.
- Permits modification, redistribution, and use (even commercially).Does **not require** derivative works to be open source.
- Includes a **patent clause** to protect users against patent litigation.**Example:** Apache HTTP Server, Android, and Spark use the Apache 2.0 license.

Chapter 2: Open Source Tools

2.1. Introduction

2.1.2 Licensing

4. The MIT License

- A **very permissive** and **simple** license.
- Allows anyone to use, copy, modify, merge, or distribute the code.
- Requires only that the **original copyright notice** be preserved.
- Common in academic and startup projects.
- **Example:** jQuery, React, and Ruby on Rails use the MIT license.
- **5. The BSD License**
- Originating from the **Berkeley Software Distribution (BSD)** Unix system. Similar to MIT: very permissive, minimal conditions.
- Promotes **wide reuse** in both open-source and proprietary software.**Example:**FreeBSD, OpenBSD, and macOS (Darwin) are BSD-licensed.

Chapter 2: Open Source Tools

- **2.2. Development Environment** (, Introduction to code editors)

- **2.2.1 Introduction to Open-Source Environments**

In the world of modern computing, **free and open environments** occupy an essential place. They embody a philosophy based on **sharing**, **collaboration** and the **freedom** to use software without **restriction.**

- Unlike proprietary environments, which limit access to the source code and restrict user rights, **free environments offer users the ability** to **use**, **study**, **modify** and **redistribute** software according to their needs.
- Free environments are not limited to standalone programs; rather, they form a complete ecosystem that brings together :
  - *A free operating system, such as GNU/Linux;*
  - *Free applications (office tools, multimedia, .);*
  - *An active community of users and developers*

(who ensure the maintenance, documentation, and continuous evolution of these tools.)

Chapter 2: Open Source Tools

- **2.2. Development Environment** (, Introduction to code editors)
- **2.2.1 Introduction to Open-Source Environments**

2.1.2 Introduction to Linux

- In the strict sense, Linux refers to **the kernel of a free**, **multitasking**, **multi-platform** and **multi-user** operating system created by **Linus Torvalds**. It is commonly known as the **Linux kernel**.
- By 1991, the GNU project already offered a large collection of free software tools, but it lacked a crucial component: **the kernel.**
- This component is fundamental, as it manages memory, the processor, and hardware devices such as thekeyboard, mouse and hard drives.

Chapter 2: Open Source Tools

2.2. Development Environment

2.1.2 Introduction to Linux

- Linux is a free and open-source operating system (OS). Due to its stability, flexibility, and performance, it has become one of the most widely used systems worldwide. It is commonly used in:
  - servers and data centers,
  - supercomputers,
  - embedded systems (Android, routers, IoT devices),
  - Cyber security environments,
  - Cloud platforms and DevOps infrastructures,
  - Increasingly by software developers.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.2 Introduction to Linux

Why learn Linux?

- Learning Linux is essential because it is:
- **free and open-source**,
- **stable and secure**, even under heavy workloads,
- **widely used in enterprises and cloud environments**,
- **a core skill** for:
  - system and network administration,
  - DevOps and CI/CD,
  - artificial intelligence and distributed computing,
  - cybersecurity and digital forensics.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.2 Introduction to Linux

Structure of a Linux System

- **2.1. The Kernel**
- The ***kernel*** is the core of the Linux operating system. It is responsible for:
  - managing processes and scheduling,
  - managing memory,
  - interacting with hardware drivers,
  - Handling file systems,
  - Enabling communication between applications and hardware resources.
- In summary:

  Linux = Kernel + system tools + applications.

  - .

Chapter 2: Open Source Tools

2.2. Development Environment

2.2.2 Introduction to Linux

Structure of a Linux System

2.3. The File System

- Linux uses a hierarchical file system that starts at the root directory:
  - ‘’/ ‘’

Key directories include:

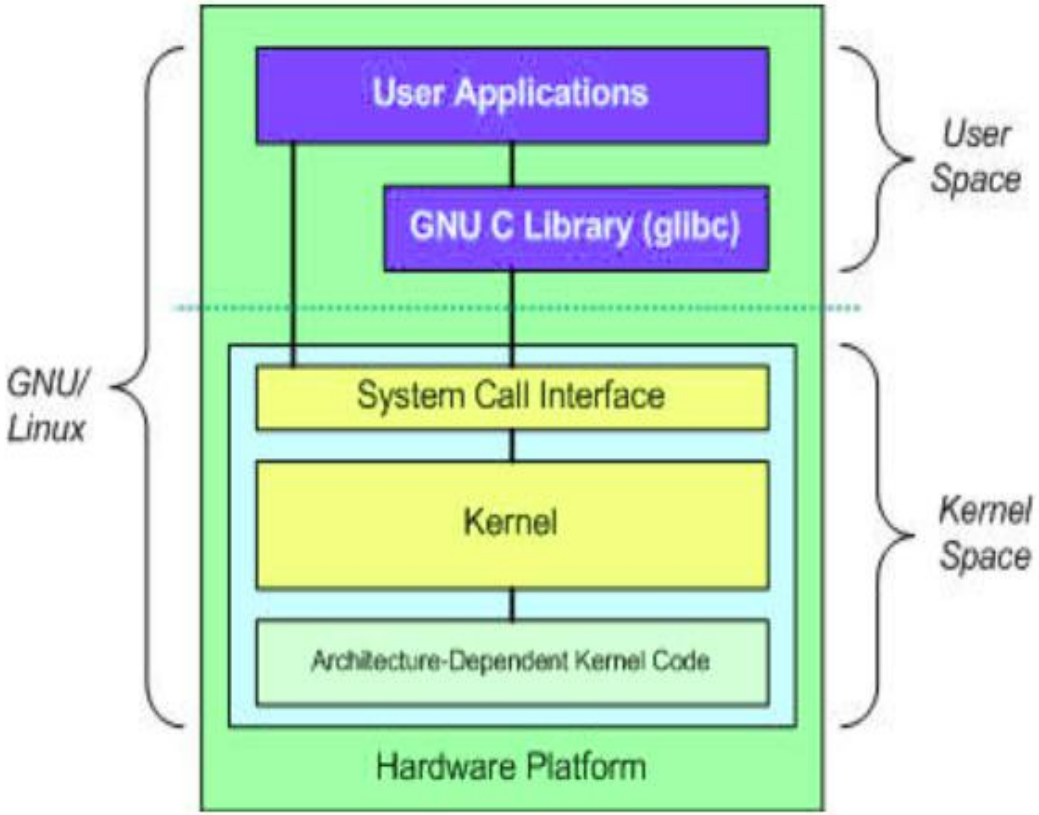
- /home → users' personal directories
- /etc → system configuration files
- /bin, /sbin → essential system commands
- /var → variable files (logs, cache, temporary data)
- /usr → installed programs, libraries and resources

Chapter 2: Open Source Tools

- 2.2. Development Environment

2.2.2 Introduction to Linux

Architecture of a Linux System



Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.2 Introduction to Linux

Main Characteristics of Linux

- **1. Multiplatform**
  - Linux can run on a wide range of hardware architectures, such as:
  - personal computers (x86, x86_64),
  - servers (ARM, PowerPC, RISC-V),
  - smartphones (Android),
  - routers and embedded systems (IoT, Raspberry Pi).
- **2. Multitasking** : Linux can execute several tasks at the same time:
  - Downloading, running applications, hosting a website,
  - running background services.
  - It manages process priorities efficiently.
  - Advantage: smooth performance with multiple programs.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.2 Introduction to Linux

Main Characteristics of Linux

- **3. Multi-user**
  - Several users can use the same system simultaneously:
  - each has a personal home directory,
  - different permissions,
  - secure separation of data.
  - Advantage: essential for servers, universities, and enterprise systems.
- **2. 5. High Security**
  - Linux uses strong security mechanisms: file permissions (read/write/execute),
  - users and groups, separation between root (administrator) and standard users, frequent security updates.
  - Advantage: very resilient to malware and attacks.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.2 Introduction to Linux

Main Characteristics of Linux

5. Modular Architecture

- The Linux kernel can load and unload modules dynamically:
- hardware drivers, file systems, networking components.
- Advantage: flexible and lightweight.

6. Hybrid Kernel

- Linux uses a hybrid approach:
- monolithic kernel → high performance
- loadable modules → flexibility
- Advantage: balance between efficiency and extensibility.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Main Characteristics of Linux

5. Modular Architecture

- The Linux kernel can load and unload modules dynamically:
- hardware drivers, file systems, networking components.
- Advantage: flexible and lightweight.

6. Hybrid Kernel

- Linux uses a hybrid approach:
- monolithic kernel → high performance
- loadable modules → flexibility
- Advantage: balance between efficiency and extensibility.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Basic Linux Terminal Commands

4.1. The BASH Command Interpreter and the Terminal

- Regardless of the Linux distribution being used, the system always provides a **terminal application** that can be launched to interact with the operating system. The default command interpreter in most Linux environments is **Bash (Bourne Again Shell)**, which is the most widely used shell for executing commands and scripts. Bash offers powerful features for file manipulation, process control, automation, and system administration.
- It can also be enabled and used on **Windows 10** through the *Windows Subsystem for Linux (WSL)*, allowing users to access a Linux-like command-line environment directly from Windows.

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Main Characteristics of Linux

Command	Role / Description	Example
pwd	Displays the current working directory.	`pwd` → `/home/student/Documents`
ls	Lists files and directories in the current location.	`ls -l` → long, detailed listing
cd	Changes the current directory.	`cd /etc`
mkdir	Creates a new directory.	`mkdir Project1`
rmdir	Removes an empty directory.	`rmdir old_folder`
touch	Creates an empty file or updates file timestamps.	`touch notes.txt`
cp	Copies files or directories.	`cp report.txt /home/student/backup/`
mv	Moves or renames files and directories.	`mv oldname.txt newname.txt`

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Command	Role / Description	Example
cat	Displays the content of a file.	`cat config.txt`
nano	Opens a simple text editor in the terminal.	`nano script.sh`
echo	Displays a message or writes text to a file.	`echo "Hello" > file.txt`
man	Shows the manual/help page of a command.	`man ls`
chmod	Changes file or directory permissions.	`chmod 755 script.sh`
chown	Changes file or directory owner.	`chown user:user file.txt`
grep	Searches for patterns inside files.	`grep "error" logs.txt`

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Command	Role / Description	Example
find	Searches for files and directories in the system.	`find /home -name "*.pdf"`
top	Displays real-time system processes and usage.	`top`
df	Shows disk space usage.	`df -h`
du	Shows directory or file size.	`du -sh Documents/`
ps	Displays running processes.	`ps aux`
kill	Terminates a running process.	`kill 1234`

Chapter 2: Open Source Tools

- 2.2. Development Environment
- 2.2.3 Introduction to code editors

Command	Role / Description	Example
wget	Downloads files from the internet.	`wget https://example.com/file.zip`
tar	Creates or extracts tar archives.	`tar -xvf archive.tar`
sudo	Executes commands with administrator privileges.	`sudo apt update`
uname	Shows system information.	`uname -a`
history	Displays previously executed commands.	`history`

Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

Introduction

- LibreOffice is a **free and open-source office suite** that allows users to create and edit various types of documents, including text files, spreadsheets, presentations, drawings, databases, and mathematical formulas.
  It is one of the best alternatives to Microsoft Office and works on Windows, Linux, and macOS.
- LibreOffice is developed by **The Document Foundation** and uses open standards, ensuring long-term compatibility and document sustainability.

Chapter 2: Open Source Tools

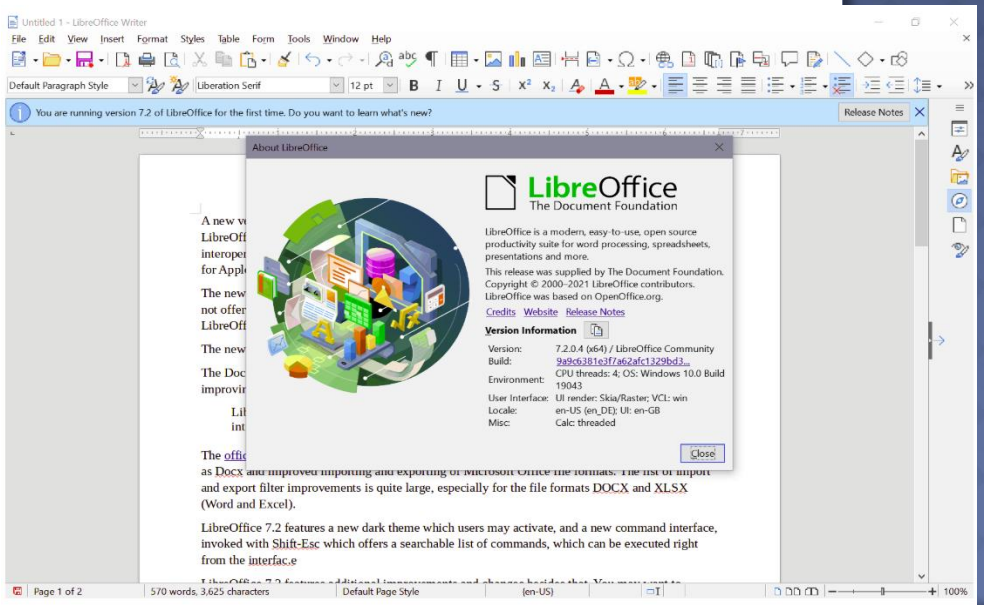
2.3 LibreOffice: Introduction and Components of the Office Suite

Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

Main Components of LibreOffice

- **1. Writer (Word Processing)**
- Writer is used to create written documents such as letters, reports, articles, and academic papers.
- Native format: **ODT**
- Key features: text formatting, tables, images, styles, headers/footers, and formula insertion.
- Supports quick export to **PDF**.

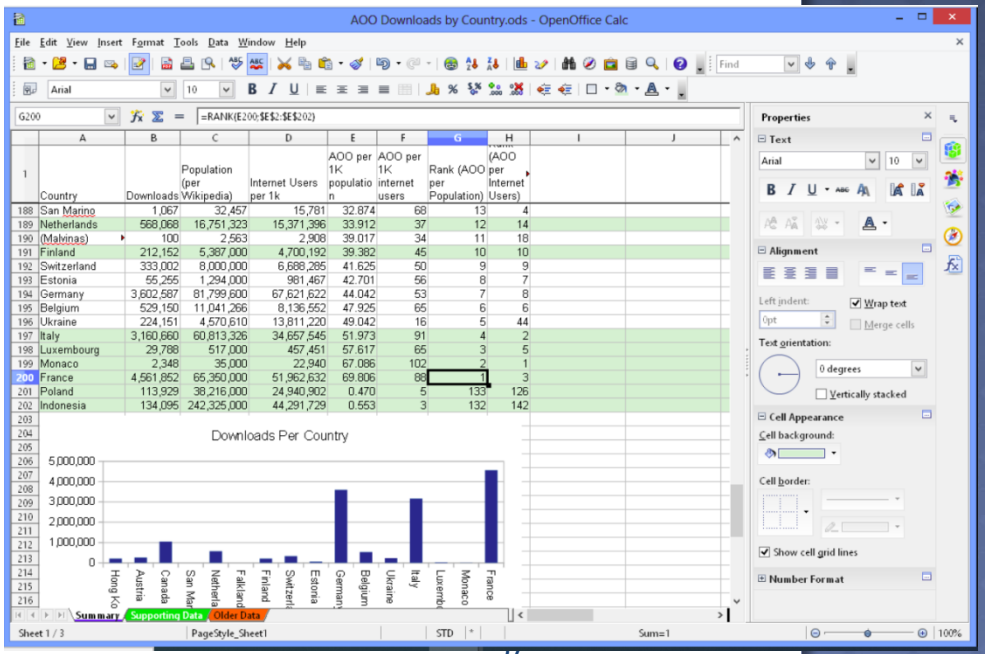


Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

Main Components of LibreOffice

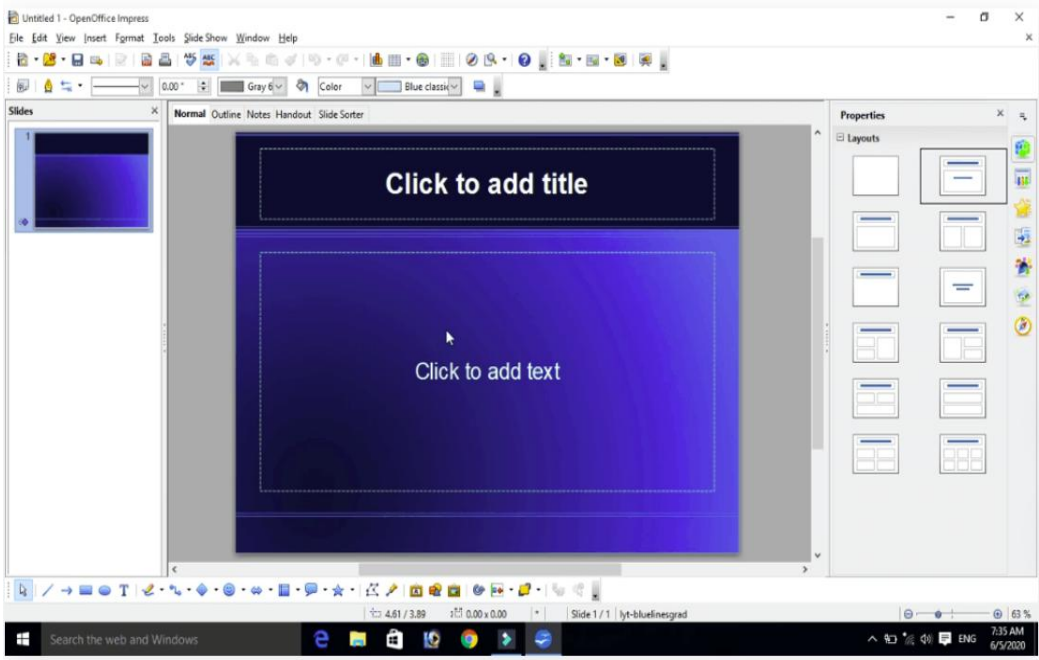
- **2. Calc (Spreadsheet)**
- Calc is used to organize data in tables and perform automatic calculations.
- Native format: **ODS**
- Features: formulas, charts, sorting, filtering, and data analysis tools.
- Comparable to Microsoft Excel.



Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

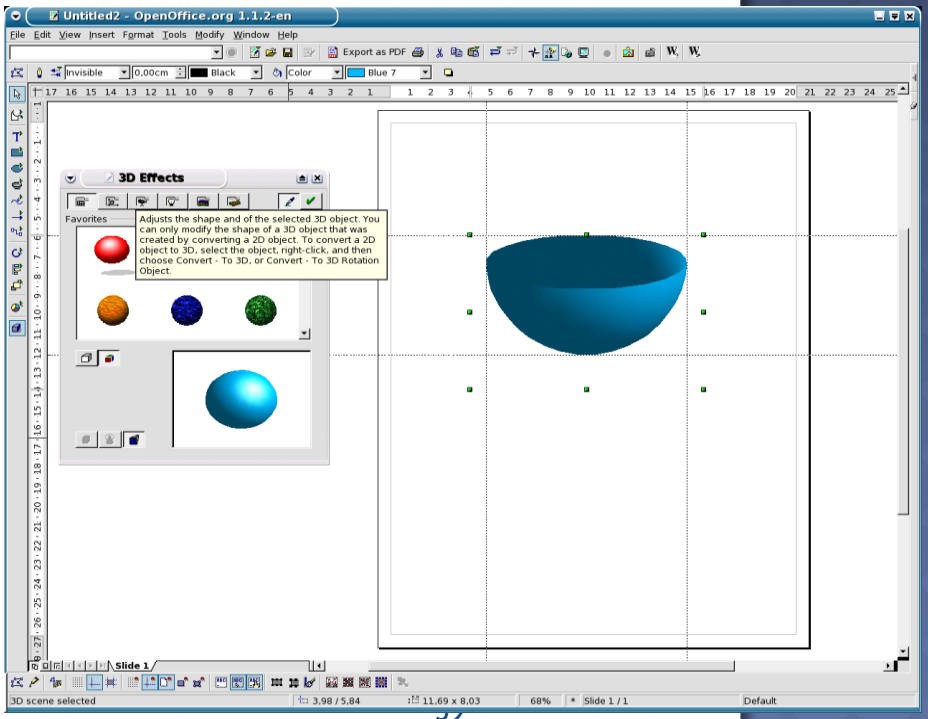
- **Main Components of LibreOffice**
- **3. Impress (Presentations)**
- Impress is used to create slideshow presentations for courses, seminars, or professional meetings.
- Native format: **ODP**
- Features: templates, transitions, animations, multimedia insertion.



Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

- **Main Components of LibreOffice**
- **4. Draw (Vector Graphics Editor)**
- Draw is a tool for creating diagrams, flowcharts, mind maps, and for editing PDF files.
- Ideal for designing simple technical illustrations.
- Comparable to Microsoft Visio.

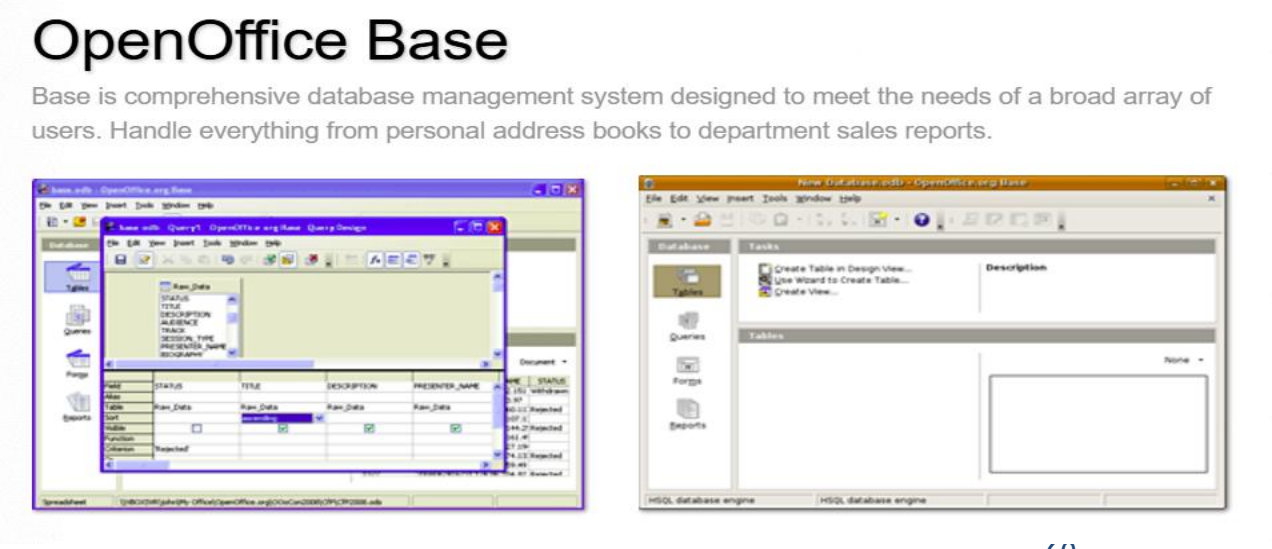


Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

Main Components of LibreOffice

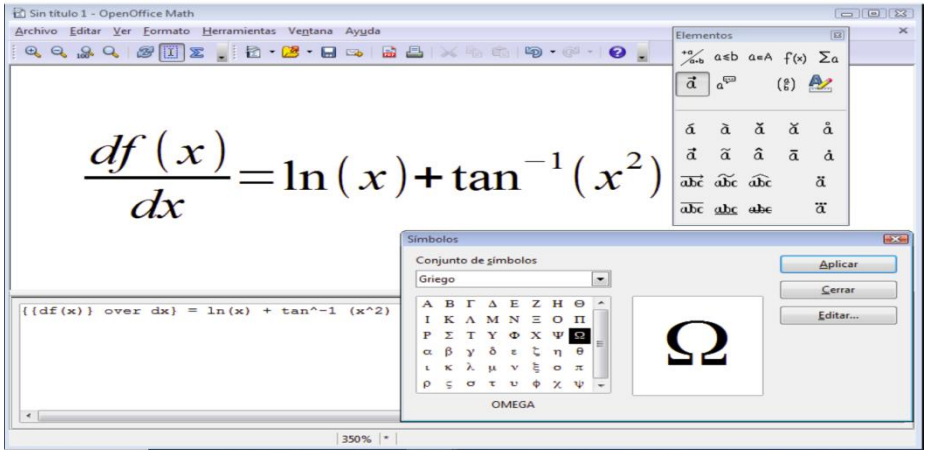
- **5. Base (Database Management)**
- Base is used to create and manage databases.
- Supports multiple database engines: HSQLDB, MariaDB, PostgreSQL, etc.
- Useful for learning tables, queries, forms, and reports.



Chapter 2: Open Source Tools

2.3 LibreOffice: Introduction and Components of the Office Suite

- **Main Components of LibreOffice**
- **6. Math (Formula Editor)**
- Math allows users to insert mathematical expressions into documents.
- Uses a simple markup language (example: sum from i=1 to n a_i)
- Very helpful for scientific or academic writing.



Chapter 2: Open Source Tools

2.4. Collaboration

Collaboration (Storage and Sharing)

- In the world of Free and Open Source Software (FOSS), projects are developed by people from many countries.
- Therefore, effective tools are needed to **work together**, **store code**, **share files**, and **coordinate contributions**.

2.4 .1. Code Storage and Version Control

A version control system allows developers to:

- record every change made to the code,
- identify who changed what and when,
- return to a previous version if a mistake occurs,
- work on different parts of the project at the same time.

Chapter 2: Open Source Tools

2.4. Collaboration

2.4 Collaboration (Storage and Sharing)

➢ Git

- Git is the most widely used system. It allows developers to:
- track the full history of the project,
- create *branches* to test new ideas safely,
- merge changes from multiple people,
- keep a local copy of the project and work offline.
- Other systems: **Mercurial** and **Subversion (SVN)** (less common today).

❖ Chapter 2: Open Source Tools

❖ 2.4. Collaboration

◉ 2.4.2. Code Sharing Platforms

- To store and share Git repositories, developers use online platforms such as:
- ➢ **GitHub :** The most popular platform for open-source projects.
- ➢ **GitLab :** Common in professional environments, includes strong CI/CD tools.
- ➢ **Bitbucket :** Often used by companies and private teams.

hese platforms provide essential features:

- repository hosting
- bug tracking (Issues)
- code review (Pull Requests or Merge Requests)
- project documentation (README, Wiki)
- collaboration and discussion tools

Chapter 2: Open Source Tools

2.4. Collaboration

2.4.3.Communication and Coordination Tools

Collaboration is not only about code; communication is also crucial.

Examples of tools used:

- **Mailing lists** and **forums** – announcements and technical discussions
- **Chat platforms** – Slack, Discord, Matrix, IRC
- **Project Wikis** – architecture, guidelines, workflows
- **Example:**
- A mapping software project may include developers from Algeria, France, and Japan.
  Git tracks changes, GitHub manages issues and discussions, and the team communicates through Slack or a mailing list.

Chapter 2: Open Source Tools

2.5 Contributing to an Open Source Project

- Contributing means helping improve a project: fixing bugs, adding features, improving documentation, or assisting users.

2.5.1. Selecting a Project

Choose a project based on:

- your programming skills,
- your interests (AI, health, education, mapping, etc.),
- your experience level (many projects have “beginner-friendly” issues).
- Check project activity (recent commits, active maintainers, clear documentation).

❖ Chapter 2: Open Source Tools

2.5 Contributing to an Open Source Project

- Contributing means helping improve a project: fixing bugs, adding features, improving documentation, or assisting users.

2.5.2. Understanding the Project

Before contributing:

- read the documentation,
- explore the code structure,
- review open issues,
- identify the community's coding standards.

2.5.3. Reporting Issues

- If you find a bug:
- open an Issue on GitHub/GitLab,
- describe the problem clearly,
- include steps to reproduce it,
- explain expected vs. actual behavior.

❖ Chapter 2: Open Source Tools

2.5 Contributing to an Open Source Project

◉ 2.5.4. Submitting Contributions (Pull Request / Merge Request)

◉ Standard workflow:

1. **Fork** the repository.
2. **Clone** it to your computer.
3. **Create a new branch** for your fix or feature.
4. Implement and test your changes.
5. **Push** your branch.
6. Open a **Pull Request**.
7. Maintainers review, request improvements, and merge your work.

❖ Chapter 2: Open Source Tools

2.5 Contributing to an Open Source Project

Contributing means helping improve a project: fixing bugs, adding features, improving documentation, or assisting users.

◉ 2.5.5. Best Practices

- ◉ Write clear commit messages.
- ◉ Follow the project's guidelines.
- ◉ Add or update tests if necessary.
- ◉ Be respectful and patient during code reviews.

◉ 2.5.6. Benefits of Contributing

Contributing to open source helps you:

- ◉ develop programming and teamwork skills,
- ◉ build a public portfolio,
- ◉ connect with experts worldwide,
- ◉ improve tools used by thousands or millions of people.