Operating System: Unix

Thomas de Jaeger

thomas.de-jaeger@epita.fr

Spring 2025

Class details

- 27h: 2h*12 + 1h mid term +2h exam final

- Grade: 15% Participation+Labs/quizz
  35% Mid term exam
  50% Final exam

- All the material will be online: https://github.com/tdejaeger/Epita (does not work yet)

- For anonymous feedback:Feedback - Google Docs

Class organisation

- Lectures
- Labs during class
- Quizz

Syllabus

- **Session 1:** Introduction to Unix/Linux
- **Session 2:** Installation of Linux (WSL, VirtualBox, Mac)
- **Session 3:** Linux Architecture & File Systems
- **Session 4:** CLI Basics: Commands & Navigation
- **Session 5:** Advanced CLI: I/O & Text Processing
- **Session 6:** Shell Scripting I: Basics & Automation
- **Session 7:** Shell Scripting II: Advanced Scripts
- **Session 8:** Processes & Init Systems
- **Session 9:** Users, Groups & Permissions
- **Session 10:** Networking Essentials
- **Session 11:** Text Editors: vi/vim, nano, emacs
- **Session 12:** System Administration & Security Essentials

History of Unix/Linux

Session 1

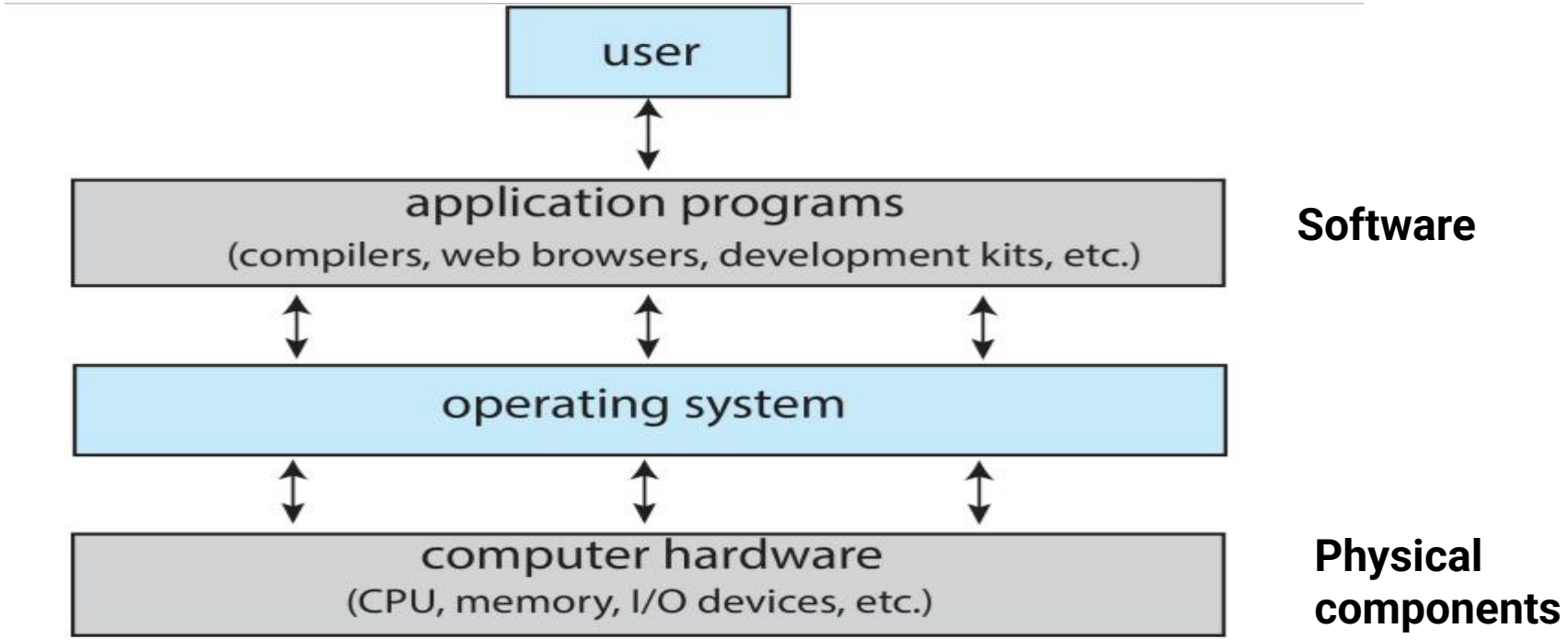
Objectives: Understand the origin, evolution, and impact of UNIX/Linux systems

History of Unix/Linux

What is an operating system?

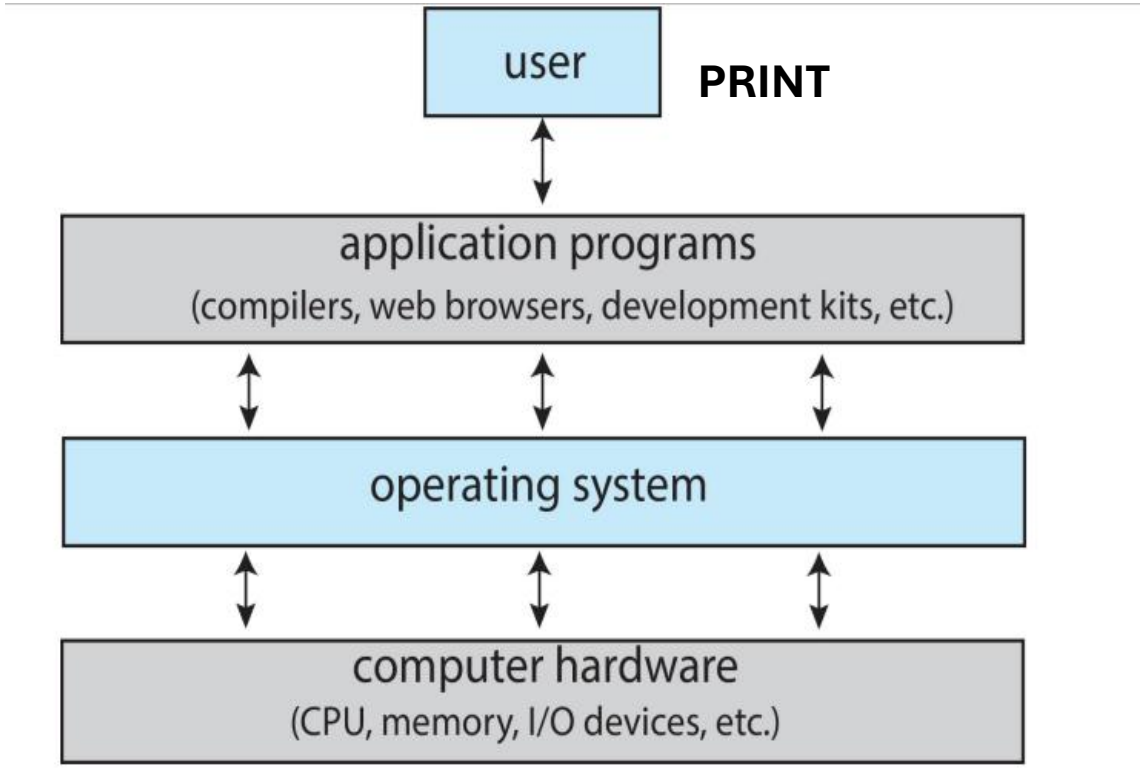
Operating system

An OS is a layer of software that sits between your applications and your computer's hardware. **It manages the computer's resources like the CPU, memory, and devices such as printers, keyboards, and screens**. Without an operating system, you'd need to program directly in machine language for each piece of hardware.



Operating systems

Imagine two programs both trying to print at the same time. The OS steps in to manage that, ensuring only one gets access to the printer at once. It also makes sure that if a program is waiting for something—like data from the network—it doesn't block the whole system. Instead, it pauses that task and lets others continue. This efficient management makes multitasking possible. Plus, it provides abstractions—like files instead of raw disk blocks—so developers can focus on writing useful programs without worrying about hardware details



Operating systems

Operating system goals:

- Execute user programs and make solving user problems easier
- Make the computer system convenient to use
- Use the computer hardware in an efficient manner, acts like a resource manager

These goals may sometimes conflict. For instance, improving performance might reduce convenience. So operating system designers have to make careful choices to balance these needs.

Operating systems

- Examples:

Operating systems

- Examples: Android, macOS, Windows, Linux/GNU

Operating systems

Advantages:

- Operating System manages external and internal devices for example, printers, scanners, and other.
- Operating System provides interfaces and drivers for proper communication between system and hardware devices without needing to understand every technical detail..
- Allows multiple applications to run simultaneously
- Organizes and manages files on storage devices
- Operating system allocates resources to various applications and ensures their efficient utilization.

Operating systems

Disadvantages:

- If an error occurred in your operating system, then there may be a chance that your data may not be recovered therefore always have a backup of your data.
- Threats and viruses can attack our operating system at any time, making it challenging for the OS to keep the system protected from these dangers.
- learning about new operating system can be a time-consuming
- Operating systems consume system resources, including CPU, memory, and storage, which can affect the performance of other applications.

Operating systems

What was the first operating system?

Operating systems

- **1940s-1950s: Early Beginnings:**
  - Computers operated without operating systems
  - Programs were manually loaded and run, one at a time
  - Operators had no direct interface with the system, and job implementation was done in a batch system, meaning that users have to prepare a job separately to perform it.

  - The first operating system was introduced in **1956**. It was a batch processing system GM-NAA I/O (1956) that automated job handling

Operating systems

- **1960s: Multiprogramming and Timesharing**
  - Introduction of multiprogramming to utilize CPU efficiently.
  - Timesharing systems, it allowed multiple users to access the same machine simultaneously. Two major systems were CTSS in 1961 and Multics in 1969.

These systems showed the power of shared computing, and they heavily influenced what came after.

Operating systems

- **1970s: Unix and Personal Computers**
    - Unix was born in 1971, and it brought something new: simplicity, portability, and multitasking.
    - Personal computers emerged, leading to simpler OSs like CP/M (1974) and PC-DOS (1981).

Systems like CP/M and later DOS allowed people to run computers at home or in small offices. But Unix was special—it was built for power users and developers, and its influence would be enormous

Operating systems

- **1980s: GUI and Networking**

In the 1980s, computing moved closer to the general public

  - Graphical User Interfaces (GUIs) gained popularity with systems like Apple Macintosh (1984) and Microsoft Windows (1985). Now you didn't need to memorize commands—you could point and click

  - Unix systems adopted TCP/IP, the protocol that powers the internet today. So now computers could not only be personal—they could also be connected.

Operating systems

- **1990s: Linux and Advanced GUIs**
  - Linux (1991). Linus Torvalds released the Linux kernel as a free alternative to proprietary Unix systems
  - Graphical interfaces became more advanced. Windows 95, Mac OS, and desktop environments like KDE and GNOME on Linux made computers more attractive and intuitive.

  It was a period of rapid growth, especially as the internet took off.
- **2000s-present: Mobility & Clouds**
  - Phones became smartphones. Apple released iOS in 2007, and Android—based on Linux—arrived in 2008.
  - Cloud computing exploded, with Amazon, Google, and Microsoft offering powerful Linux-based servers on demand.
  - AI integration

Why UNIX was created?

- **The Computing Context of the 1960s**
  - Computers were large, expensive, and shared
  - Operating systems were **complex, proprietary**, and specific to each machine. Each vendor had its own OS, and they weren't compatible
  - Developers wanted **more portable**, **efficient**, and **multi-user** environments.

→ Existing systems were **not flexible or user-friendly** for developers.

Why UNIX was created?

- **Multics – The Ambitious Predecessor:**
  - **(Multiplexed Information and Computing Service)** was started in the **mid-1960s** by **MIT**, **Bell Labs**, and **General Electric**
  - Aimed to create a **next-generation, time-sharing** operating system that would:
    - Serve **many users simultaneously**.
    - Be **modular** and easily expandable.
  - Ambitious goals:
    - Real-time performance and reliability
    - Hierarchical file system (tree-like structure).
    - Dynamic linking and segmented memory management
    - Written in a high-level language—very advanced for its time

Why UNIX was created?

- **Multics – The Ambitious Predecessor:**

But it failed

- Multics became **overly complex and hard to manage**
- **Development delays** and **high costs** led Bell Labs to **pull out in 1969**
- Although **innovative**, it was too ambitious for the hardware and software capabilities of the era.
- Ken Thompson, who had worked on Multics, wanted to build something simpler—something that captured the good ideas, but left the bloat behind. That became Unix

Why UNIX was created?

UNIX:

- The **idea of a time-sharing OS**.
- A **hierarchical file system**.
- **User access control** and **shell-based interaction**.

Multics inspired UNIX but also showed the importance of **simplicity and elegance** in system design

History of UNIX

•In **1969**, **Ken Thompson** at AT&T Bell Labs developed the first prototype of Unix on the PDP-7 which was a small machine in the early days of computing, but perfect for experimentation.

The original name was ‘Unics’—a pun on Multics. It eventually became Unix

What’s amazing is that this project, born from a failed one, ended up shaping the future of computing.

History of UNIX

•In **1969**, **Ken Thompson** at AT&T Bell Labs developed the first prototype of Unix on the PDP-7 which was a small machine in the early days of computing.

•In **1972**, **Dennis Ritchie** at AT&T Bell Labs created the **C programming language**. C was designed to be a high-level, portable language that was still close enough to the hardware for system-level programming

History of UNIX

•In **1969**, **Ken Thompson** at AT&T Bell Labs developed the first prototype of Unix on the PDP-7 which was a small machine in the early days of computing.

•In **1972**, **Dennis Ritchie** at AT&T Bell Labs created the **C programming language**. C was designed to be a high-level, portable language that was still close enough to the hardware for system-level programming

•The process of rewriting Unix in C began around **1973**, after Ritchie and **Ken Thompson** realized that C's portability would allow Unix to run on multiple types of machines. This was a turning point: portability made Unix viral. It could spread. And that's exactly what it did.

History of UNIX

•In **1969**, **Ken Thompson** at AT&T Bell Labs developed the first prototype of Unix on the PDP-7 which was a small machine in the early days of computing.

•In **1972**, **Dennis Ritchie** at AT&T Bell Labs created the **C programming language**. C was designed to be a high-level, portable language that was still close enough to the hardware for system-level programming.

•The process of rewriting Unix in C began around **1973**, after Ritchie and **Ken Thompson** realized that C's portability would allow Unix to run on multiple types of machines. This was a turning point: portability made Unix viral. It could spread. And that's exactly what it did.

→ Rewrite Unix in C was revolutionary because it made Unix **portable** for the first time (unlike the PDP-7 or PDP-11). Unix could now be adapted to run on various machines, from mainframes to minicomputers.

History of UNIX

•In **1969**, **Ken Thompson** at AT&T Bell Labs developed the first prototype of Unix on the PDP-7 which was a small machine in the early days of computing.

•In **1972**, **Dennis Ritchie** at AT&T Bell Labs created the **C programming language**. C was designed to be a high-level, portable language that was still close enough to the hardware for system-level programming.

•The process of rewriting Unix in C began around **1973**, after Ritchie and **Ken Thompson** realized that C's portability would allow Unix to run on multiple types of machines.

•1979: After years of development and academic distribution, **AT&T (Bell Labs)** officially **commercialized UNIX in 1979** with the release of **Version 7 Unix (V7)**.

The last true UNIX version

Open-source

Open-source

•**Open Source** means the source code of a software is made freely available, and users are allowed to **view**, **modify**, **use**, and **redistribute** it.

This idea is the foundation for Linux and many other modern tools. It’s not just about cost—it’s about freedom, collaboration, and transparency. Open source enables innovation by letting people around the world work together on the same software.

Open-source

- **Open Source** means the source code of a software is made freely available, and users are allowed to **view**, **modify**, **use**, and **redistribute** it.

- **BSD's Open Source Legacy:**

  - In **1977**, researchers at the **University of California, Berkeley** released the **first version of BSD**, an enhanced Unix-like system based on **Unix Version 6 (V6)** from Bell Labs

  - BSD began as a **set of add-ons and improvements** to AT&T's Unix, including tools like the **C shell (csh)**, improved performance, and networking utilities.

  - It quickly evolved into a **full Unix-like operating system** used widely in academia and eventually in commercial products.

Open-source

•**Open Source** means the source code of a software is made freely available, and

- Because BSD was originally based on AT&T Unix code, **AT&T sued the University of California in the early 1990s**, claiming that BSD still contained proprietary Unix source code.
- This lawsuit created uncertainty in the Unix world and **delayed the adoption of free Unix-like systems**.
- Eventually, the **lawsuit was settled** in **1994**, and the BSD codebase was cleaned of AT&T code, leading to the **fully open and free 4.4BSD-Lite** release.

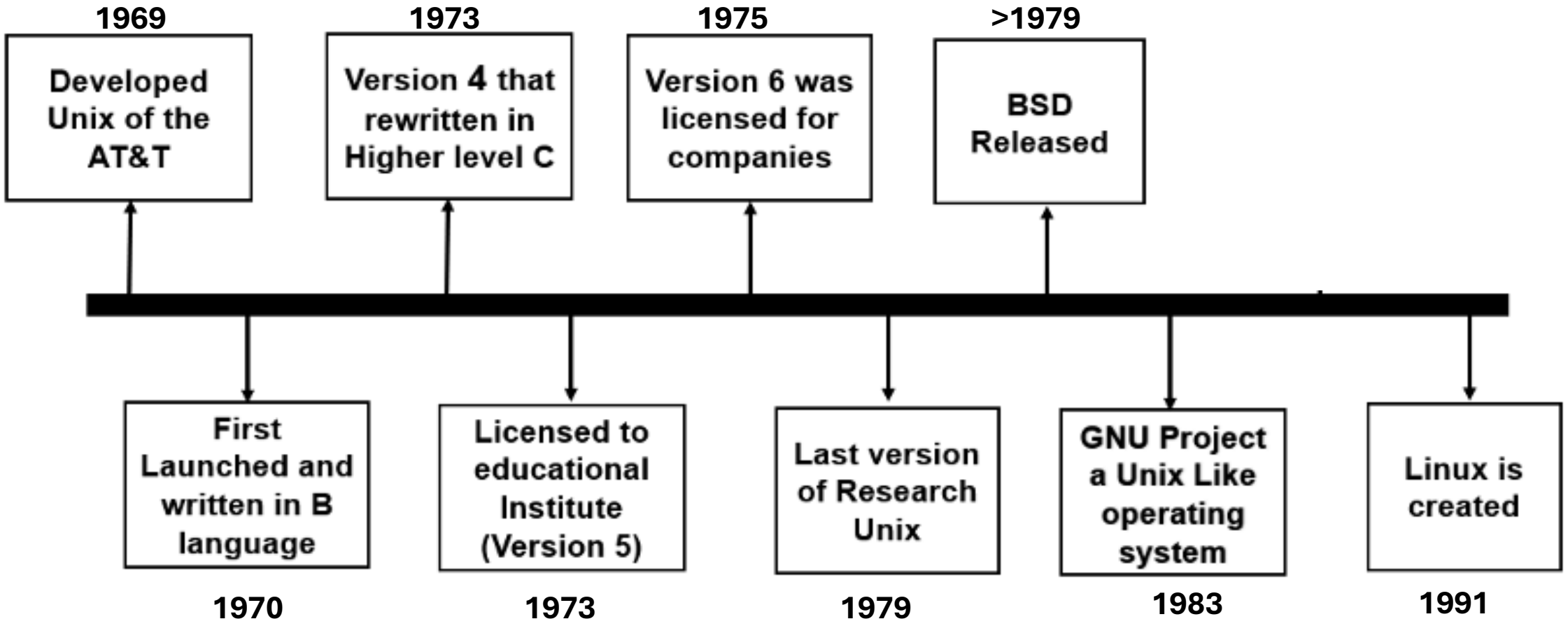
academia and eventually in commercial products.

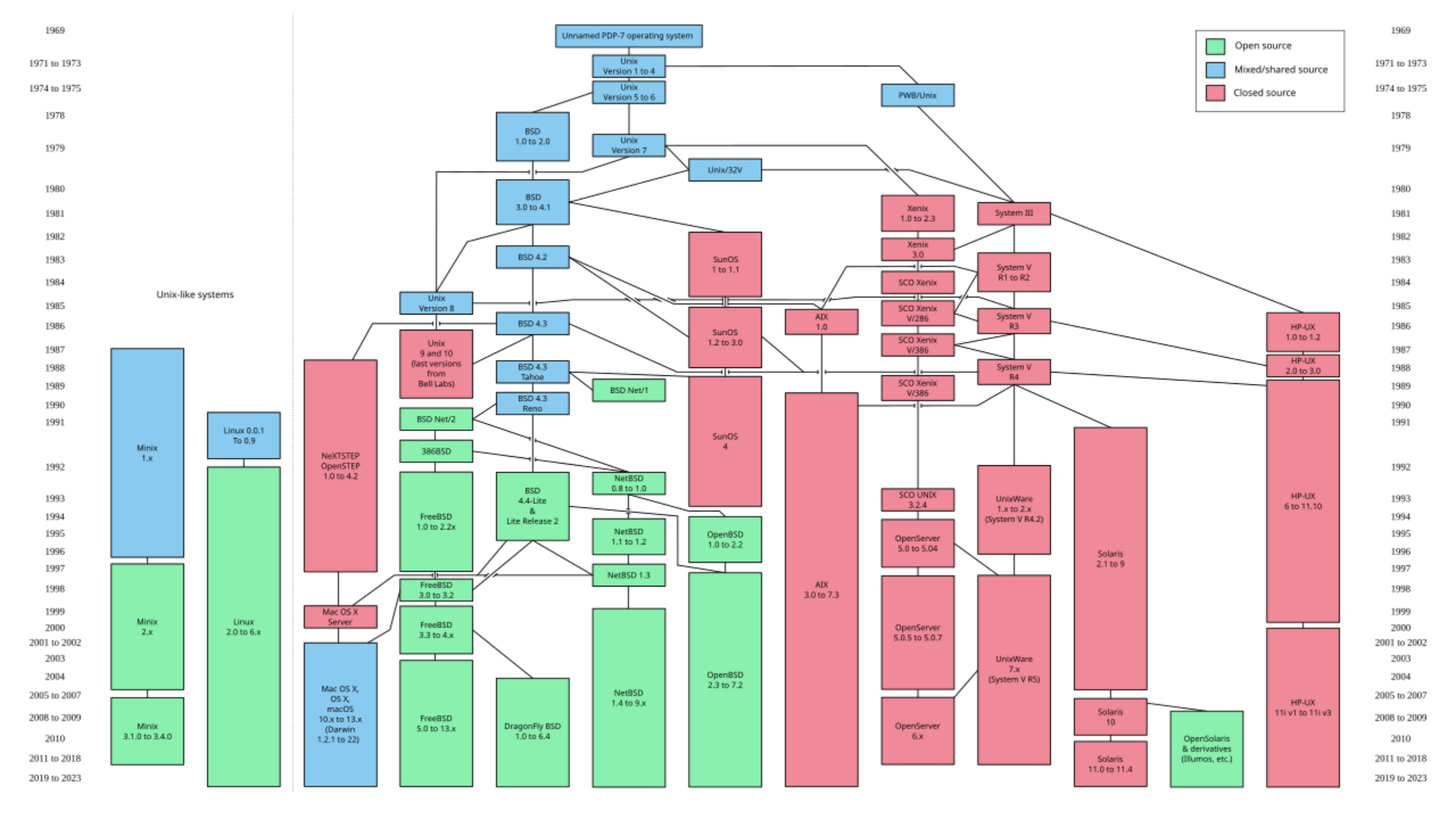
Open-source

- **GNU project :**
  - In 1983, another major player emerged: Richard Stallman launched the GNU Project
  - Philosophy: Build a **completely free** Unix-like operating system.
  - "GNU" = "**GNU's Not Unix**".
  - GNU is also a software license – allows for code modifications as long as they are shared. This ensures freedom and collaboration.

GNU wasn't a full OS at first, but it provided all components except the kernel. Its philosophy is about user freedom: to use, modify, and share software

History of UNIX



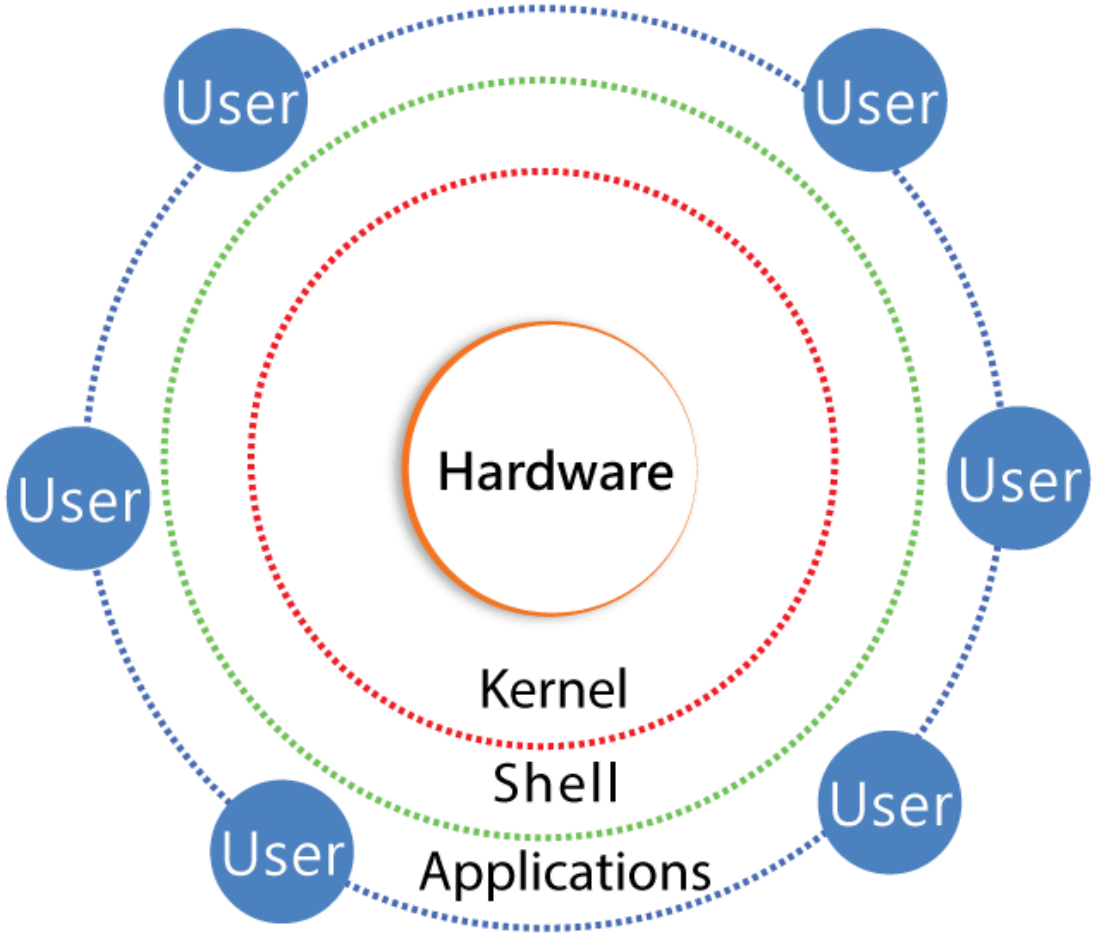

Open source
Mixed/shared source
Closed source
Unix-like systems
1969
1971 to 1973
1974 to 1975
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001 to 2002
2003
2004
2005 to 2007
2008 to 2009
2010
2011 to 2018
2019 to 2023
Unnamed PDP-7 operating system
Unix Version 1 to 4
Unix Version 5 to 6
Unix Version 7
PWB/Unix
BSD 1.0 to 2.0
Unix/32V
BSD 3.0 to 4.1
Xenix 1.0 to 2.3
Xenix 3.0
System III
BSD 4.2
SunOS 1 to 1.1
System V R1 to R2
SCO Xenix
Unix Version 8
SCO Xenix V/286
AIX 1.0
System V R3
BSD 4.3
SunOS 1.2 to 3.0
SCO Xenix V/386
HP-UX 1.0 to 1.2
Unix 9 and 10 (last versions from Bell Labs)
BSD 4.3 Tahoe
BSD Net/1
System V R4
HP-UX 2.0 to 3.0
BSD 4.3 Reno
NeXTSTEP OpenSTEP 1.0 to 4.2
BSD Net/2
SunOS 4
Minix 1.x
Linux 0.0.1 To 0.9
386BSD
NetBSD 0.8 to 1.0
FreeBSD 1.0 to 2.2x
BSD 4.4-Lite & Lite Release 2
SCO UNIX 3.2.4
UnixWare 1.x to 2.x (System V R4.2)
Solaris 2.1 to 9
HP-UX 6 to 11.10
NetBSD 1.1 to 1.2
OpenBSD 1.0 to 2.2
OpenServer 5.0 to 5.04
NetBSD 1.3
FreeBSD 3.0 to 3.2
AIX 3.0 to 7.3
Minix 2.x
Linux 2.0 to 6.x
Mac OS X Server
FreeBSD 3.3 to 4.x
OpenBSD 2.3 to 7.2
OpenServer 5.0.5 to 5.0.7
UnixWare 7.x (System V R5)
Mac OS X, OS X, macOS 10.x to 13.x (Darwin 1.2.1 to 22)
NetBSD 1.4 to 9.x
Solaris 10
HP-UX 11i v1 to 11i v3
Minix 3.1.0 to 3.4.0
FreeBSD 5.0 to 13.x
DragonFly BSD 1.0 to 6.4
OpenServer 6.x
OpenSolaris & derivatives (Illumos, etc.)
Solaris 11.0 to 11.4

Uniqueness

- **Multitasking:** UNIX is capable of running multiple processes at the same time.
- **Multiuser support:** UNIX allows multiple users to simultaneously access the same system and share resources
- **Portability:** UNIX can run on a wide variety of hardware platforms, from small embedded systems to large mainframe computers.
- **Shell scripting:** UNIX provides a powerful scripting language that allows users to automate tasks.
- **Security:** UNIX has a robust security model that includes file permissions, user accounts, and network security features.
- **Process Tracking:** UNIX maintains a record of the jobs that the user creates (improves system performance by monitoring CPU usage and use that information to regulate disk space)

System structure

- **Hardware**: Physical machine (CPU, memory, devices) — *not part of the OS*.
- **Kernel**: Core of the OS that manages and controls hardware resources.
- **Shell**: Interface that allows users to interact with the kernel.
- **User Applications**: Programs run by users, relying on the shell and kernel to function.



Shell, terminal, prompt, command line?

- **Shell**: A program that provides a user interface to interact with the operating system. It interprets and executes commands typed by the user (e.g. bash, ksh, csh, zsh, fish) .

Shell, terminal, prompt, command line?

- **Shell**: A program that provides a user interface to interact with the operating system. It interprets and executes commands typed by the user (e.g. bash, ksh, csh, zsh, fish) .

- **Terminal**: A program that provides the environment where the user can interact with the shell (e.g. GNOME Terminal, Konsole, or xterm).

The terminal is **the interface**, and the shell is **what processes your command**

Shell, terminal, prompt, command line?

- **Shell**: A program that provides a user interface to interact with the operating system. It interprets and executes commands typed by the user (e.g. bash, ksh, csh, zsh, fish) .

- **Terminal**: A program that provides the environment where the user can interact with the shell (e.g. GNOME Terminal, Konsole, or xterm).

- **Prompt**: The text that appears in the terminal, indicating that the shell is ready to accept a command. It usually ends with a symbol such as $ or # (for user or root, respectively) .

Shell, terminal, prompt, command line?

- **Shell**: A program that provides a user interface to interact with the operating system. It interprets and executes commands typed by the user (e.g. bash, ksh, csh, zsh, fish) .

- **Terminal**: A program that provides the environment where the user can interact with the shell (e.g. GNOME Terminal, Konsole, or xterm).

- **Prompt**: The text that appears in the terminal, indicating that the shell is ready to accept a command. It usually ends with a symbol such as $ or # (for user or root, respectively) .

- **Command line**: The actual input field where you type commands in the terminal. It’s the space where you write and execute commands.

Shell, terminal, prompt, command line?

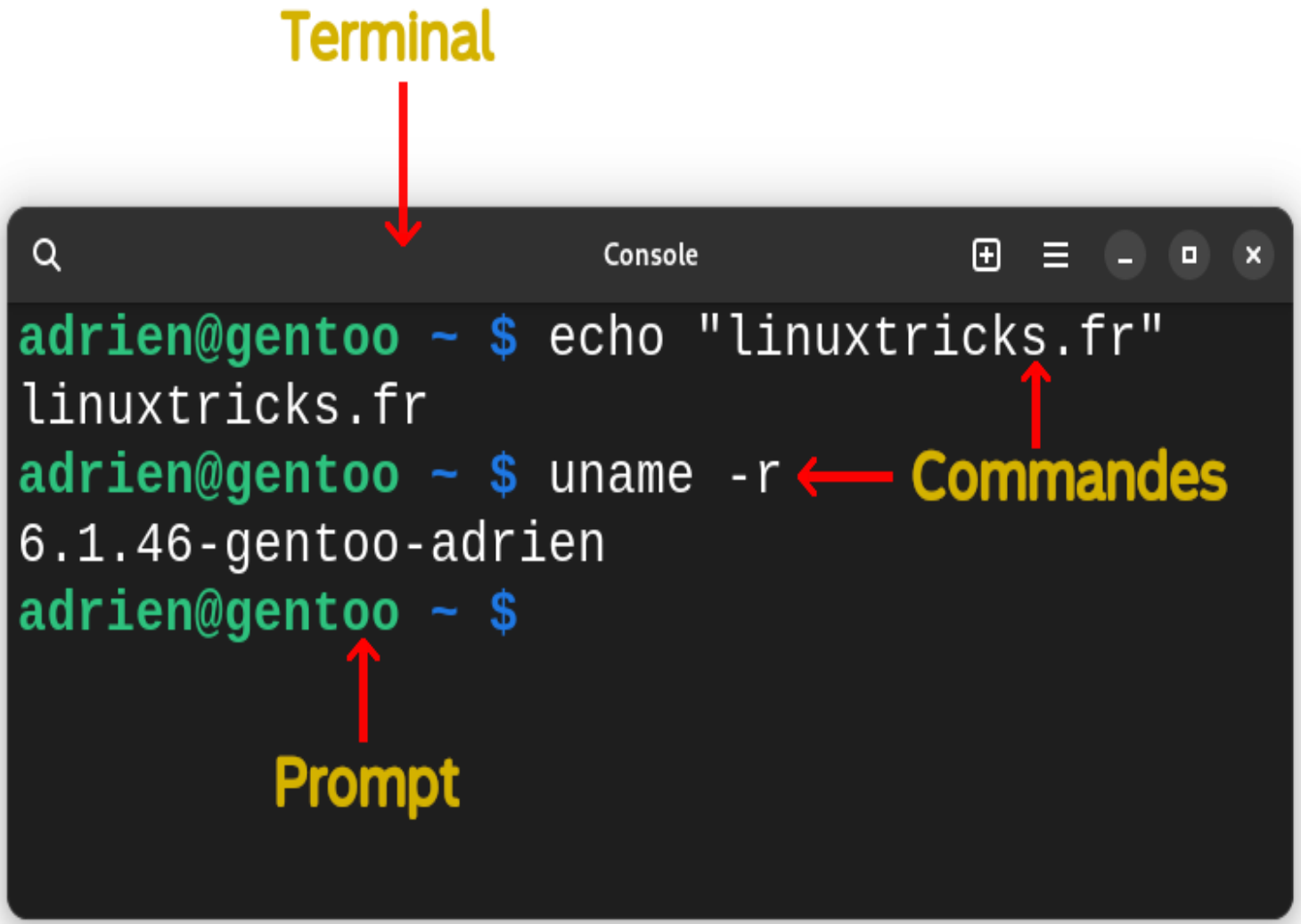
So, in short:

- **Shell**: The program interpreting commands.

- **Terminal**: The program displaying the shell's prompt and results.

- **Prompt**: The text shown, waiting for commands.

- **Command line**: The area where you type your commands.



What is LINUX?

Linux is NOT an open source operating system on its own.

- 🐧 Linux is a **kernel**, first released by **Linus Torvalds** in 1991. It's combined with GNU software to form a complete operating systems like Ubuntu or Fedora (GNU tools—like compilers, libraries, and shells)

- The code used to create Linux **is free and available to the public to view, edit**, and—for users with the appropriate skills—to contribute to.

- Unix and Linux have comparable components, including the kernel, shell, and programs.

Linux versus UNIX

- Unix and Linux are similar in many ways, and in fact, Linux was originally created to be similar to Unix. Both have similar tools for interfacing with the systems, programming tools, filesystem layouts, and other key components.

- However, Unix is not free.

- Over the years, a number of different operating systems have been created that attempted to be “unix-like” or “unix-compatible,” but Linux has been the most successful, far surpassing its predecessors in popularity.

Linux versus UNIX

Feature	Linux	UNIX
Source Code	Open Source	Mostly Proprietary
Cost	Free	Commercial License (varies by version)
Usage	Servers, Desktops, Mobile (e.g., Android)	Enterprise Servers, Workstations
Development	Community-driven (collaborative)	Vendor-specific (AT&T, IBM, Oracle, etc.)
Customization	Highly customizable	Limited customization
Examples	Ubuntu, Fedora, Debian, Arch, CentOS	AIX (IBM), HP-UX, Solaris (Oracle)
User Interface	CLI and GUI options	Mostly CLI (some with GUI)
Compatibility	Compatible with many hardware platforms	Limited to specific hardware

History of LINUX

- Linux was created in 1991 by Linus Torvalds, a then-student at the University of Helsinki. Torvalds built Linux as a free and open source alternative to Minix, another Unix clone that was predominantly used in academic settings.

- At first, he wanted to call it ‘Freax’—a combination of ‘free’ and ‘freak’—but the server admin who hosted it renamed the folder to ‘Linux’ after a combination of Torvalds’ first name and the word Unix, and the name stuck.

Why LINUX?

Why LINUX?

- ✅ **Free and Open Source**

No license fees. You can use, modify, and distribute it.

- 🛡️ **Secure and Stable**

Long uptimes, strong permissions model, fewer viruses.

- 🌐 **Community-Supported**

Thousands of contributors and online resources.

- ☁️ **Widely Used**

Powers servers, supercomputers, IoT, mobile (Android), cloud infrastructure.

- 🛠️ **Ideal for Developers**

Built-in tools, scripting, and full control over the system.

Where is LINUX?

- **Widespread Usage** 🌍 :
  - **Servers** 🖥️ : Dominates web servers, data centers, and cloud infrastructures (AWS, Google Cloud, Azure).
  - **Supercomputers** ⚡ : Over 90% of the world's top supercomputers run on Linux.
  - **Embedded Systems & IoT** 🔌 : Linux powers devices like smartphones, routers, smart TVs, and IoT devices.
- **Linux in Consumer Devices** 📱 :
  - **Android OS** 📲 : Based on Linux, used in nearly all smartphones.
  - **Desktop** 💻 : Used by enthusiasts and in some enterprise environments (e.g., Ubuntu, Fedora).
- **Enterprise & Security** 🔒 :
  - **Enterprise Servers** 🏢 : Linux is widely used for web hosting, databases, and enterprise applications.
  - **Security** 🛡️ : Known for its robustness and security, preferred for sensitive environments.

Linux

What are the different Linux/GNU distributions?

Linux

Over 600 Linux distributions, 300 actively developped. This diversity is a strength—it means people can tailor Linux for servers, desktops, security, privacy, education, gaming, and more. Some distros are designed for specific hardware; others for specific use cases

Linux

Over 600 Linux distributions, 300 actively developped. This diversity is a strength—it means people can tailor Linux for servers, desktops, security, privacy, education, gaming, and more. Some distros are designed for specific hardware; others for specific use cases

Linux

Lab:

Each group picks a distribution and answers (15-20min):

- Origin (date, place)
- Target audience
- Technical features
- 2min presentations with 1-2 slides (all the presentations will be added to the class)