Fundamentos de los Sistemas Operativos (FSO)

Departamento de Informática de Sistemas y Computadoras (DISCA)
Universitat Politècnica de València

Part 1: Introduction

Unit 1

Operating System Concept

Goals and bibliography

- Goals
  - Introducing the Operating System (OS) **concept**
  - Describing the **functions** an OS perform
  - Reviewing the evolution of operating systems to help understanding what services an OS provides and how they are provided
- Bibliography
  - A. Silberschatz, P. B. Galvin. Chapters 1 and 2
  - Wikipedia and other Internet sources:
    - A Brief History of Computing - Operating Systems https://trillian.randomstuff.org.uk/~stephen/history/timeline-OS.html
    - Timeline: 40 years of OS milestones http://www.computerworld.com/article/2531905/operating-systems/timeline--40-years-of-os-milestones.html

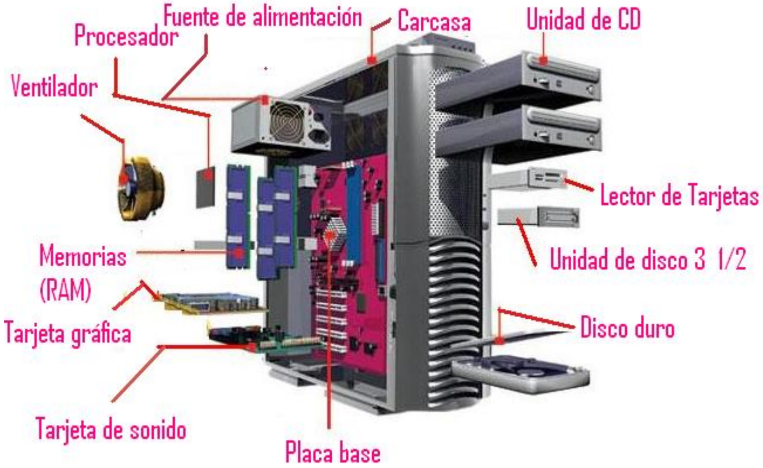
Content

- **Operating system concept**
- Operating system structure
- CPU utilization
- Historical evolution

Terms:	
OS	Operating system
I/O	Input / Output
CPU	Centra processing unit (processor)
API	Application Programming Interface
HAL	Hardware abstraction layer
UNIX	Portable, multitask and multiuser operating system
Linux	Free and open SO kernel based on UNIX

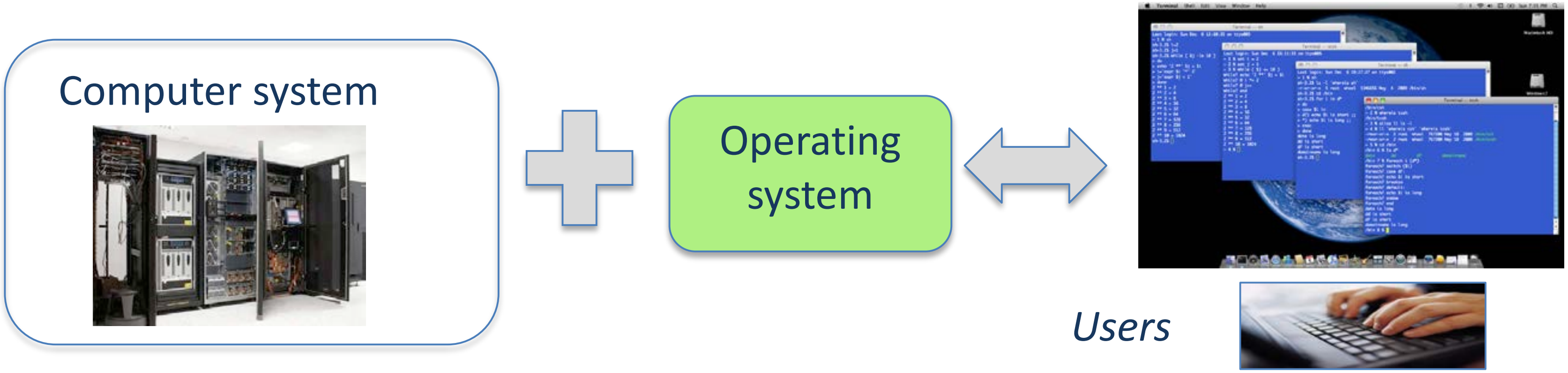
Operating system concept

- A computer system is the set of hardware elements, organized according to a specific "Architecture”, that form a computing device
  - The direct management of these hardware elements is complex and dependent on the devices features
  - These hardware elements have limited capabilities and this leads to the need to establish operating criteria that optimize their use



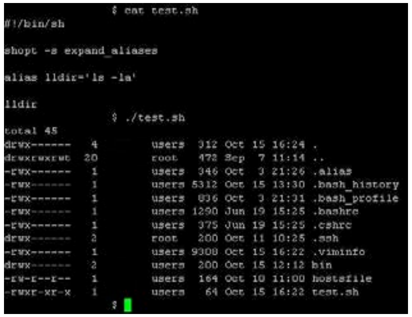
Operating system concept

- Definition
  - **An OS is the set of software** that allows the operation of computer systems and offers a friendly interface to users
- Purpose
  - Creating a **comfortable an efficient environment** to run programs
- OS goals: accessibility, commodity, efficiency, security, portability, etc
  - It acts as the **intermediate** between the user and the system
  - It guaranties the **correct computer operation**
  - It **easies the application creation** task for programmers
  - It manages **efficiently** the hardware resources available



Operating system concept

- An OS should provide **services** to the several kinds of computer **users**
- User types:
  - Application user
  - Application programmer
  - System programmer
  - System administrator

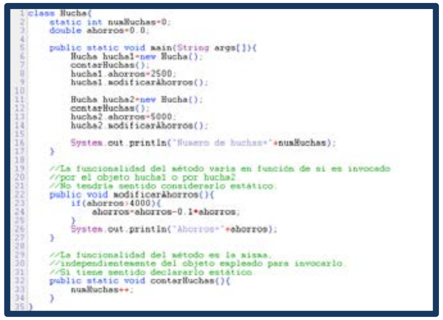


```
                 $ cat test.sh
#!/bin/sh

shopt -s expand_aliases

alias lldir='ls -la'

lldir
                 $ ./test.sh
total 45
drwx------   4        users  312 Oct 15 16:24 .
drwxrwxrwt  20        root   472 Sep  7 11:14 ..
-rwx------   1        users  346 Oct  3 21:26 .alias
-rwx------   1        users 5312 Oct 15 13:30 .bash_history
-rwx------   1        users  836 Oct  3 21:31 .bash_profile
-rwx------   1        users 1290 Jun 19 15:25 .bashrc
-rwx------   1        users  375 Jun 19 15:25 .cshrc
drwx------   2        root   200 Oct 11 10:25 .ssh
-rwx------   1        users 9308 Oct 15 16:22 .viminfo
drwx------   2        users  200 Oct 15 12:12 bin
-rw-r--r--   1        users  164 Oct 10 11:00 hostsfile
-rwxr-xr-x   1        users   64 Oct 15 16:22 test.sh
                 $
```



```
class Hucha{
    static int numHuchas=0;
    double ahorros=0.0;

    public static void main(String args[]){
        Hucha hucha1=new Hucha();
        contarHuchas();
        hucha1.ahorros=2500;
        hucha1.modificarAhorros();

        Hucha hucha2=new Hucha();
        contarHuchas();
        hucha2.ahorros=5000;
        hucha2.modificarAhorros();

        System.out.println("Numero de huchas="+numHuchas);
    }

    //La funcionalidad del método varia en función de si es invocado
    //por el objeto hucha1 o por hucha2
    //No tendria sentido considerarlo estático
    public void modificarAhorros(){
        if(ahorros>4000){
            ahorros=ahorros-0.1*ahorros;
        }
        System.out.println("Ahorros="+ahorros);
    }

    //La funcionalidad del método es la misma,
    //independientemente del objeto empleado para invocarlo
    //Si tiene sentido declararlo estático
    public static void contarHuchas(){
        numHuchas++;
    }
}
```

Operating system concept

- Sights
  - The **operating system abstracts and manages** the operation of all system hardware/software components that make up the computer system
    - **System sight**: Resource Manager and protection
      - SO components and their interrelation
    - **User sight**: Abstraction of resources aimed at facilitating its use -> Extended machine
      - Services provided
      - Interfaces provided to programmers and final users

Operating system concept

- Functions
  - Providing user, programmer and administrator interfaces
    - **Hardware Abstraction**
  - Offering a range of services in the form of "**system calls**"
  - **Manage resources.** Is responsible for deciding which program may use a hardware device and for how long
    - Process, memory, files and I/O management
  - **Security and protection:** Controlling and supervising resource access to avoid conflicts and unauthorized access

Operating system concept

- Systems with OS

Server Pool

Computer Server

Personal Computer

Router

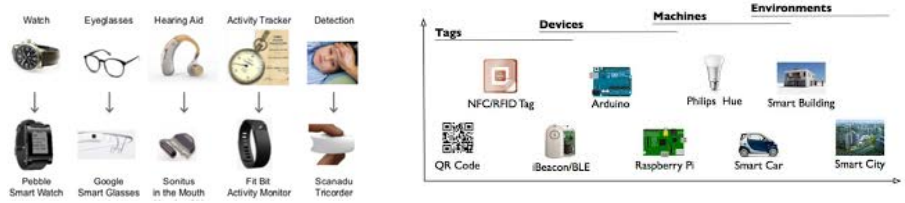
Tablet

Smart phone

Video Console

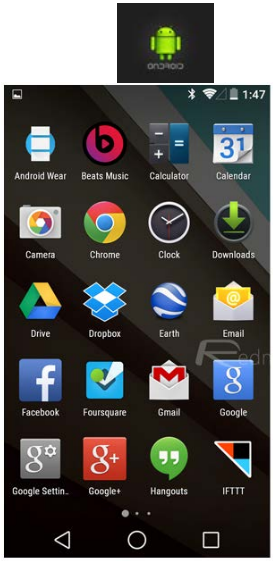
Smart TV

Smart Devices & IoT



Operating system concept

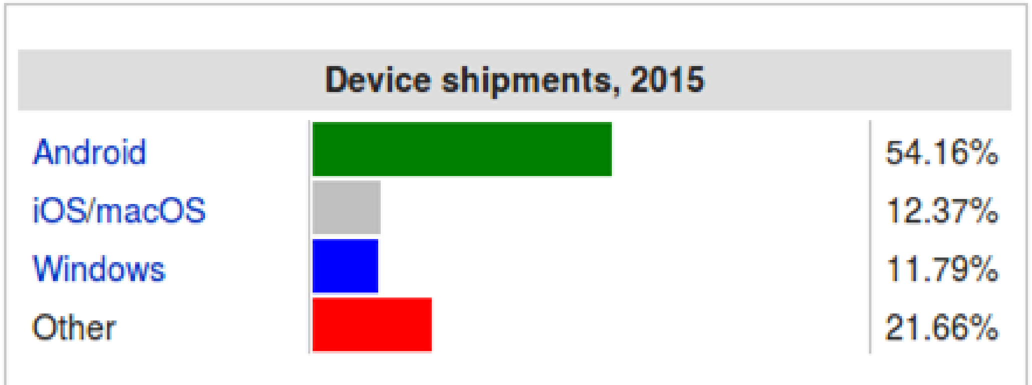
- Nowadays OSs



Operating system concept

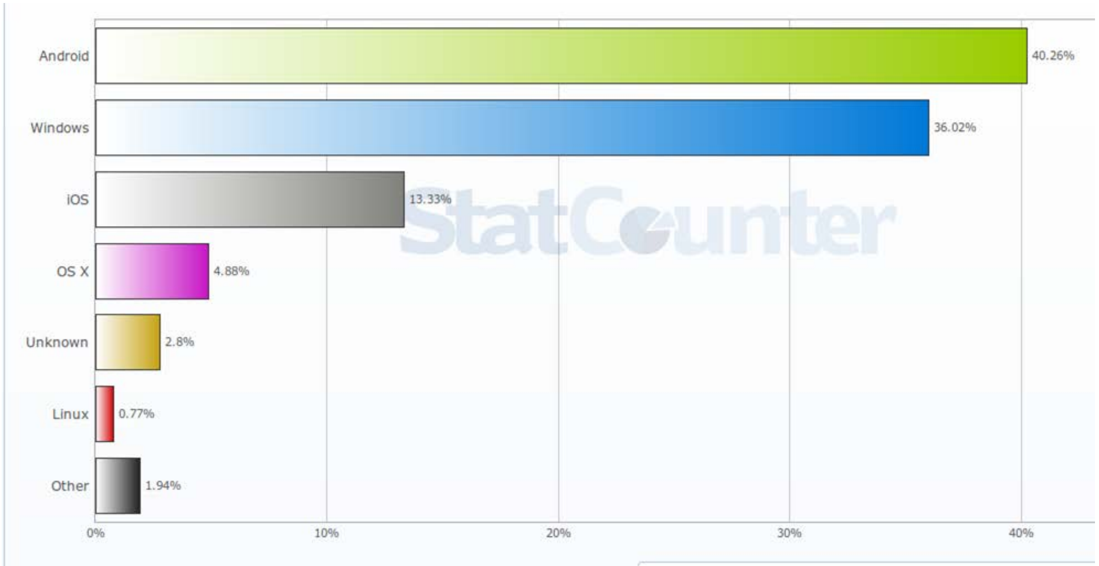
fSO

- OS use statistics
  - *Gartner* [1] (2015)
    - *Smartphones, tablets, laptops and PCs together*



Device shipments, 2015		
Android	🟩	54.16%
iOS/macOS	▨	12.37%
Windows	🟦	11.79%
Other	🟥	21.66%

  - *Statcounter* [2] (2017)



Fuente:

[1] Wikipedia <https://en.m.wikipedia.org/wiki/Usage_share_of_operating_systems>

[2] Statcounter. Operating System Market Share Worldwide. Junio 2017. <http://gs.statcounter.com/#desktop-os-ww-monthly-201508-201508-bar>

Operating system concept

- Android
  - *Android Runtime + kernel GNU/Linux*
  - *Apache License 2.0* + GNU GPL v2
  - “Android Terminal Emulator", …
- IOS/macOS (Mac OS X, OS X)
  - *Darwin + kernel XNU*
  - *Closed source* (*with open source components*) ← *NeXTSTEP, BSD, FreeBSD, Mach, …*
  - “Terminal” (*bash*)
- Windows
  - *Universal App Platform + kernel Windows NT*
  - *Closed / shared source*
  - “PowerShell” = “Windows PowerShell” (privativo) +
    “PowerShell Core” (abierto → GitHub)

Operating system concept

Category	Linux	Unix and Unix-like	Windows	In-house	Other
Desktop, laptop (excluding Android and Chrome OS)	2.18% (Ubuntu, etc.)	6.43% (macOS)	91.39% (10, 8.1, 7, Vista)		
Smartphone, tablet	68.31% (Android)	23.35% (iOS)	1.25% (Windows 10 Mobile, Windows Phone 8.1 and older)		9.86%
Server (web)	66.6% (Ubuntu 35.8%, Debian 31.9%, CentOS 20.6%, Red Hat 3.3%, Gentoo 2.7%, Fedora 0.9%)	1% (BSD)	33% (Windows Server 2016, W2K12, W2K8)		
Supercomputer	99.79% (Custom)	0.21%			
Mainframe	28% (SLES, RHEL)	72% (z/OS) UNIX System Services			
Gaming console, Handheld game console (7th & 8th generation only)		34.1% (PS4, PS3, Vita, PSP) ?? PS4 Obis OS (FreeBSD) ??% N Switch (FreeBSD)	16.36% (Xbox One, Xbox 360)	49.54% (Wii U, Wii, 3DS, DS)	0%
Embedded (automotive, avionics, health, medical equipment, consumer electronics, intelligent homes, telecommunications)	29.44% (Android plus other non-Android Linux)	4.29% (QNX)	11.65% (WCE 7)	13.5%	41.1%

Source: Wikipedia <https://en.m.wikipedia.org/wiki/Usage_share_of_operating_systems>

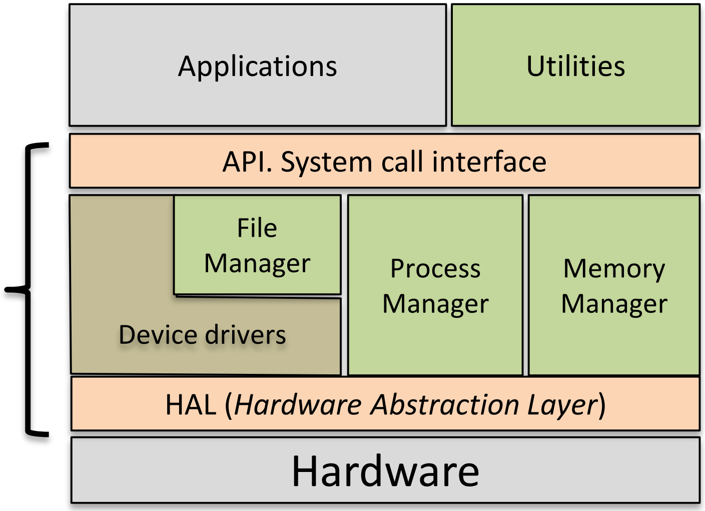
Content

- Operating system concept
- **Operating system structure**
- CPU utilization
- Historical evolution

Terms:	
OS	Operating system
I/O	Input / Output
CPU	Centra processing unit (processor)
API	Application Programming Interface
HAL	Hardware abstraction layer
UNIX	Portable, multitask and multiuser operating system
Linux	Free and open SO kernel based on UNIX

Operating system structure

- **Kernel**
  - File Systems
  - Memory Manager
  - Process Manager
  - Device drivers



- **System Utilities**:
  - They extend the OS providing key utilities not included in the OS kernel
    - Shell, GUI, Monitoring, Maintenance, Administration …

Operating system structure

- Kernel architectures
  - **Microkernel**: provides only the basic hardware abstractions and minimal services.
    - The resource usage policies are implemented as "servers" that run in user space. It has been much debate about their efficiency problems.
    - Examples: Mach, QNX
  - **Monolithic**: All kernel components are in the same address space.
    - Only one program contains the whole kernel functionality (it has to be recompiled after every change).
    - Example: Linux
  - **Hybrid**: This is a modified microkernel that includes not essential components whose execution speed is critical.
    - Examples : Windows NT, XNU (Mac OSX)
  - ….

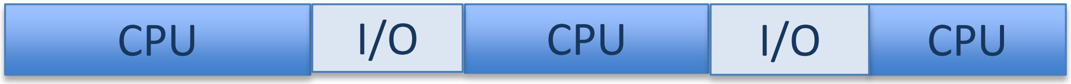
Content

- Operating system concept
- Operating system structure
- **CPU utilization**
- Historical evolution

Terms:	
OS	Operating system
I/O	Input / Output
CPU	Centra processing unit (processor)
API	Application Programming Interface
HAL	Hardware abstraction layer
UNIX	Portable, multitask and multiuser operating system
Linux	Free and open SO kernel based on UNIX

CPU utilization

- **System workload**
  - The workload of a computer system consists of a set of programs to be executed
  - In a simplified description a program execution can be seen as a sequence of CPU and I/O bursts
    - CPU burst → time interval required to perform consecutive CPU operations by a program
    - I/O burst → idem with I/O operations



- CPU bound and I/O bound programs
  - A program may be **bound by the CPU speed**
  - A program may be **bound by the I/O speed**

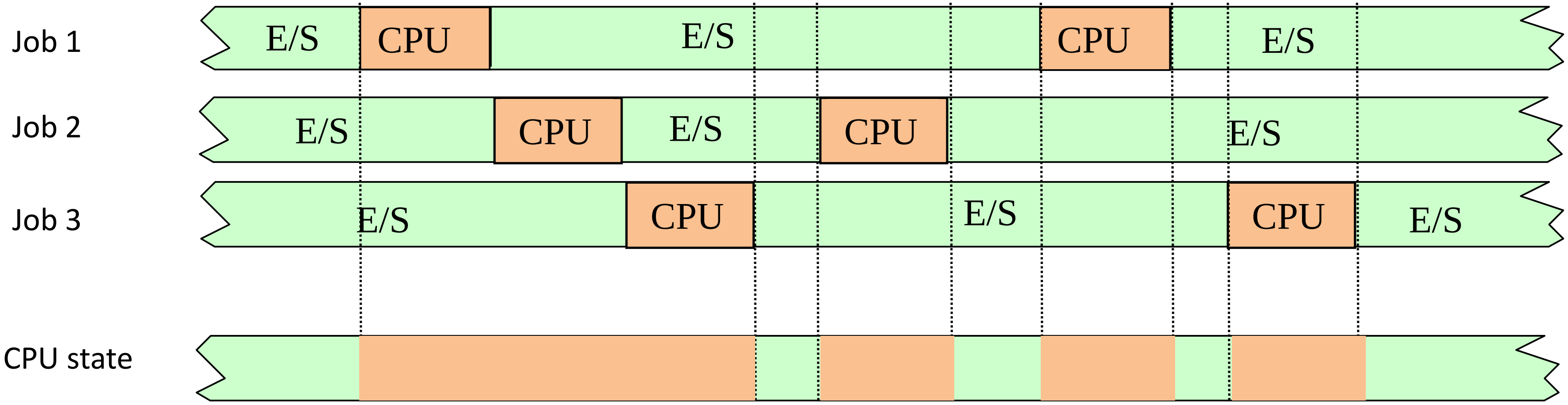
CPU utilization

- **CPU utilization concept**
  - The CPU is the main computer component
  - OSs have to achieve that the CPU be active as much as possible
  - **CPU utilization**: Fraction of time when the CPU is active in relation to the whole required to end the system tasks

$$\textbf{CPU\_utilización} = \frac{\textbf{CPU busy time}}{\textbf{Total time}}$$

CPU utilization

- Multiprogramming
  - Alternative use of the CPU by running programs
    - When a process is blocked waiting for a pendent I/O operation, the CPU executes instructions from another ready process
    - A "context switch" is performed when an I/O operation is demanded
  - CPU utilization increases
  - The system performance increases: more jobs end with less time



Content

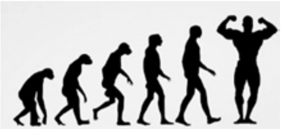
- Operating System Concept
- Operating System Structure
- CPU utilization
- **Historical evolution**

Terms:	
OS	Operating system
I/O	Input / Output
CPU	Centra processing unit (processor)
API	Application Programming Interface
HAL	Hardware abstraction layer
UNIX	Portable, multitask and multiuser operating system
Linux	Free and open SO kernel based on UNIX

Historical evolution

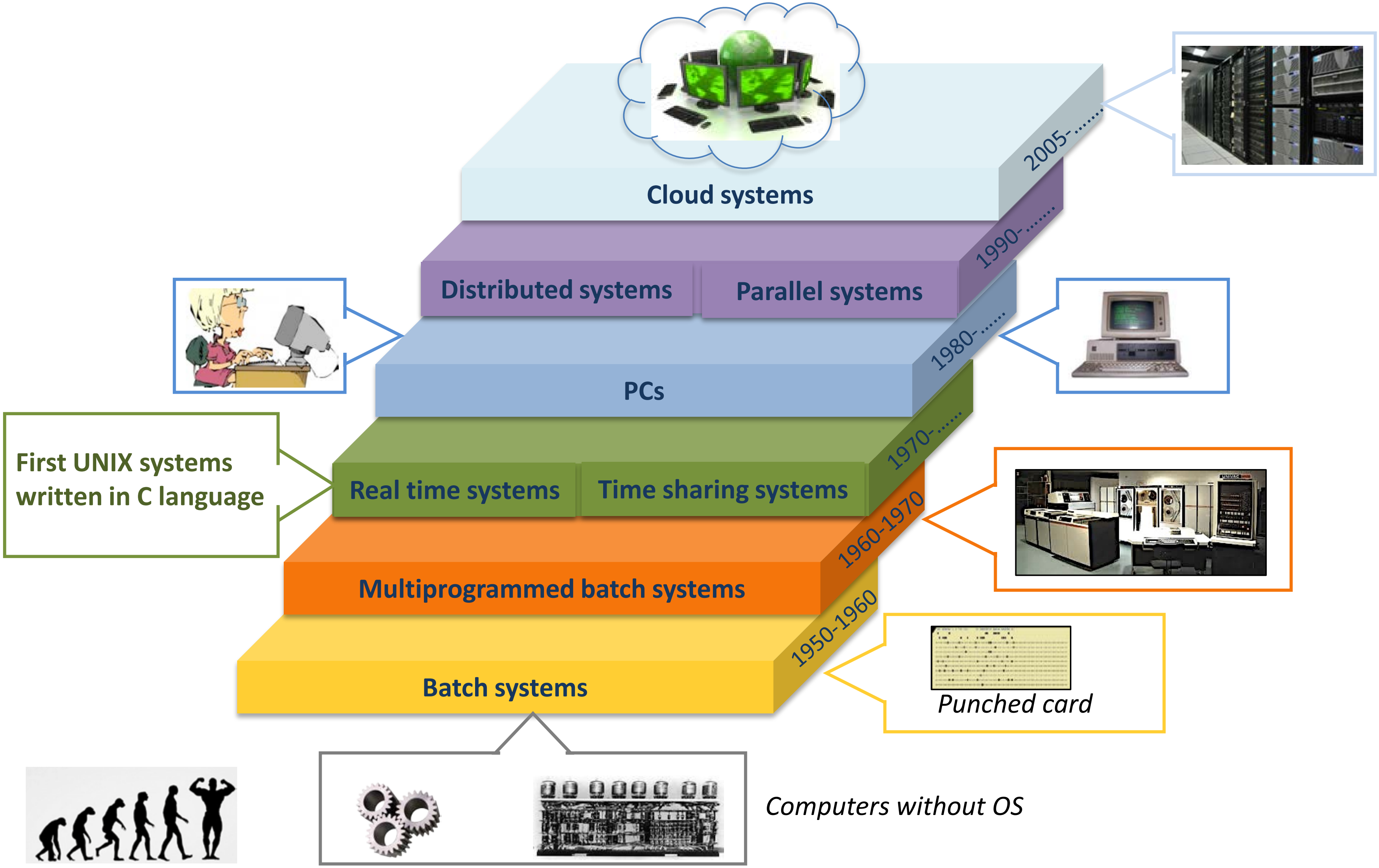
- OS capabilities

Single user: Only one system can be active at a given time	Active users support	**Multiuser**: Several users can be active at the same time
No direct user-machine interaction: Only dedicated machine operators can directly deal with the system	Direct user-machine operation support	**Direct user-machine interaction:** Users can dialog with the system posting commands and waiting form immediate response
Text mode: Users interact with the system with text commands and receiving text response	User interface (UI)	**Graphic mode**: The system offers a graphic user interface (GUI) made of windows, icons and menus
Single task: The OS performs tasks sequentially, a task must wait to the previous one to finish before starting	Tasks support	**Multitask:** Several tasks can be active in the system simultaneously
Single processor: The OS can only work with one CPU	Processors support	**Multiprocessor**: The OS can work with several CPUs simultaneously



Historical evolution

- The OS evolution is conducted by the technological innovations on computer architecture, communications and storage media



Cloud systems

2005-…….

Distributed systems

Parallel systems

1990-…….

PCs

1980-…….

First UNIX systems written in C language

Real time systems

Time sharing systems

1970-…….

Multiprogrammed batch systems

1960-1970

Batch systems

1950-1960

Punched card

Computers without OS

Historical evolution

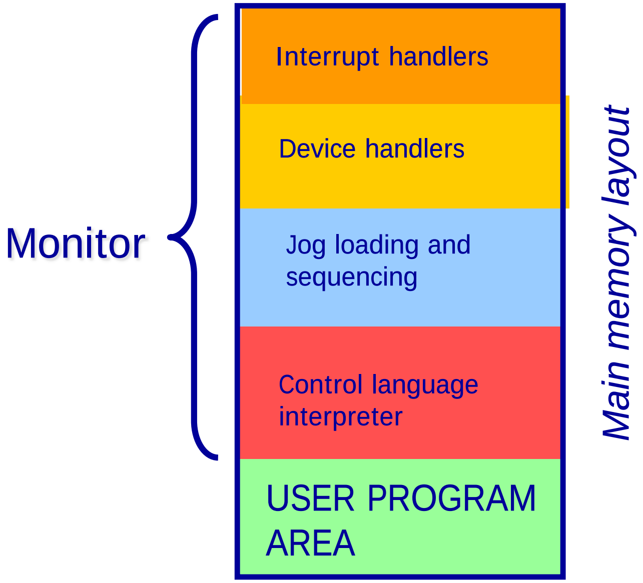
- First OSs: Batch systems

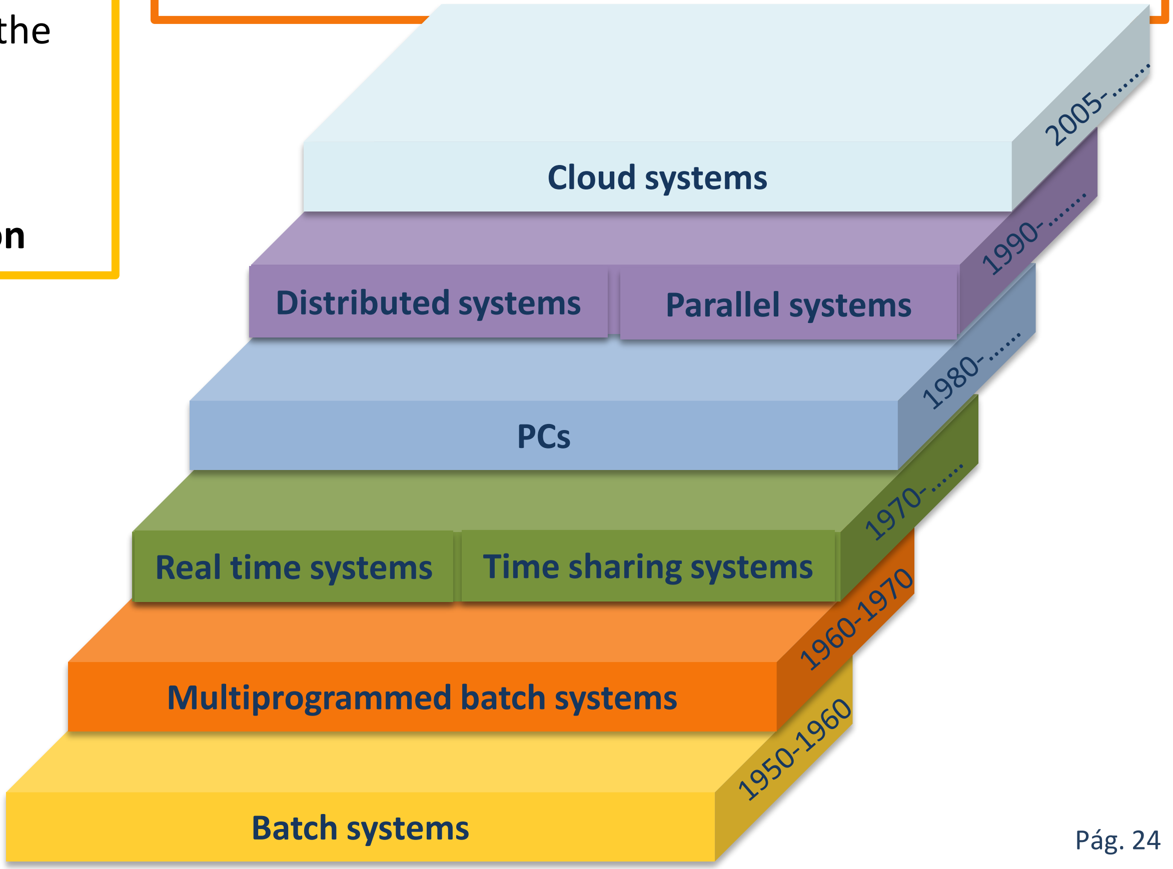
Basic batch systems

- Jobs are processed sequentially: the CPU is idle when the active jobs is performing I/O
- Low CPU utilization
- Resident monitor that automatizes some tasks: job ending, error treatment, loading and executing the next job
- Batch processing
- I/O Access
- **No direct user-machine interaction**

Multiprogrammed batch systems

- Job/CPU scheduling
- Multiprogramming
- Memory management and protection based on fixed memory partitions
- Disk Management
- **No user-machine interaction**





Historical evolution

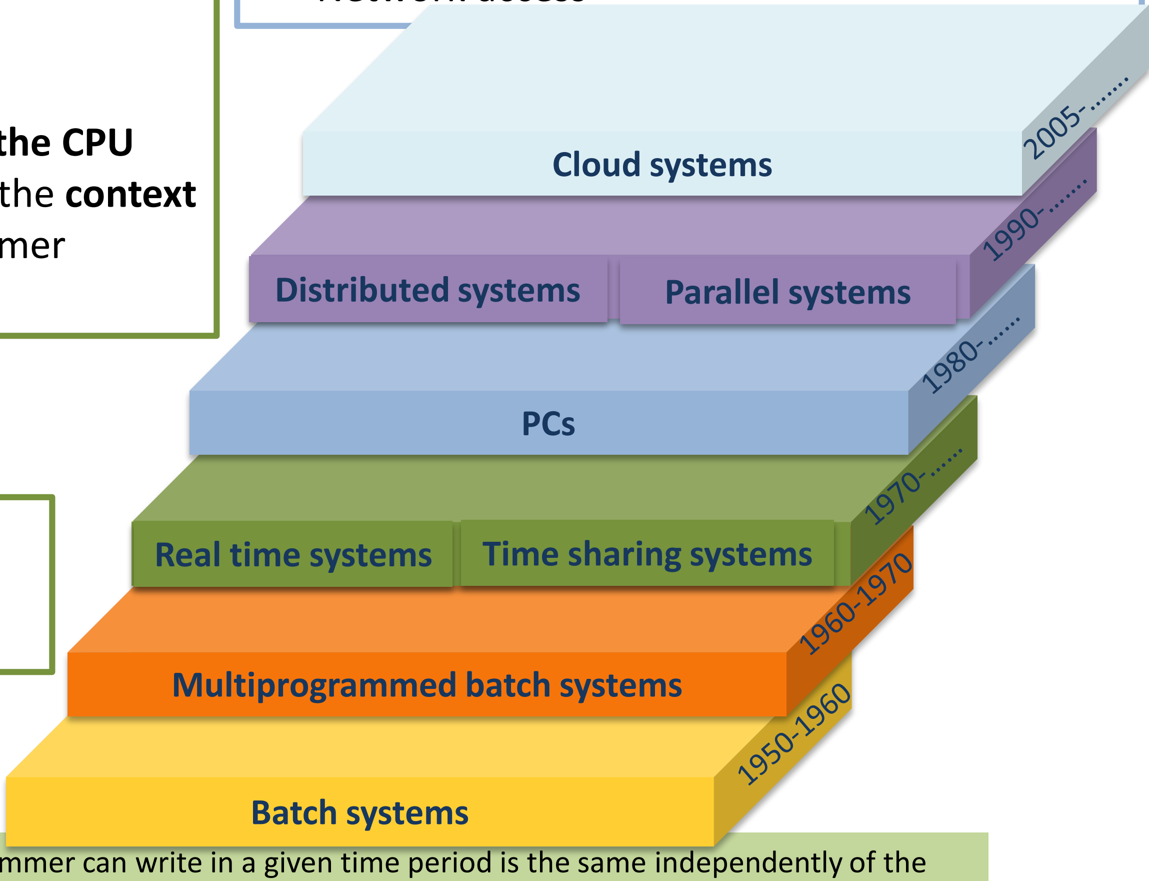
- Modern OSs

Time sharing systems

- **Direct user-macnine interaction with multiprogramming**
- Jobs synchronization and communication
- File systems that manage files
- Protection
- Virtual memory
- Process scheduler: The OS **limits ethe CPU occupancy** by a process by means the **context switch** mechanism that relies on timer interrupts

PC systems

- Personal use
- Friendly user interfaces based on windows and mouse
- Multimedia capabilities
- Plug-and-play support
- Network access



Real time systems

- For executing tasks with a fixed deadline

Corbato's law: The number of lines that a programmer can write in a given time period is the same independently of the programming language used -> increasing the programming language capabilities the programmers throughput will increase.

Historical evolution

- Modern OSs (cont.)

Parallel systems

- Multiprocessor (Multicore):
  - ➢ Several processors/cores coupled by shared memory
- Reliability

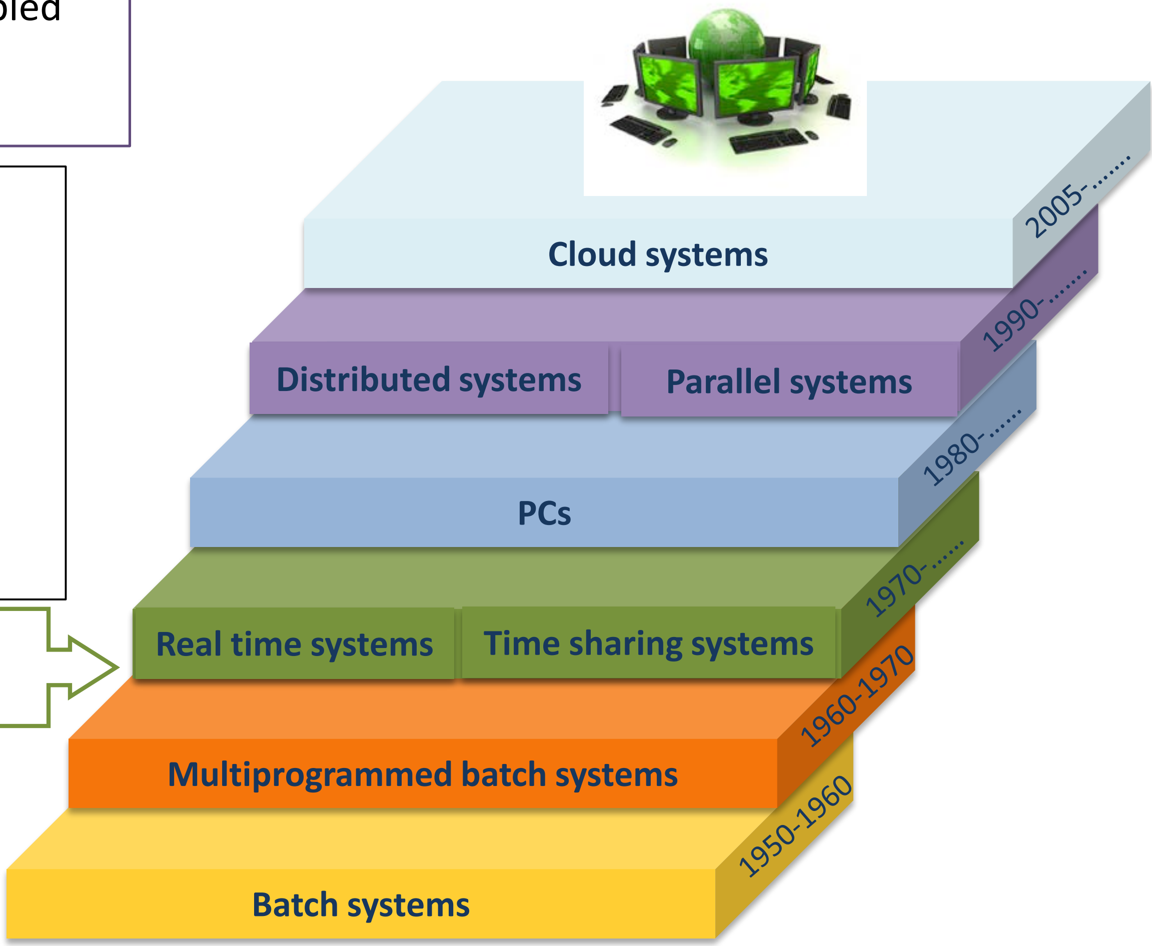
Distributed systems

- The whole computation is distributed among several computers connected with a network
- Internode communication
- Resource sharing
- Workload sharing

Cloud systems

- Storage and computation as a service

First UNIX written in high level language (C)



Historical evolution

- UNIX and C origin

1968
- Software crisis

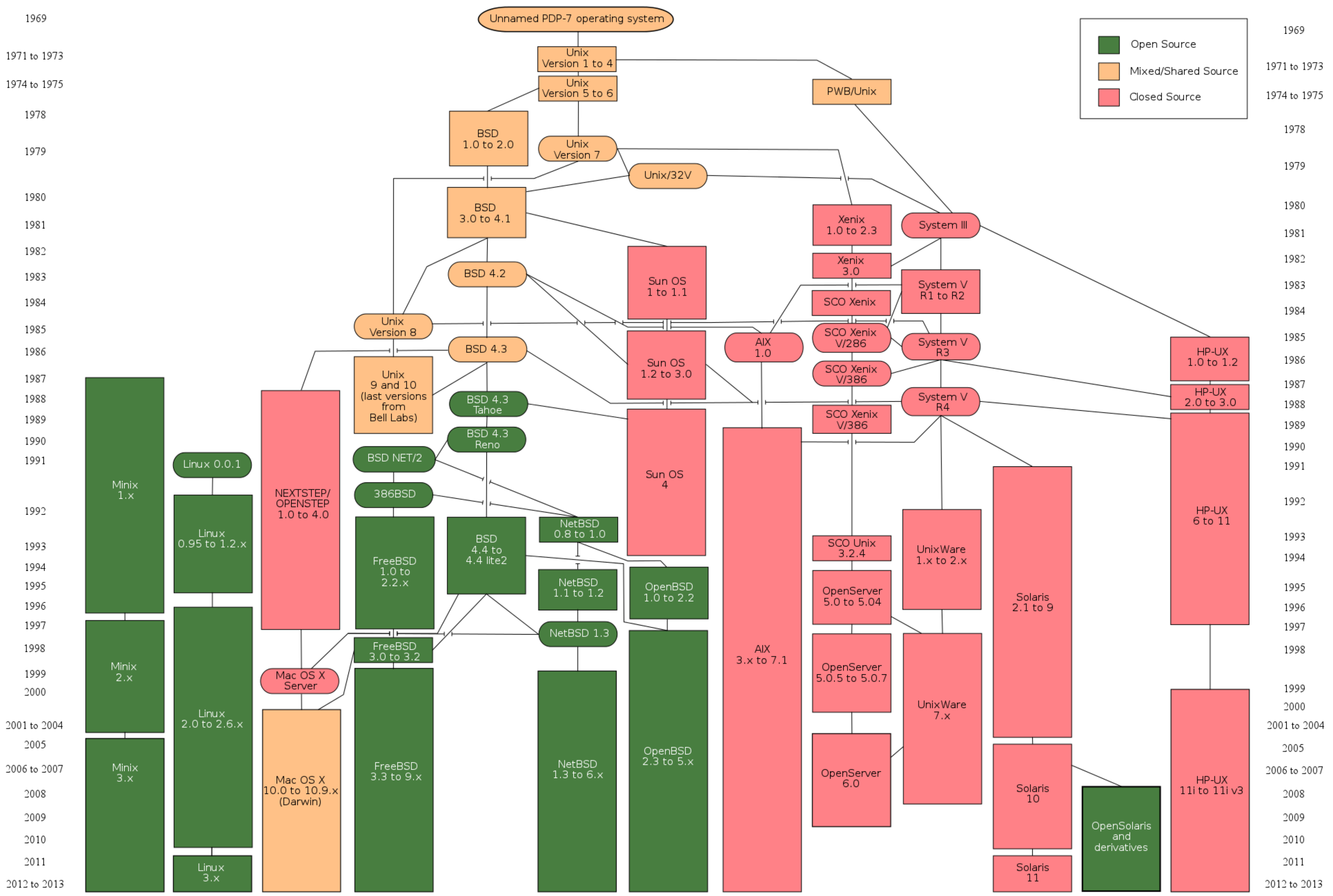
1969
- First UNIX systems
  - SO Written in high level language C

1988
  - First version of POSIX standard IEEE 1003: Standardization of the system calls interface and other UNIX components. Interoperability at the source code level

Dennis Ritchie and Ken Thompson working on PDP-11 computers during the early development of Unix

1975
- Incorporation of virtual memory addressing in the PDP-11 processor
  - Digital Computer (DEC) VAX 11/780 with VAX11/VMS OS (VMS: Virtual Memory System)

UNIX chronicle



🟩 Open Source

🟧 Mixed/Shared Source

🟥 Closed Source

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 2004 · 2005 · 2006 to 2007 · 2008 · 2009 · 2010 · 2011 · 2012 to 2013

Unnamed PDP-7 operating system

Unix Version 1 to 4

Unix Version 5 to 6

PWB/Unix

BSD 1.0 to 2.0

Unix Version 7

Unix/32V

BSD 3.0 to 4.1

Xenix 1.0 to 2.3

System III

Xenix 3.0

Sun OS 1 to 1.1

BSD 4.2

SCO Xenix

System V R1 to R2

Unix Version 8

SCO Xenix V/286

System V R3

AIX 1.0

HP-UX 1.0 to 1.2

BSD 4.3

Sun OS 1.2 to 3.0

SCO Xenix V/386

Unix 9 and 10 (last versions from Bell Labs)

HP-UX 2.0 to 3.0

BSD 4.3 Tahoe

System V R4

SCO Xenix V/386

BSD 4.3 Reno

Minix 1.x

NEXTSTEP/ OPENSTEP 1.0 to 4.0

BSD NET/2

Sun OS 4

Linux 0.0.1

386BSD

HP-UX 6 to 11

NetBSD 0.8 to 1.0

BSD 4.4 to 4.4 lite2

FreeBSD 1.0 to 2.2.x

Linux 0.95 to 1.2.x

SCO Unix 3.2.4

UnixWare 1.x to 2.x

Solaris 2.1 to 9

NetBSD 1.1 to 1.2

OpenBSD 1.0 to 2.2

OpenServer 5.0 to 5.04

AIX 3.x to 7.1

NetBSD 1.3

FreeBSD 3.0 to 3.2

Minix 2.x

Mac OS X Server

OpenServer 5.0.5 to 5.0.7

UnixWare 7.x

Linux 2.0 to 2.6.x

Mac OS X 10.0 to 10.9.x (Darwin)

FreeBSD 3.3 to 9.x

NetBSD 1.3 to 6.x

OpenBSD 2.3 to 5.x

OpenServer 6.0

Solaris 10

HP-UX 11i to 11i v3

Minix 3.x

OpenSolaris and derivatives

Linux 3.x

Solaris 11

Historical evolution

- The birth of personal computer

1964 – *¿Programma 101* → *programmable calculator*?

1972 – *¿Xerox Alto* → *Alto Executive Workstation*?

1975 – *¿Altair 8800 Kit*?
CP/M, Altair BASIC, ..

1977 – *Apple II* → intérprete BASIC
¿DISER Lilith (*Workstation*) → Oberon?

1980 – 86-DOS /x86DOS /QDOS

1981 – IBM PC → PC-DOS / MS-DOS

1982 – Amstrad CPC → CP/M
ZX Spectrum → Sinclair BASIC
Commodore 64 → GEOS

1984 – Apple Macintosh → MacOS
Commodore Amiga 1000 → AmigaO
Amstrad PCW → CP/M Plus

1985 – Atari ST → Atari T

Fuente: imágenes de <http://www.wikipedia.com >

Exercise

- Try to find the version of the Linux kernel installed and active in your system, and the Linux distribution, using the shell commands `uname` and `lsb_release`.

a) Find the kernel version running the command:

```
$ uname -rs
```

b) Find the SO name with the command:

```
$ uname -o
```

c) Find the processor architecture with the command:

```
$ uname -m
```

d) Find the SO distribution version with the command:

```
$ lsb_release -i
```

WARNING The character $ at the beginning of every command it the UNIX prompt **you don’t have to write it**

WARNING You can use the commands `man uname` and `man lsb_release` to get help