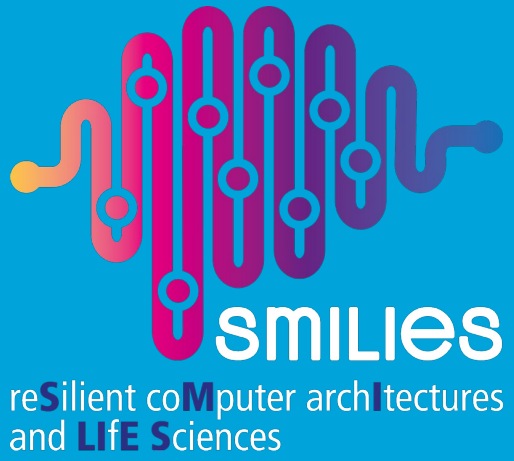


Politecnico
di Torino

Department of Control and
Computer Engineering

INTRODUCTION TO EMBEDDED LINUX

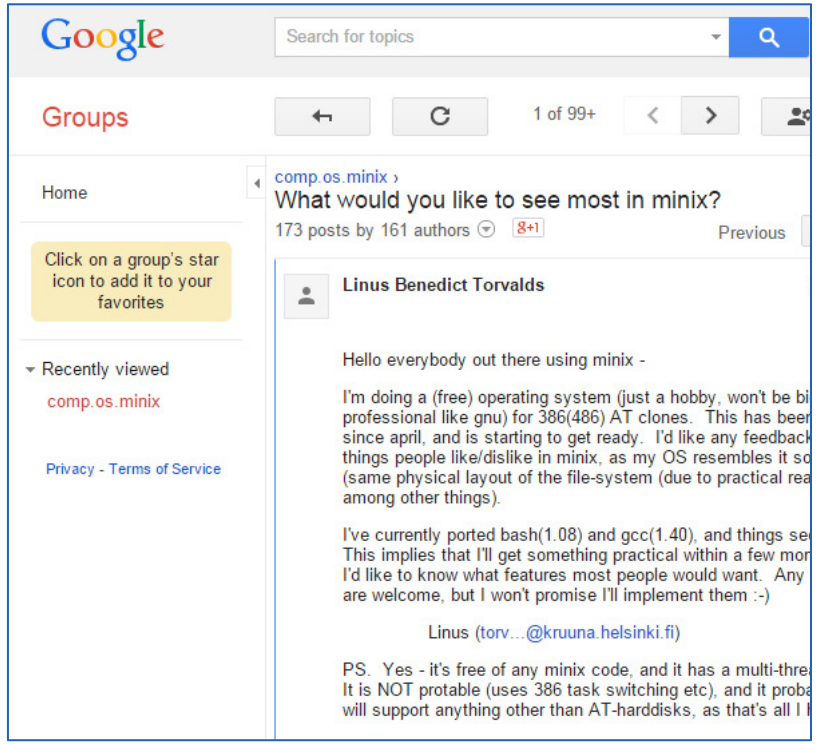
STEFANO DI CARLO

WHY LINUX-BASED EMBEDDED SYSTEMS?

- Open Source (under GNU General Public License v2.0 : GPLv2)
  - The full source code is available for learning and adaptation
- Engaged community maintaining and improving Linux regularly
  - Companies
  - Individuals
  - Academics
  - Hobbyists
- Flexible and adaptable: supports many hardware/System-on-Chip (SoC) configurations
  - Based on ARM, x86, PowerPC, SPARC, RISC-V, etc.
- Proven in many different scenarios (see next slides)
- Supported by a very large ecosystem of software
  - Bootloader, system programs, networking services, advanced graphic services, etc.
- Royalty-free

LINUX EVOLUTION

- August 26, 1991: everything started with this post to comp.os.minix



- Today several kernel categories exist, including:
  - Prepatch or "RC" kernels, which are pre-releases maintained and released by Linus Torvalds.
  - Mainline kernel is maintained by Linus Torvalds, and is where all new features are introduced. New mainline kernels are released every 2-3 months.
  - Long-term kernels are older releases subject to “long-term maintenance”. Important bug fixes are applied to such kernels.

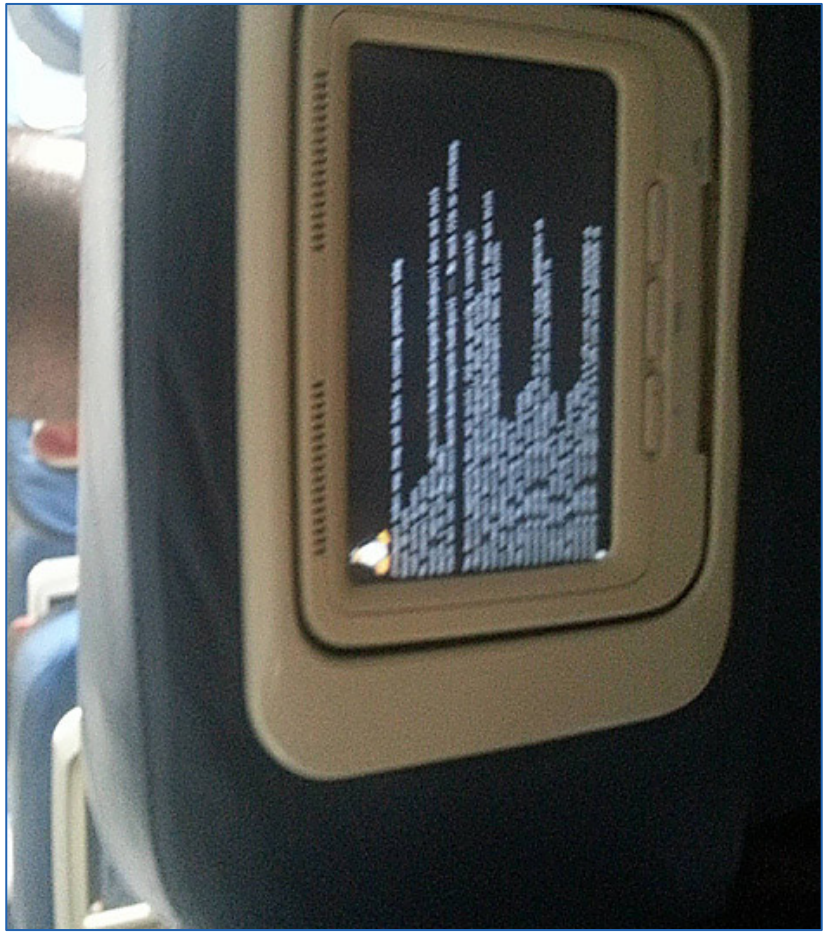
Longterm release kernels

Version	Maintainer	Released	Projected EOL
4.4	Greg Kroah-Hartman	early 2016	Feb, 2018
4.1	Greg Kroah-Hartman	2015-06-21	Sep, 2017
3.18	Sasha Levin	2014-12-07	Jan, 2017
3.14	Greg Kroah-Hartman	2014-03-30	Aug, 2016
3.12	Jiri Slaby	2013-11-03	2016
3.10	Greg Kroah-Hartman	2013-06-30	End of 2015
3.4	Li Zefan	2012-05-20	Sep, 2016
3.2	Ben Hutchings	2012-01-04	May, 2018
2.6.32	Willy Tarreau	2009-12-03	Early 2016

LINUX-BASED EMBEDDED SYSTEM: EXAMPLE 1

- In-flight entertainment systems

> "Linux is particularly suited for in-flight entertainment because it's simple, not weighed down by accompanying programs, and easily adaptable to many environments."
>
> http://www.linuxinsider.com/story/The-Flying-Penguin-Linux-In-Flight-Entertainment-Systems-65541.html



LINUX-BASED EMBEDDED SYSTEM: EXAMPLE 2

- Tim Horton's Café and Bake Shop

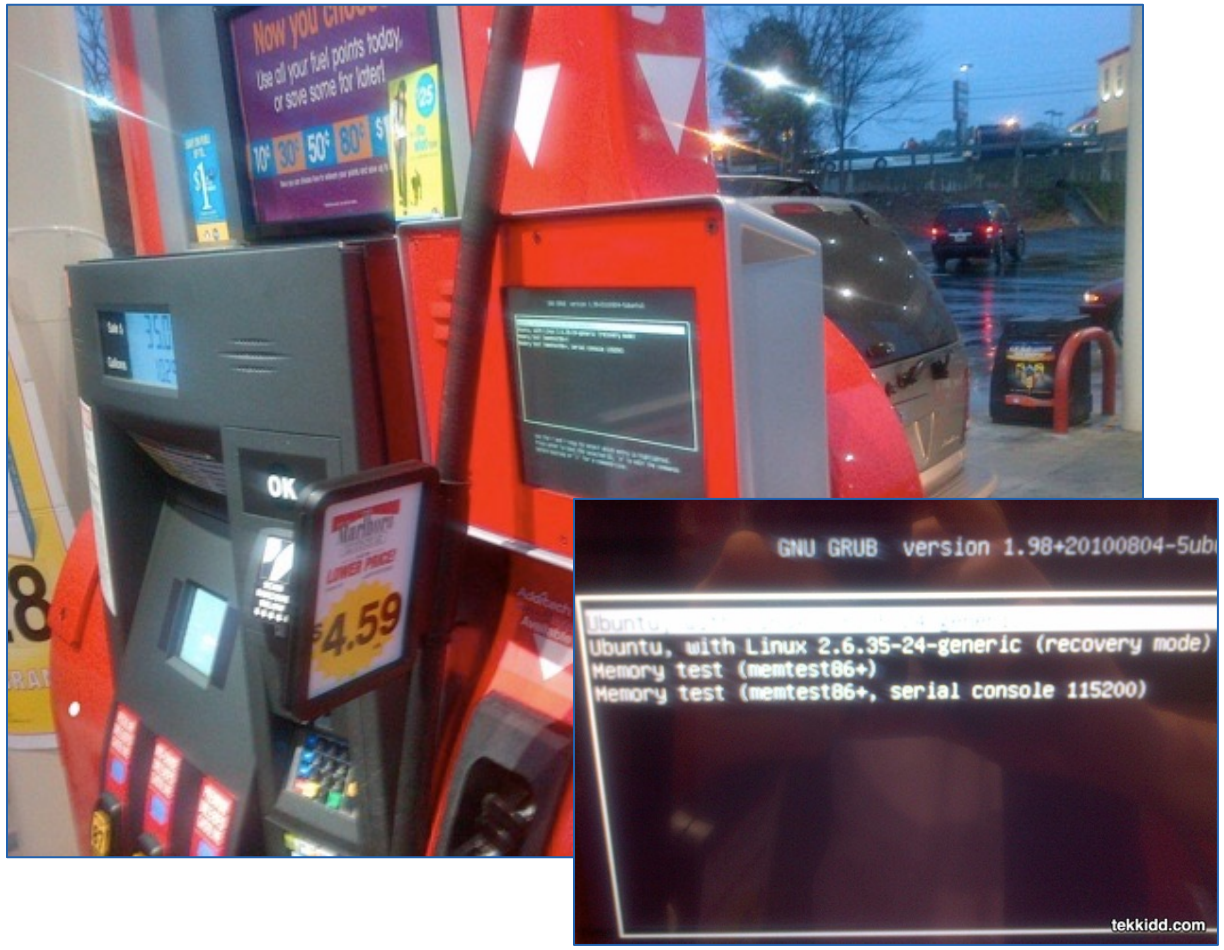
The screen displays the messages Linux produces during boot-up.

We can recognize a kernel panic, as the kernel is not able to find the root file system.

LINUX-BASED EMBEDDED SYSTEM: EXAMPLE 3

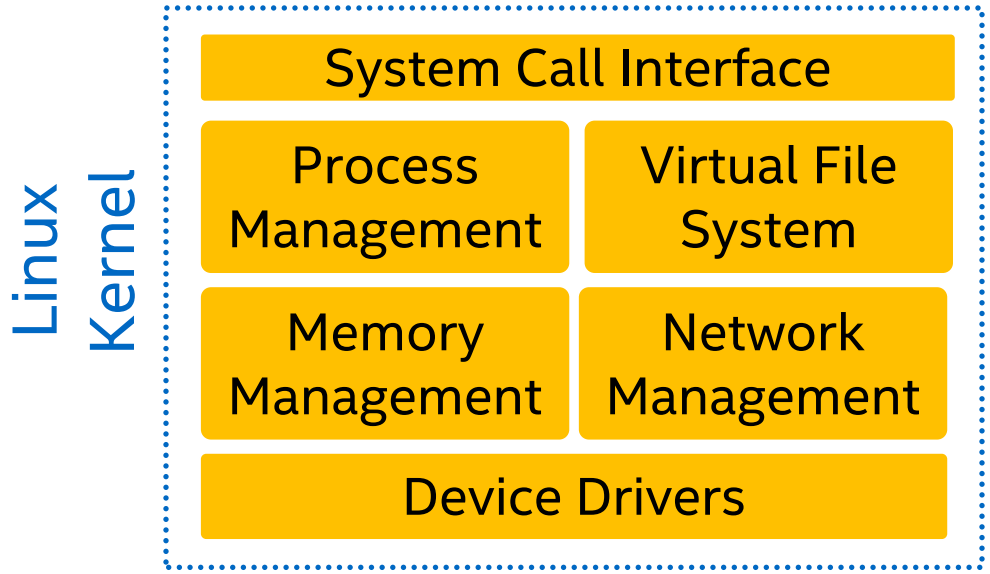
- A gas station pump

The screen displays the messages of a Linux bootloader.
This gas station is powered by Linux Ubuntu distribution with Kernel 2.6.35.



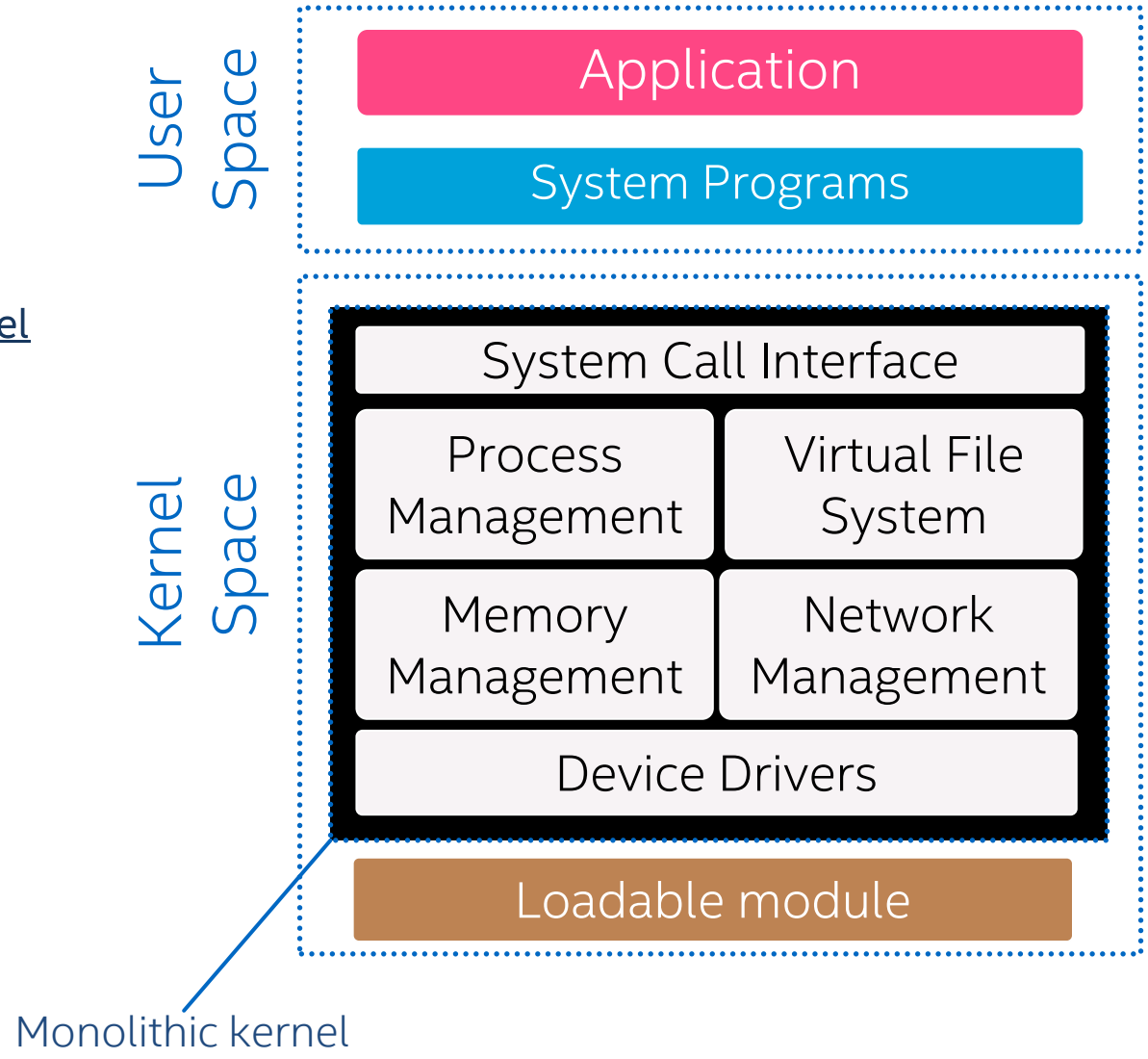
LINUX KERNEL

- It is the software responsible for managing optimally the hardware resources of the embedded system
- It offers services such as:
  - Process management
  - Process Scheduling
  - Inter-process Communication
  - Memory management
  - I/O management (Device drivers)
  - File System
  - Networking
  - ...



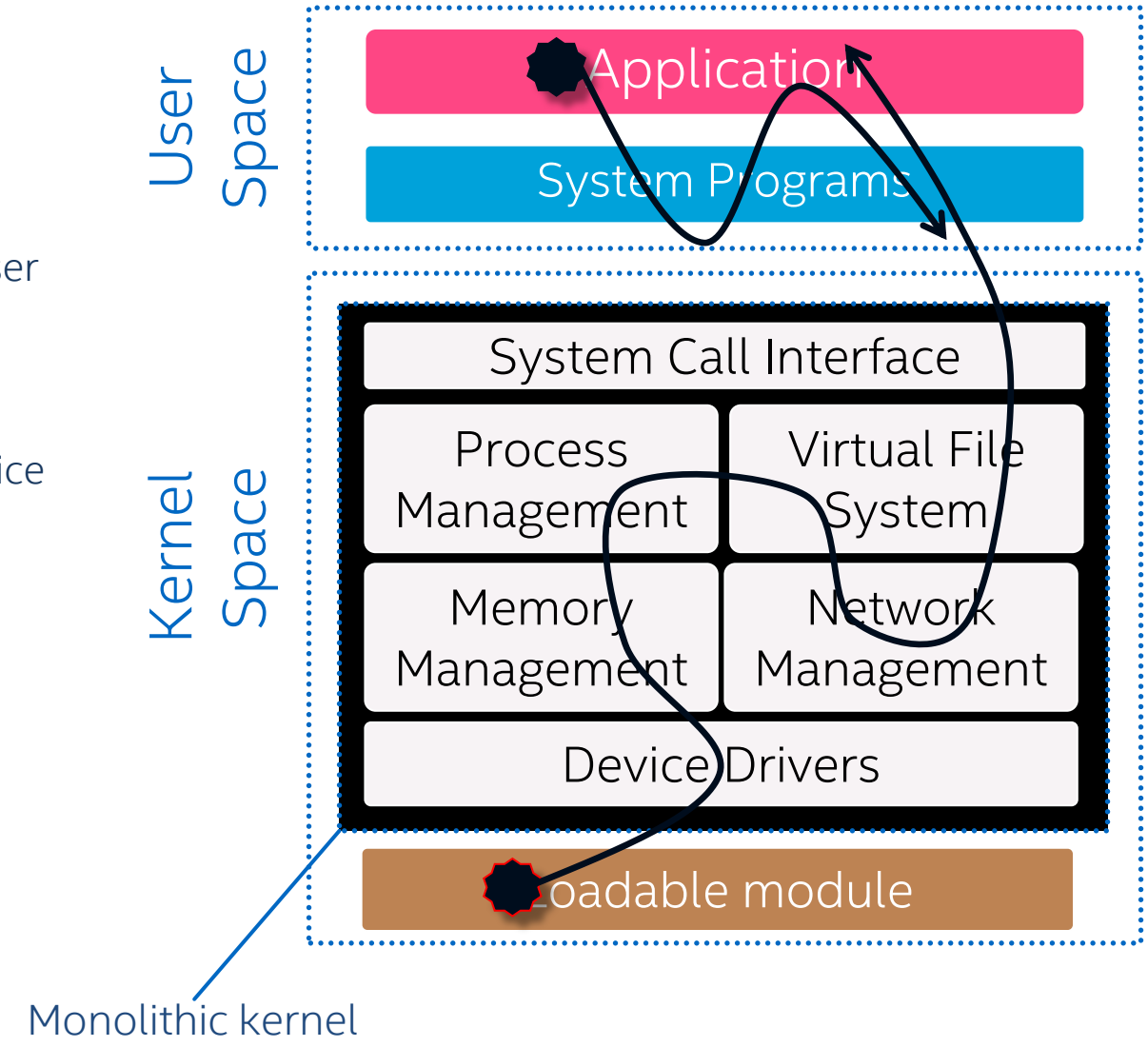
LINUX KERNEL

- It adopts the layered operating system architecture:
  - The operating system is divided into two layers, one (user space) built on top of the other (kernel space)
  - User space and kernel space are different address spaces
  - Basic services are delivered by a single executable, monolithic kernel
  - Services can be extended at run-time though loadable kernel modules



LINUX KERNEL

- Advantages
  - Good separation between Application/System Programs and Kernel, in case of bugs in the User Space the Kernel is not corrupted
- Disadvantages
  - Bugs in one Kernel component (e.g., a new device driver) crashes the whole system



DEVICE TREE

- To manage hardware resources the Kernel must know which resources are available in the embedded system, i.e., the hardware description: I/O devices, Memory, …
- Two solutions are available to provide such information to the Kernel:
  - They are hardcoded into the Kernel binary code; each modification to the hardware definition requires recompiling the source code
  - They are provided to the Kernel by the Bootloader using a binary file, the device tree blob
- The device tree blob (DTB) file is produced starting from a device tree source(DTS)
  - Hardware definition can be changed more easily as recompilation of the DTS is needed, only
  - Kernel recompilation is not needed upon changes to the hardware definition → big time saver

SYSTEM PROGRAMS

- System programs provide a convenient environment for program development and execution
- They can be divided into:
  - File manipulation
  - Status information
  - File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Application programs

APPLICATION

- It is the software providing the service to the user for which the embedded system is conceived for
- Examples can be found in many different products:
  - Network Attached Storage (NAS)
  - Network Router
  - In-vehicle Infotainment
  - Specialized lab equiments
  - ...

ROOT FILE SYSTEM

- The Linux Kernel needs a file system, called Root File System, at startup
  - It contains configuration file needed to prepare the execution environment for the application (e.g., setting-up the Ethernet address)
  - It contains the first user-level process (init)
- The root file system can be:
  - A portion of the RAM treated as a file system known as Initial RAM Disk (initrd), which is the typical case when the embedded system does not have to store data persistently during its operations
  - A persistent storage in the embedded system, which is the typical case when the embedded system has to store data persistently during its operations
  - A persistent storage accessed over the network, which is the typical case during development of a Linux-based embedded system

TYPICAL LAYOUT OF THE ROOT FILE SYSTEM

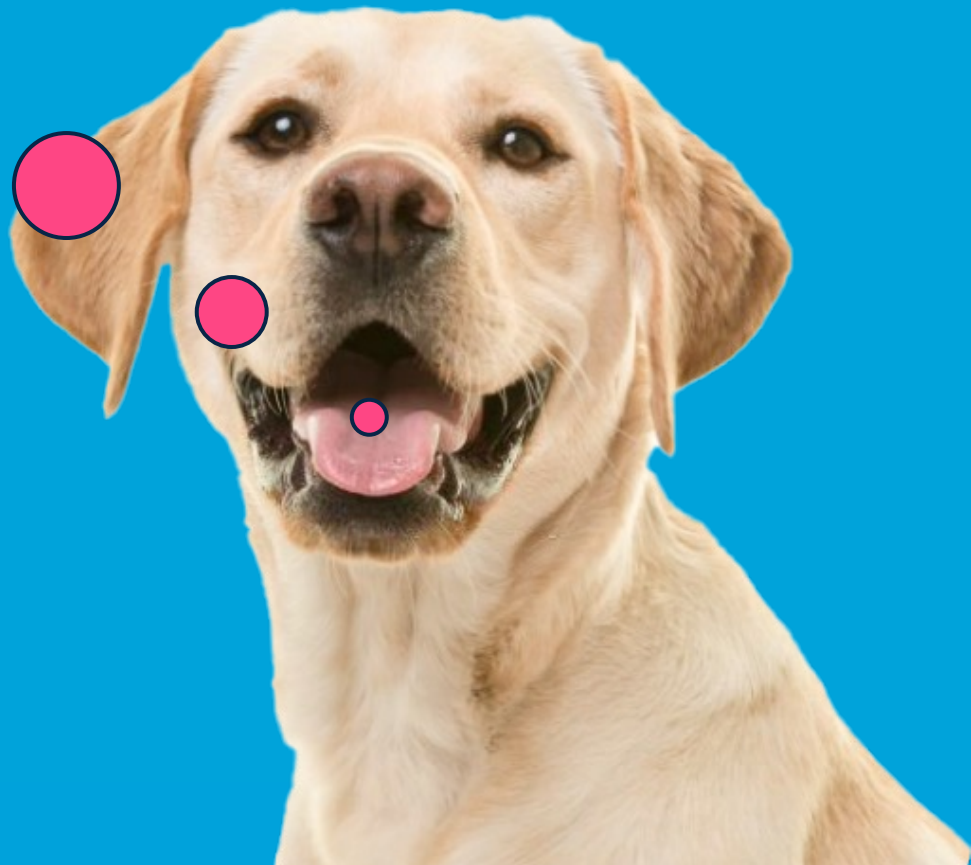
```
/                    # Disk root
     /bin            # Repository for binary files
     /lib            # Repository for library files
     /dev            # Repository for device files
          console c 5 1  # Console device file
          null c 1 3     # Null device file
          zero c 1 5     # All-zero device file
          tty c 5 0      # Serial console device file
          tty0 c 4 0     # Serial terminal device file
          tty1 c 4 1     #
          tty2 c 4 2     #
          tty3 c 4 3     #
          tty4 c 4 4     #
          tty5 c 4 5     #
     /etc            # Repository for config files
          inittab        # The inittab
          /init.d        # Repository for init config files
               rcS  # The script run at sysinit
     /proc               # The /proc file system
     /sbin               # Repository for accessory binary files
     /tmp            # Repository for temporary files
     /var            # Repository for optional config files
     /usr            # Repository for user files
     /sys            # Repository for system service files
     /media              # Mount point for removable storage
```

SMILIES
reSilient coMputer archItectures and LIfE Sciences

Politecnico di Torino
Department of Control and Computer Engineering

DAUIN

QUESTIONS?



THANK YOU!