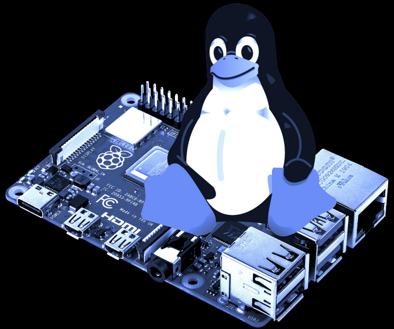
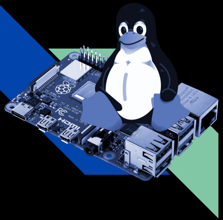


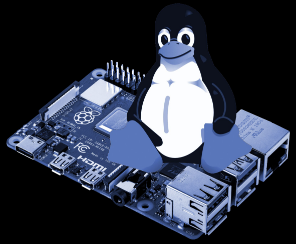
Embedded Linux
General Discussion

Sherif Mousa



Agenda

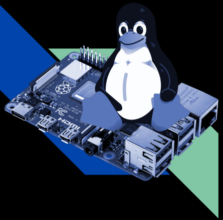
- Introduction (Free Software, GNU/Linux)
- What is Embedded Linux
- Embedded Linux System Components
- Building Embedded Linux Systems
- Mixed Criticality Systems (Hypervisors)



Introduction

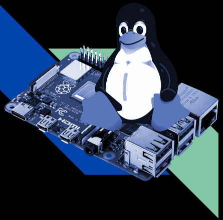
Free Software

GNU/Linux



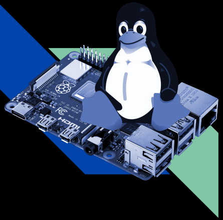
Introduction .. Free Software .. GNU/Linux

- Unix (Ken Thompson & Dennis Ritchie) AT&T company 1970s
- 1983, Richard Stallman: GNU project and concept of "free software". Start of development of gcc, gdb, glibc, etc. developed.
- 1991, Linus Torvalds launches the Linux project, a Unix-like operating system kernel. Together with GNU software and other free software components, it creates a complete and usable operating system: GNU/Linux
- ~ 1995, Linux is more and more widely used on server systems.
- ~ 2000, Linux is more and more widely used in embedded systems.
- ~ 2005, Linux is more and more widely used in desktop systems.



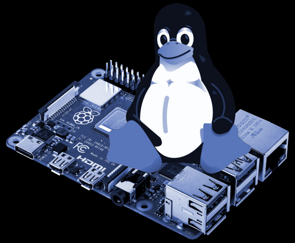
Introduction .. Free Software .. GNU/Linux

- A program is considered free when its license offers to all its users the following freedoms:
  - Freedom to run the software, for any purpose
  - Freedom to study how the software works, and change it
  - Freedom to redistribute copies
  - Freedom to distribute copies of modified versions
- Those freedoms imply that the source code is available, it can be modified to match the needs of a given product, and the result can be distributed to customers. Which is perfect match for Embedded Systems.

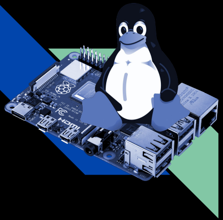


Introduction .. Free Software .. GNU/Linux

- Advantages of open-source in embedded systems
  - Re-using components -> focus on the added value of your product.
  - Low cost -> higher budget for the hardware or increase skills.
  - Full control over the software parts of your system.
  - Quality -> design your system with high-quality components.
  - Community support -> quick solutions for problems during development.
  - Taking part into the community.
- Drawbacks
  - Large choices
  - New skills needed
  - Fear of Licenses

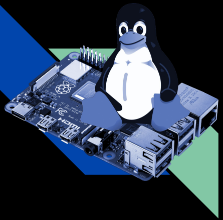


What is Embedded Linux



What is Embedded Linux

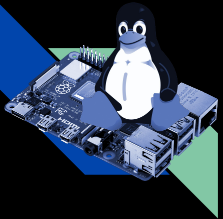
- Embedded Linux is the use of Linux as the main OS for embedded computer systems.
- Adapting the Linux kernel and customizing the user-land libraries and utilities to embedded applications such as those in use in consumer electronics, wearables, military, medical, industrial, network, and auto industries.
- Creating an Embedded Linux based system is like a puzzle, putting the right pieces together will create the final perfect image for your needs.



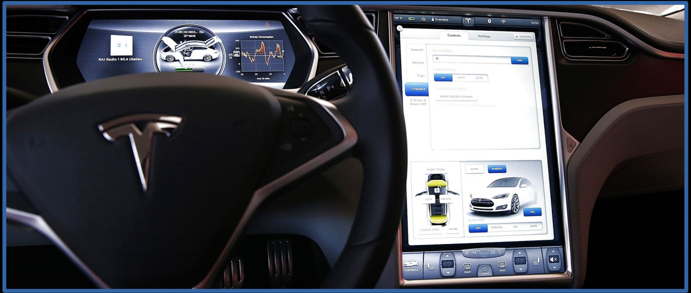
Why Embedded Linux

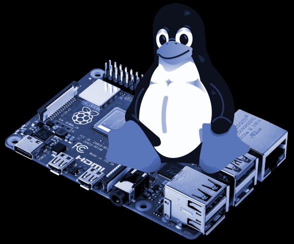
- Why OS for Embedded ?
  - Make use of micro-processor capabilities (Multi-Tasking …)
  - Easy to program.
  - System scalability.
- Why Embedded not Normal OS ?
  - Systems with small and limited resources.
  - Special-Purpose systems.
- Why Linux ?
  - Inexpensive, robust, easy to program, open source.
  - Ported to a variety of CPU architectures (X86, ARM, PowerPc, ...).
  - Large device drivers coverage.

"When you bake it at the factory that's what it does forever" *Tim Bird, Sony Entertainment*

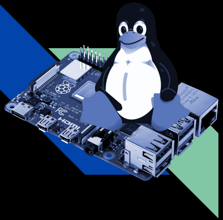


Systems using Linux





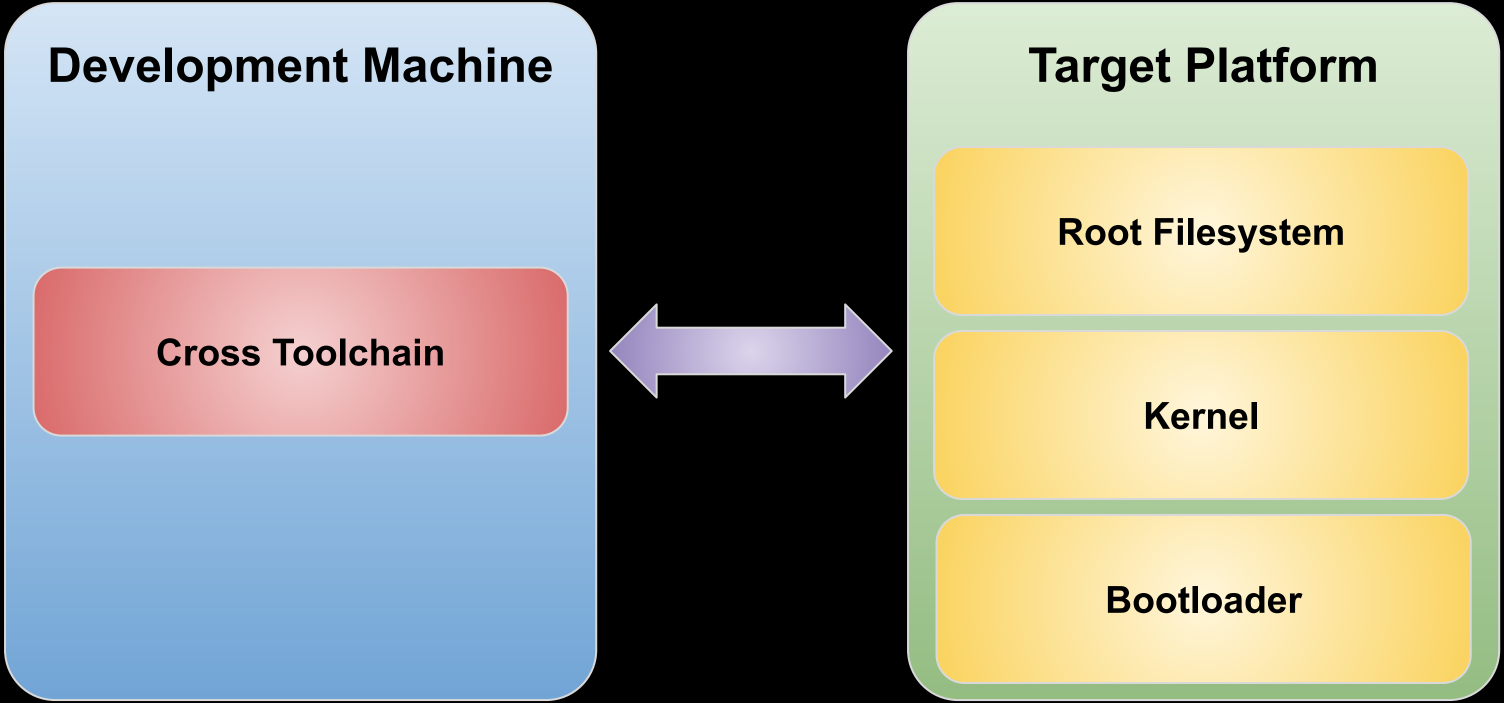
Embedded Linux System Components

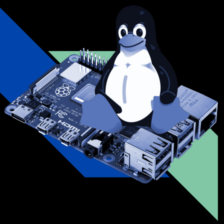


Embedded Linux System Components

- Toolchain: which doesn’t run on the target platform, but allows to generate code for the target from a development machine.
- Bootloader: which is responsible for the initial boot process of the system, and for loading the kernel into memory.
- Linux Kernel: with all the device drivers (all, arch, SoC, board)
- Root filesystem: which contains all the applications (GUI or non-GUI) and libraries.

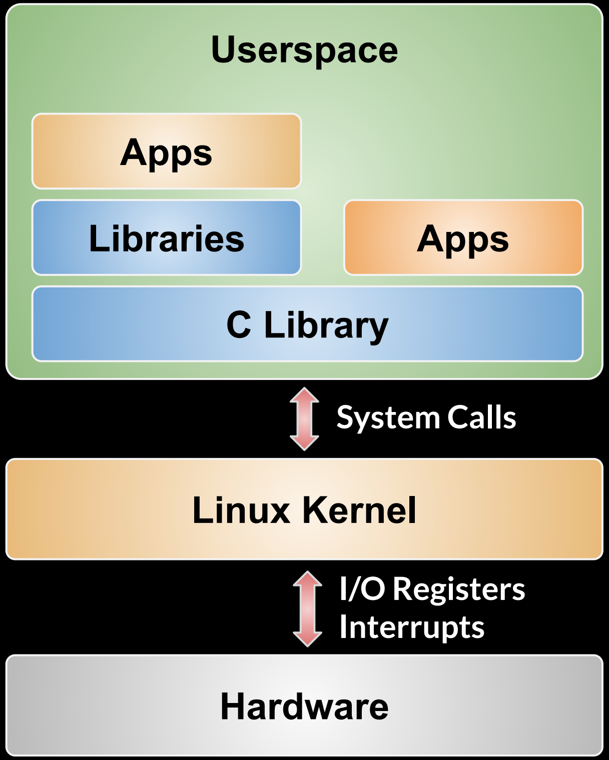
Embedded Linux System Components

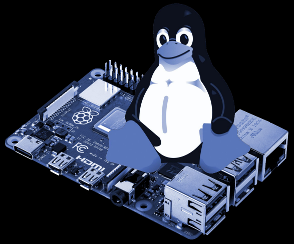




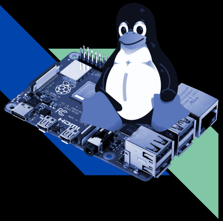
Kernel vs. Userspace

- The Linux kernel runs in privileged mode. It can access and control the hardware. Its role is to multiplex the available resources, and provide coherent and consistent interfaces to userspace.
- Userspace is the set of applications and libraries that run on the system. They work in unprivileged mode. They must go through the kernel to access the system resources, including the hardware. The kernel provides isolation between applications.



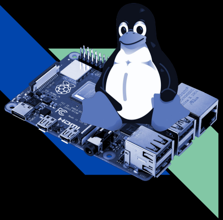


Building Embedded Linux Systems



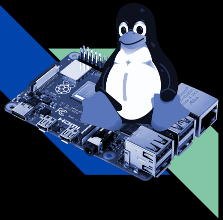
Embedded Linux development process

- **Board Support Package development, BSP**
    - **Base of the system, that heavily depends on the hardware (toolchain, bootloader and Linux kernel)**
- **System Integration (Build systems, CI-CD, ...)**
    - **Integrating all components needed for your system and your custom libraries and applications.**
    - **This involves configuring and cross-compiling a lot of components, with sometimes complex dependencies and/or non-trivial compilation mechanism, especially in a cross-compilation context.**
- **Application Development**
    - **Coding, Building, Integration, Testing, Debugging.**



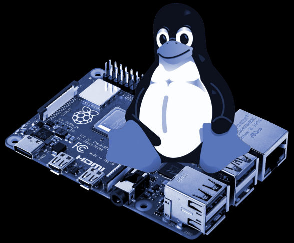
Building Embedded Linux Systems

- From scratch (*Old .. but not obsolete*)
  - The original way, you need to build every single component from scratch, from its source code, applying your configurations and customizations, building each component alone, then integrate all these parts together.
- Auto-Build tools
  - Automated tools in the form of “build scripts”, written by the experts in this field to make it easy for everyone to build a full embedded Linux system with the minimal knowledge of building process.
  - They automatically download, configure, compile and install all the components in the right order.
  - They already contain a large number of packages, that should fit your main requirements, and are easily extensible.

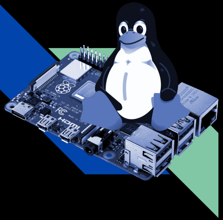


Building Embedded Linux Systems

- **Buildroot**
  - **Set of Makefiles and patches that simplifies and automates the process of building a complete Linux system for an embedded system.**
  - **Useful mainly while working with small or medium systems, using various CPU architectures – including x86, ARM, MIPS, and PowerPC.**
- **Yocto**
  - **It is a Linux Foundation workgroup**
  - **It's a complete embedded Linux development environment with tools, metadata, and documentation - everything you need.**
  - **The Yocto Project has the objective of attempting to improve the lives of developers of customized Linux systems supporting the ARM, MIPS, PowerPC and x86 architectures.**



Mixed Criticality Systems (Hypervisors)

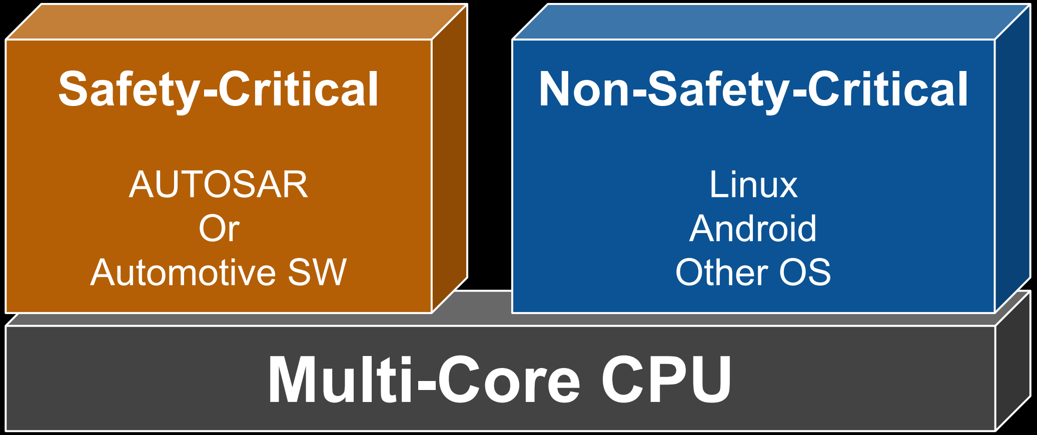


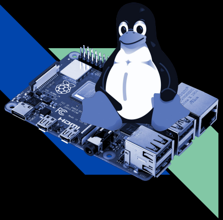
Mixed Criticality Systems

- Multiprocessor systems are now everywhere even on embedded systems.
- Designers of the embedded real-time systems aim to increase the number of functionalities.
- They also want to reduce the number of chips, as increasing the quantity of electronic control units (ECUs) further and thereby also the network and maintenance complexity is neither technically nor economically justifiable.
- This leads to the requirement of executing applications with different levels of criticality on a same multiprocessor chip.

Mixed Criticality Systems

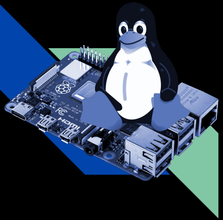
- They are systems on which safety-critical (AUTOSAR for example) and non safety-critical (Linux for example) software must run simultaneously on the same processor.





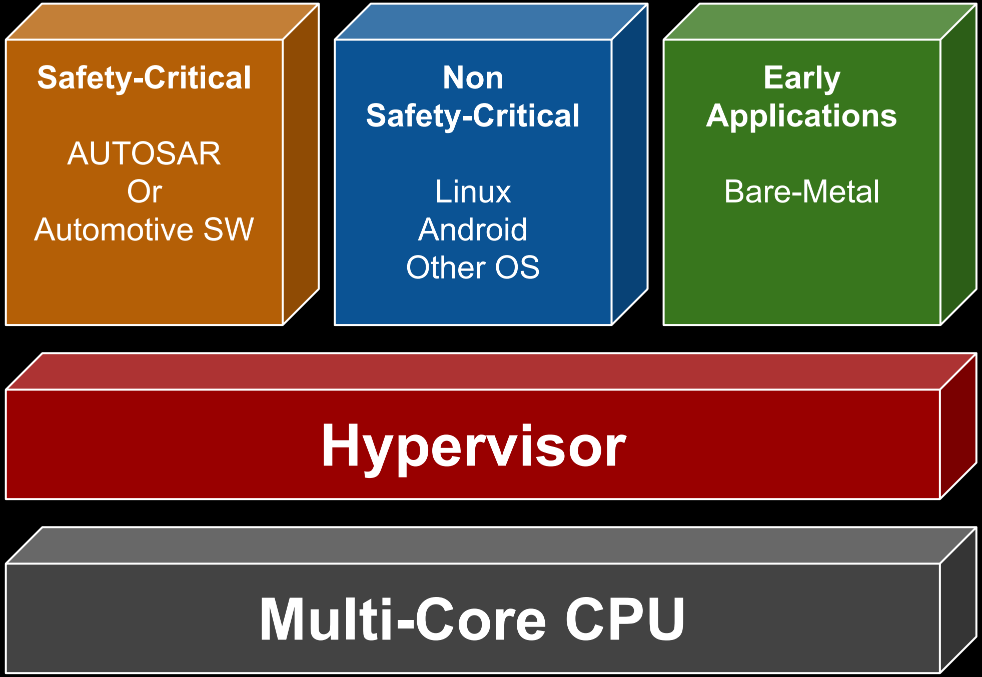
Mixed Criticality Systems

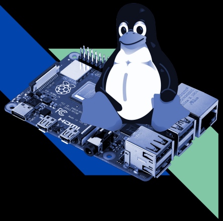
- The critical partition takes care of vehicle network communication and other critical software that requires an RTOS or thin OS.
- On the other hand, media, connectivity, voice features, and other UI-related features are placed in the non-critical partition. This partition runs on a high-end, consumer-based OS such as Linux or Android.



Hypervisors

- Hypervisor is a software layer that enables multiple operating systems to be run simultaneously on a single hardware platform. It offers a higher level of virtualization efficiency and security.
- They have been used for decades on mainframes, more recently on desktop computers, but are now beginning to be very relevant to embedded developers.





Hypervisors

- The Good
  - Increase security
  - Increase flexibility
  - Reduction of Hardware Costs
  - Increased Hardware Efficiency
- The Bad
  - Performance overhead
  - Little practical experience
- The Ugly
  - Single hardware failure could inherently affect more functionality and more applications

- Bootlin Trainings
  - https://bootlin.com/training/
- Embedded Linux Wiki
  - https://elinux.org/Main_Page
  - 2019 ELC presentations https://elinux.org/ELC_Europe_2019_Presentations
- Yocto quick start guide
  - https://www.yoctoproject.org/docs/current/brief-yoctoprojectqs/brief-yoctoprojectqs.html
- Hypervisors
  - http://www.webopedia.com/quick_ref/hypervisor.asp
  - https://www.cs.unc.edu/~jerickso/icess2010-mixedcriticality.pdf
  - http://biblio.telecom-paristech.fr/cgi-bin/download.cgi?id=13235
  - https://blogs.mentor.com/colinwalls/blog/2013/11/11/embedded-hypervisors/

References

Why
What
Where
Who
When
Thank You!