Bootloaders in Limbo

Between Platform Initialization and Operating System

Daniel Maslowski

Hello, I am Daniel :-)

Work and education

- IT security and computer science
- software engineer
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

Hello, I am Daniel :-)

Work and education

- IT security and computer science
- software engineer
- infrastructure and web
- apps, UIs, ecommerce

Open Source contributions

- hardware and firmware
- operating systems
- software distributions
- reverse engineering

I joined RISC-V International as an Individual Member.

Agenda

- Bootloader = Business
  - Satisfying Customers
  - Scopes and Goals
  - Classification
- Projects and Stacks
  - Protocols, Interfaces and Features
  - Platforms, Ports and Flows
  - OS integration
- Success Stories
  - Case Studies
  - Saving Costs

Bootloader = Business

Elevator Pitch

Elevator Pitch

Fast, convenient, safe and secure systems sell best.

Elevator Pitch

Fast, convenient, safe and secure systems sell best.

Choose your components wisely.

What is a Bootloader again?

What is a Bootloader again?

Platform Initialization

aka *firmware*

- SoC
- clocks
- GPIOs
- DRAM controller

Bootloader

today's topic

- needs *flexibility*
- fetches OS kernel
- checks for integrity
- maybe interactive menu

Operating System

- Linux
- FreeBSD
- Plan 9
- Oberon
- Haiku
- …

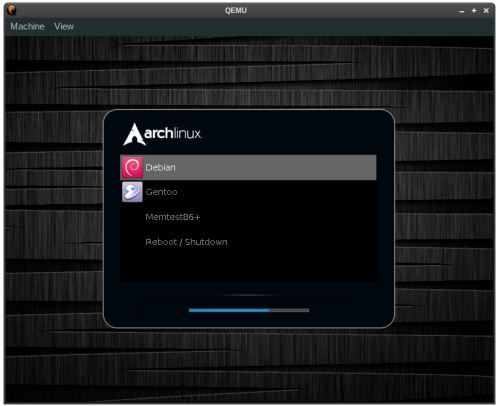
Common Bootloader Functionalities

Common Bootloader Functionalities

A bootloader *is an application that loads and executes another application.*

Common Bootloader Functionalities

A bootloader *is an application that loads and executes another application.*



- target application may rely on a specific protocol
- often configurable via files or customizable at build time
- can offer an interactive menu, e.g. for switching OSs
- GSoC 2023 project: porting GRUB to coreboot for RISC-V
  - only UEFI at this point

image source: https://github.com/hartwork/grub2-theme-preview

Satisfying Customers

Watching the Demand

Watching the Demand

Over time, requirements develop and change.

[1] https://eclypsium.com/blog/supply-chain-risk-from-gigabyte-app-center-backdoor/

Watching the Demand

Over time, requirements develop and change.

Ownership and Control

People have different needs regarding

- how systems boot
- integrating OSes
- platform security

Enterprise customers need to fully control their machines[1].

[1] https://eclypsium.com/blog/supply-chain-risk-from-gigabyte-app-center-backdoor/

Rising Interest in oreboot and LinuxBoot

Rising Interest in oreboot and LinuxBoot

https://www.reddit.com/r/RISCV/comments/13ksvsz/comment/jkm63qh/

Why dom't they collaborate with Oreboot project?

Rising Interest in oreboot and LinuxBoot

https://www.reddit.com/r/RISCV/comments/13ksvsz/comment/jkm63qh/
Why dom't they collaborate with Oreboot project?

https://forum.rvspace.org/t/oreboot-visionfive-2-support-effort/2211

Rising Interest in oreboot and LinuxBoot

https://www.reddit.com/r/RISCV/comments/13ksvsz/comment/jkm63qh/
Why dom't they collaborate with Oreboot project?

https://forum.rvspace.org/t/oreboot-visionfive-2-support-effort/2211

Someone from Intel contacted me, evaluating oreboot and LinuxBoot on RISC-V.

[2] https://www.youtube.com/watch?v=gB3wgOuvLJQ

Rising Interest in oreboot and LinuxBoot

https://www.reddit.com/r/RISCV/comments/13ksvsz/comment/jkm63qh/
Why dom't they collaborate with Oreboot project?

https://forum.rvspace.org/t/oreboot-visionfive-2-support-effort/2211

Someone from Intel contacted me, evaluating oreboot and LinuxBoot on RISC-V.

ByteDance[2] and many hyperscalers are using LinuxBoot.

[2] https://www.youtube.com/watch?v=gB3wgOuvLJQ

Industry Collaboration

[3] https://riseproject.dev/
[4] https://lists.riscv.org/g/tech-prs

Industry Collaboration

Multiple companies, including Intel, Ventana and Rivos, are working together to define UEFI/ACPI[3] and platform runtime services[4] for RISC-V.

[3] https://riseproject.dev/
[4] https://lists.riscv.org/g/tech-prs

Industry Collaboration

Multiple companies, including Intel, Ventana and Rivos, are working together to define UEFI/ACPI[3] and platform runtime services[4] for RISC-V.

RISE
RISC-V Software Ecosystem

The RISC-V Software Ecosystem (RISE) project is a collaborative effort led by industry leaders with a mission to accelerate the development of open source software for the RISC-V architecture.

[3] https://riseproject.dev/
[4] https://lists.riscv.org/g/tech-prs

RISC-V

Bootloader Scopes and Goals

Drivers, Parsers, Loaders

Drivers, Parsers, Loaders

Drivers

- talk to hardware, e.g., graphics output
- abstract concepts, e.g., file systems
- may be provided by environment, such as UEFI DXE or Linux

Drivers, Parsers, Loaders

Drivers

- talk to hardware, e.g., graphics output
- abstract concepts, e.g., file systems
- may be provided by environment, such as UEFI DXE or Linux

Parsers

- understand data formats
- translate raw data to a usable form
- for configuration files and binaries

Drivers, Parsers, Loaders

Drivers

- talk to hardware, e.g., graphics output
- abstract concepts, e.g., file systems
- may be provided by environment, such as UEFI DXE or Linux

Parsers

- understand data formats
- translate raw data to a usable form
- for configuration files and binaries

Loaders

- potentially pick up configuration
- load application to memory
- place additional data in memory and/or registers

Drivers, Parsers, Loaders

Drivers

- talk to hardware, e.g., graphics output
- abstract concepts, e.g., file systems
- may be provided by environment, such as UEFI DXE or Linux

Parsers

- understand data formats
- translate raw data to a usable form
- for configuration files and binaries

Loaders

- potentially pick up configuration
- load application to memory
- place additional data in memory and/or registers

Eventually, tell the platform (“CPU”) to execute from a specific memory address.

Drivers, Parsers, Loaders

Drivers

- talk to hardware, e.g., graphics output
- abstract concepts, e.g., file systems
- may be provided by environment, such as UEFI DXE or Linux

Parsers

- understand data formats
- translate raw data to a usable form
- for configuration files and binaries

Loaders

- potentially pick up configuration
- load application to memory
- place additional data in memory and/or registers

Eventually, tell the platform (“CPU”) to execute from a specific memory address.

See also my talk on webboot:

- https://programm.froscon.org/2021/events/2703.html
- https://av.tib.eu/media/59579
- https://www.youtube.com/watch?v=nZgRV7gvZRw

RISC-V

Security Insights

Security Insights

Firmware is well known to be an attack surface.

[5] https://uefi.org/sites/default/files/resources/UEFI%20Firmware%20-%20Security%20Concerns%20and%20Best%20Practices.pdf

[6] https://www.binarly.io/advisories/BRLY-2021-007/index.html

[7] https://eclypsium.com/research/everyone-gets-a-rootkit/

Security Insights

Firmware is well known to be an attack surface.

Incidents increase:

- OEM compromise (e.g., MSI)
- vulnerabilities in firmware interfaces, such as
  - UEFI, e.g. Option ROMs[5], parsing variables[6]
  - ACPI WPBT (Windows Platform Binary Table)[7]

[5] https://uefi.org/sites/default/files/resources/UEFI%20Firmware%20-%20Security%20Concerns%20and%20Best%20Practices.pdf

[6] https://www.binarly.io/advisories/BRLY-2021-007/index.html

[7] https://eclypsium.com/research/everyone-gets-a-rootkit/

Supply Chain Security

Supply Chain Security

Software Bill of Materials (SBOM)

Supply Chain Security

Software Bill of Materials (SBOM)

Executive Order 14028 on Improving the Nation's Cybersecurity

- includes a lengthy definition of SBOM
  Buyers can use an SBOM to perform vulnerability or license analysis, both of which can be used to evaluate risk in a product.

Posted on May 12, 2021

Supply Chain Security

Software Bill of Materials (SBOM)

Executive Order 14028 on Improving the Nation's Cybersecurity

- includes a lengthy definition of SBOM
  Buyers can use an SBOM to perform vulnerability or license analysis, both of which can be used to evaluate risk in a product.

Posted on May 12, 2021

This is effectively a business requirement.

Bootloader Classification

Interactive vs non-interactive

Interactive vs non-interactive

Non-interactive

Simple devices need no interaction in the bootloader, e.g., wristbands. Settings and upgrade functionality may come from other devices, such as phones.

Interactive vs non-interactive

Non-interactive

Simple devices need no interaction in the bootloader, e.g., wristbands.
Settings and upgrade functionality may come from other devices, such as phones.

Interactive

Flexible devices are designed to run arbitrary operating systems and software.
They require a rich user interface.

[8] https://archive.fosdem.org/2022/schedule/event/fw_settings_and_menus/

Interactive vs non-interactive

Non-interactive

Simple devices need no interaction in the bootloader, e.g., wristbands. Settings and upgrade functionality may come from other devices, such as phones.

Interactive

Flexible devices are designed to run arbitrary operating systems and software.

They require a rich user interface.

- change settings
- set up a trust anchor
- enjoy colorful graphics

For more, see my talk on firmware settings and menus[8].

[8] https://archive.fosdem.org/2022/schedule/event/fw_settings_and_menus/

Applications

Applications

General purpose

General purpose bootloaders can be hard to customize.
We will look at possible solutions.

[9] https://danielmangum.com/posts/risc-v-bytes-exploring-custom-esp32-bootloader/

Applications

General purpose

General purpose bootloaders can be hard to customize.
We will look at possible solutions.

Special purpose

Special purpose bootloaders often need to be tailored[9] toward a single use case.

[9] https://danielmangum.com/posts/risc-v-bytes-exploring-custom-esp32-bootloader/

Projects and Stacks

Protocols, Interfaces and Features

Stages / Phases

Stages / Phases

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Stages / Phases

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

Stages / Phases

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

Depending on the platform design, multiple further stages are necessary.

Stages / Phases

Typical SoCs have early code in their mask ROM, sometimes also called BROM (boot ROM) or ZSBL (Zero Stage Boot Loader).

Boot ROMs may offer protocols for loading over UART or USB, which is great for development, e.g., Allwinner FEL, JH71x0 Xmodem.

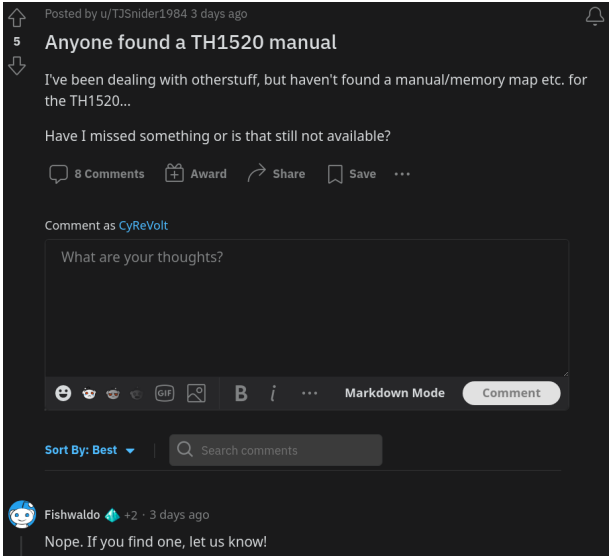
Depending on the platform design, multiple further stages are necessary.

Developers need documentation:

https://github.com/sipeed/LicheePi4A/issues/12

> *I want to know how `brom` load uboot image(emmc) to ram, because I'm try to upstream uboot. :) This is only vendor can know.*

SoC Manuals



- provide understanding of the platform
  - clocks
  - peripherals
  - registers
  - how to program them

SDK, HAL, SVD

SVD is an XML based, structured format to describe an SoC, including memory maps, peripherals, and registers, as well as elaborating text.

SDK, HAL, SVD

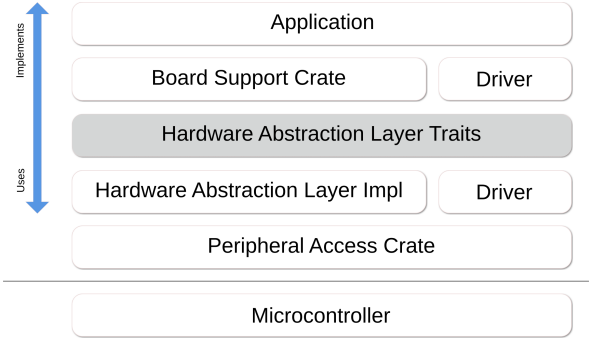
SVD is an XML based, structured format to describe an SoC, including memory maps, peripherals, and registers, as well as elaborating text.

Rust offers tooling to convert SVD into usable code.

SDK, HAL, SVD

SVD is an XML based, structured format to describe an SoC, including memory maps, peripherals, and registers, as well as elaborating text.

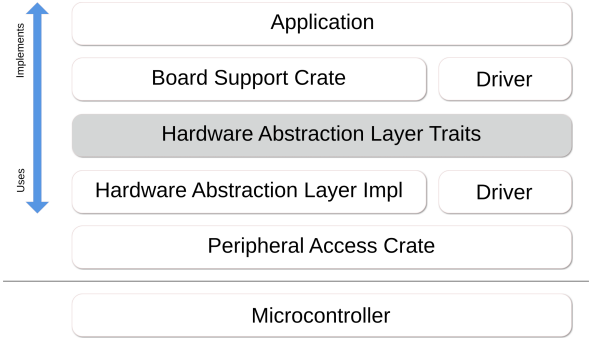
Rust offers tooling to convert SVD into usable code.



SDK, HAL, SVD

SVD is an XML based, structured format to describe an SoC, including memory maps, peripherals, and registers, as well as elaborating text.

Rust offers tooling to convert SVD into usable code.



https://docs.rs/d1-pac/latest/d1_pac

Tools for Development and Flashing

Tools for Development and Flashing

During development, or to set up a custom environment, tools are necessary to reprogram a device.

Tools for Development and Flashing

During development, or to set up a custom environment, tools are necessary to reprogram a device.

Boot ROM Tools

- sunxi-fel/xfel tools
- vf2-loader
- snagboot

Tools for Development and Flashing

During development, or to set up a custom environment, tools are necessary to reprogram a device.

Boot ROM Tools

- sunxi-fel/xfel tools
- vf2-loader
- snagboot

Provided by Bootloader

- U-Boot sf command
- Linux MTD (memory technology device) drivers

Silicon and DRAM init

Silicon and DRAM init

A bootloader for a rich OS relies on DRAM being initialized.

[10] coreboot on RISC-V 2017 https://www.youtube.com/watch?v=CDNIWuf1jAk

Silicon and DRAM init

A bootloader for a rich OS relies on DRAM being initialized.

- coreboot[10]
  - supported RISC-V from very early on
- oreboot
  - note: started with RISC-V right away
- UEFI SEC+PEI
  - e.g., Project Mu, Tianocore EDK2
- U-Boot TPL/SPL

[10] coreboot on RISC-V 2017 https://www.youtube.com/watch?v=CDNIWuf1jAk

Platforms, Ports and Flows

Tianocore EDK2 / UEFI

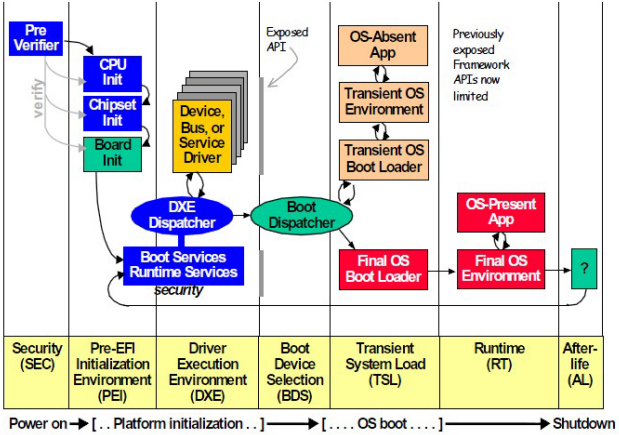


Figure 1-2. Framework Firmware Phases

DXE and BDS are effectively the UEFI bootloader.

They could also be replaced.

U-Boot

U-Boot offers a rich environment with an interactive shell and many boot options.

U-Boot

U-Boot offers a rich environment with an interactive shell and many boot options.

- supports multiple architectures
- more than 1000 boards, such as SBCs and routers
- can directly boot Linux and many other payloads

U-Boot

U-Boot

U-Boot offers a rich environment with an interactive shell and many boot options.

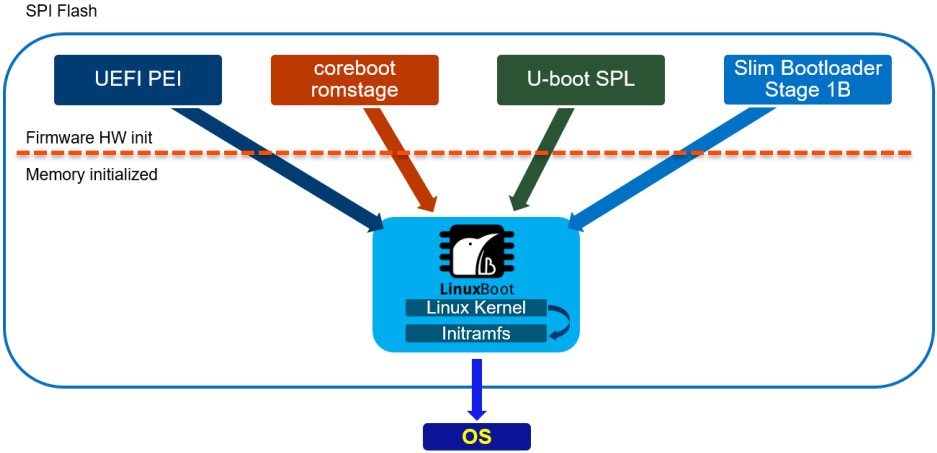
- supports multiple architectures
- more than 1000 boards, such as SBCs and routers
- can directly boot Linux and many other payloads

See also:

- State of the U-Boot, 2017 - Thomas Rini
  https://www.youtube.com/watch?v=dKBUSMa6oZI
- Implementing State-of-the-Art U-Boot Port, 2018 Edition - Marek Vasut
  https://www.youtube.com/watch?v=rJtlAi8rxgs

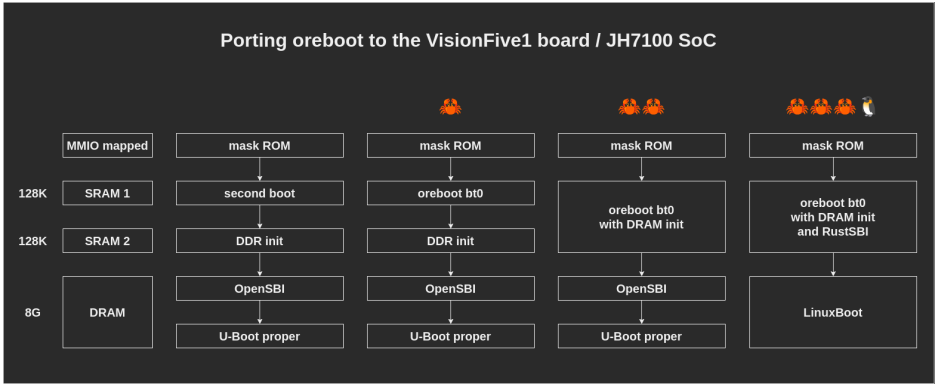
RISC-V

LinuxBoot



Linux is a well-known environment, so finding fitting engineers is easy.

LinuxBoot Integration with oreboot



LinuxBoot Environments

[11] https://u-root.org

LinuxBoot Environments

Any Linux userland can be used, depending on needs.

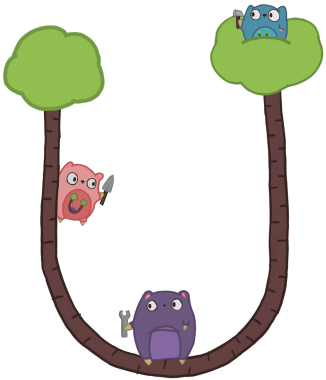
A common environment is u-root[11], the universal root filesystem.

[11] https://u-root.org

LinuxBoot Environments

Any Linux userland can be used, depending on needs.

A common environment is u-root[11], the universal root filesystem.



- written in Go
- uses the Linux drivers
- offers booting from
  - local storages
  - network
- can be embedded in flash
- easy to extend

[11] https://u-root.org

DEMO

OS Integration

Linux Distributions

Linux Distributions

OS Distributions such as openSUSE, Fedora and Ubuntu need bootloaders.

Linux Distributions

OS Distributions such as openSUSE, Fedora and Ubuntu need bootloaders.

If the bootloader in flash does not suffice, they may bring their own, e.g., GRUB.

[12] https://uapi-group.org/specifications/specs/boot_loader_specification/

Linux Distributions

OS Distributions such as openSUSE, Fedora and Ubuntu need bootloaders.

If the bootloader in flash does not suffice, they may bring their own, e.g., GRUB.

For them, the best setup is standardized, such as Boot Loader Spec[12].

[12] https://uapi-group.org/specifications/specs/boot_loader_specification/

Success Stories

Case Studies

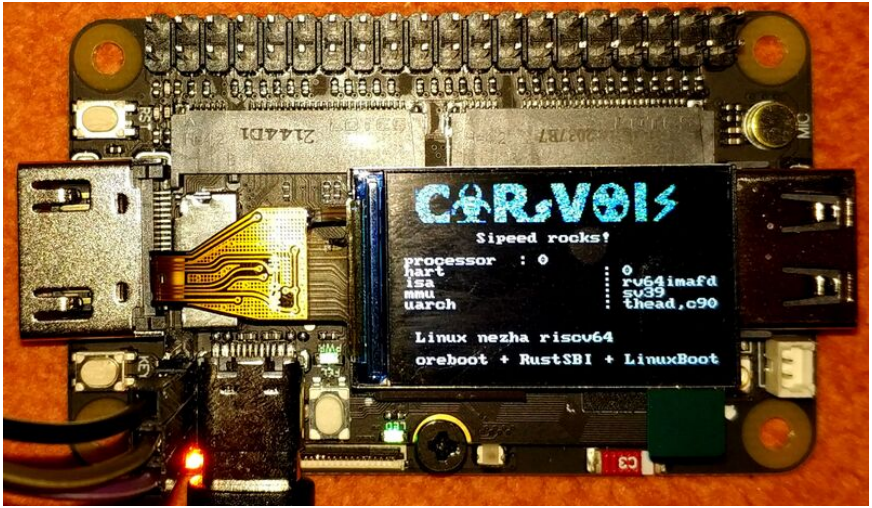
Allwinner D1 with oreboot and LinuxBoot

Allwinner D1 with oreboot and LinuxBoot

The system boots within seconds. We created environments that allow for using a D1 as a USB gadget that can be used as an additional CPU for a laptop.

Allwinner D1 with oreboot and LinuxBoot

The system boots within seconds. We created environments that allow for using a D1 as a USB gadget that can be used as an additional CPU for a laptop.



FreeBSD

kboot: Booting FreeBSD with LinuxBoot[13]

FreeBSD's kboot is a Linux binary that loads FreeBSD's kernel, modules, tuneables and other metadata via the kexec(2) API

[13] https://www.bsdcan.org/events/bsdcan_2023/schedule/session/138-kboot-booting-freebsd-with-linuxboot/

Other Operating Systems

Saving Costs

Sharing Code

Sharing Code

Device Trees

Device Trees describe specific boards and are shared between projects.

- Linux
- U-Boot
- FreeBSD

Note: U-Boot also shares parts of its driver model with Linux.

[14] https://crates.io/

Sharing Code

Device Trees

Device Trees describe specific boards and are shared between projects.

- Linux
- U-Boot
- FreeBSD

Note: U-Boot also shares parts of its driver model with Linux.

Rust

Rust code can be shared using *crates*[14], which can speed up driver development.

[14] https://crates.io/

Drivers

LinuxBoot requires only writing drivers once.

Less effort means lower costs and faster time to market.

Thanks! :)

Follow Me

https://github.com/orangecms
https://twitter.com/orangecms
https://twitch.tv/cyrevolt
https://youtube.com/@cyrevolt

Daniel Maslowski

https://github.com/oreboot/oreboot

https://metaspora.org/bootloaders-in-limbo.pdf