What comes before Linux …

or BSD, Windows, macOS, Haiku, Oberon, Plan 9, …?

Daniel Maslowski aka CyReVolt

Hello, I am Daniel aka CyReVolt :-)

Work and education

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

Open Source contributions

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

Hello, I am Daniel aka CyReVolt :-)

Work and education

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

Open Source contributions

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

I created Fiedka the firmware editor (https://fiedka.app) and started the Platform System Interface project: https://github.com/platform-system-interface/

Agenda

- Bootloaders and Firmware
- Classification, Scopes and Goals
- Projects
- Platform Ownership

Bootloaders and Firmware

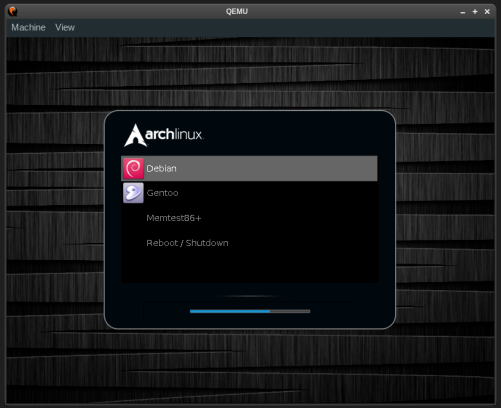
What is a Bootloader?

What is a Bootloader?

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

What is a Bootloader?

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
- well-known examples
  - GRUB
  - sd-boot
  - U-Boot (proper)

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

Between Firmware and OS

Between Firmware and OS

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
- …

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

Security Insights

Security Insights

Firmware is well known to be an attack surface.

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[a], parsing variables[b]
  - ACPI WPBT (Windows Platform Binary Table)[c]
  - LogoFAIL[d], PixieFail[e], …

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

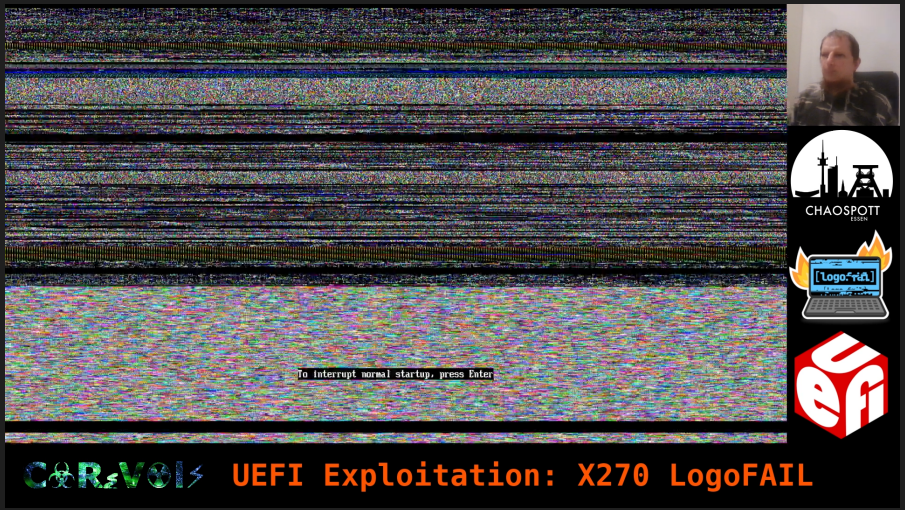
[b] https://www.binarly.io/advisories/BRLY-2021-007/index.html

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

[d] https://binarly.io/posts/finding_logofail_the_dangers_of_image_parsing_during_system_boot/

[e] https://blog.quarkslab.com/pixiefail-nine-vulnerabilities-in-tianocores-edk-ii-ipv6-network-stack.html

It really works



https://www.youtube.com/watch?v=X2X18h5Hnfk

Classification, Scopes and Goals

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 custom operating systems and software.
Security note: Runtime configurability leaves space for *vulnerabilities*.
Offer a rich user interface to

[1] 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 custom operating systems and software.
Security note: Runtime configurability leaves space for *vulnerabilities*.
Offer a rich user interface to

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

For more, see my talk on firmware settings and menus[1].

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

Applications

Applications

General purpose

General purpose bootloaders can be hard to customize.
Design them to be clear to end users for distribution and integration.

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

Applications

General purpose

General purpose bootloaders can be hard to customize.
Design them to be clear to end users for distribution and integration.

Special purpose

Special purpose bootloaders often need to be tailored[2] toward a single use case. With a clear execution flow, it is easier to understand their behavior.

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

Where it really starts

Where it really starts

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

Where it really starts

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 serial or USB ports, which is great for development, e.g., Allwinner FEL, JH71x0 XMODEM.

Where it really starts

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 serial or USB ports, which is great for development, e.g., Allwinner FEL, JH71x0 XMODEM.

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

Where it really starts

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 serial or USB ports, which is great for development, e.g., Allwinner FEL, JH71x0 XMODEM.

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

General flow: firmware -> bootloader -> OS

Where it really starts

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 serial or USB ports, which is great for development, e.g., Allwinner FEL, JH71x0 XMODEM.

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

General flow: firmware -> bootloader -> OS

Customizing code from the beginning requires a concept of *ownership*.

Early init: Silicon and DRAM

Early init: Silicon and DRAM

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

Early init: Silicon and DRAM

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

- Project Mu, Tianocore EDK2 (UEFI)
  - SEC+PEI
- coreboot
  - CAR and ROM stages
- oreboot
  - bt0 stage
- U-Boot
  - SPL, rarely TPL

Early init: Silicon and DRAM

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

- Project Mu, Tianocore EDK2 (UEFI)
  - SEC+PEI
- coreboot
  - CAR and ROM stages
- oreboot
  - bt0 stage
- U-Boot
  - SPL, rarely TPL

Note: Documentation on DRAM controllers is very sparse.
Chip vendors rarely describe how initial parts of their platforms work.

Tools for Development and Flashing

Tools for Development and Flashing

During development, or for customization, tools are necessary to reprogram a device and check/change its *OTP* (one-time programmable) configuration.

Tools for Development and Flashing

During development, or for customization, tools are necessary to reprogram a device and check/change its *OTP* (one-time programmable) configuration.

Boot ROM / Loader Tools

- Allwinner: `sunxi-fel`, `xfel`, `aw-fel-cli`
- Rockchip: `rkflashtool`, `rkdeveloptool`
- Amlogic: `pyamlboot`, `aml_boot`
- NXP: `uuu`, `imx_usb_loader`
- StarFive JH7110: `vf2-loader`
- Android: `fastboot` (details vary per vendor)
- `snagboot` (multitool)

Tools for Development and Flashing

During development, or for customization, tools are necessary to reprogram a device and check/change its *OTP* (one-time programmable) configuration.

Boot ROM / Loader Tools

- Allwinner: `sunxi-fel`, `xfel`, `aw-fel-cli`
- Rockchip: `rkflashtool`, `rkdeveloptool`
- Amlogic: `pyamlboot`, `aml_boot`
- NXP: `uuu`, `imx_usb_loader`
- StarFive JH7110: `vf2-loader`
- Android: `fastboot` (details vary per vendor)
- `snagboot` (multitool)

Provided by Bootloader

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

Projects

Tianocore EDK2 / UEFI

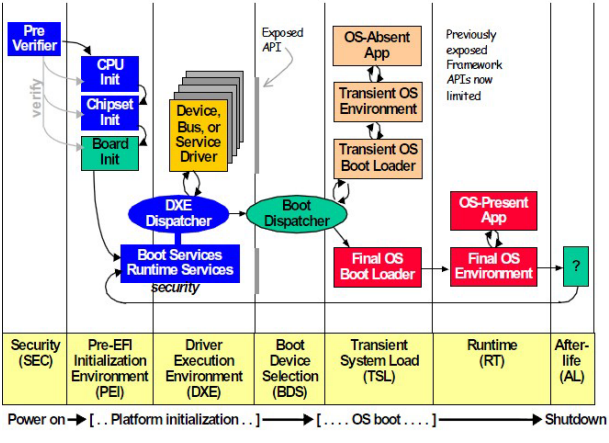


Figure 1-2. Framework Firmware Phases

DXE and BDS are effectively the UEFI bootloader, can be replaced with Linux.

Tianocore EDK2 / UEFI

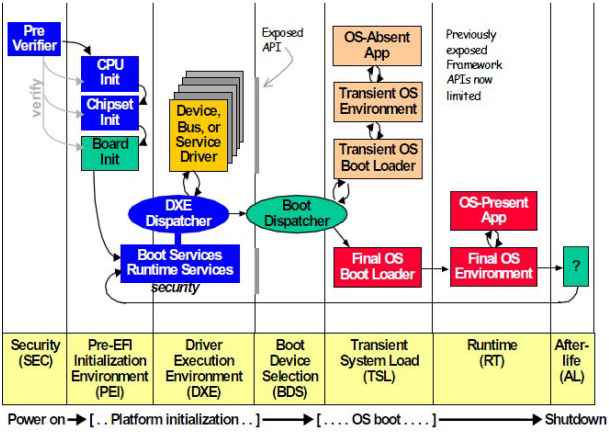


Figure 1-2. Framework Firmware Phases

DXE and BDS are effectively the UEFI bootloader, can be replaced with Linux.

Real devices usually come with OEM controlled environments.

Tianocore EDK2 / UEFI

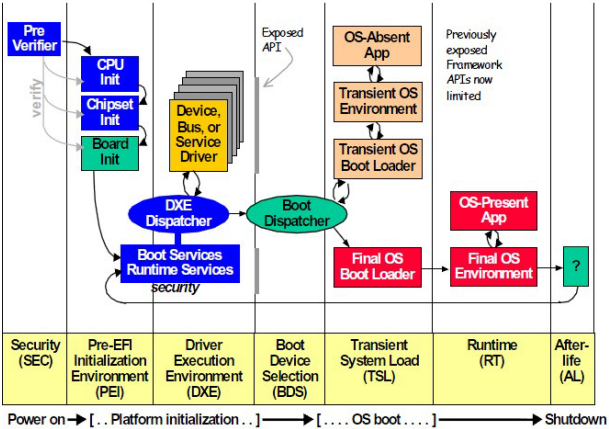


Figure 1-2. Framework Firmware Phases

DXE and BDS are effectively the UEFI bootloader, can be replaced with Linux.

Real devices usually come with OEM controlled environments.

See also “BIOS modding”.

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

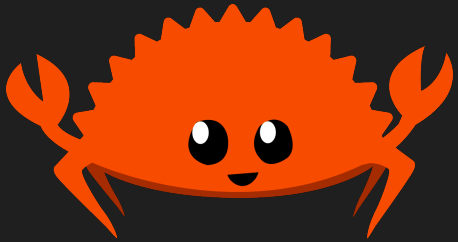
- supports multiple architectures
- more than 1000 boards, such as SBCs and routers
- can directly boot Linux and many other payloads

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

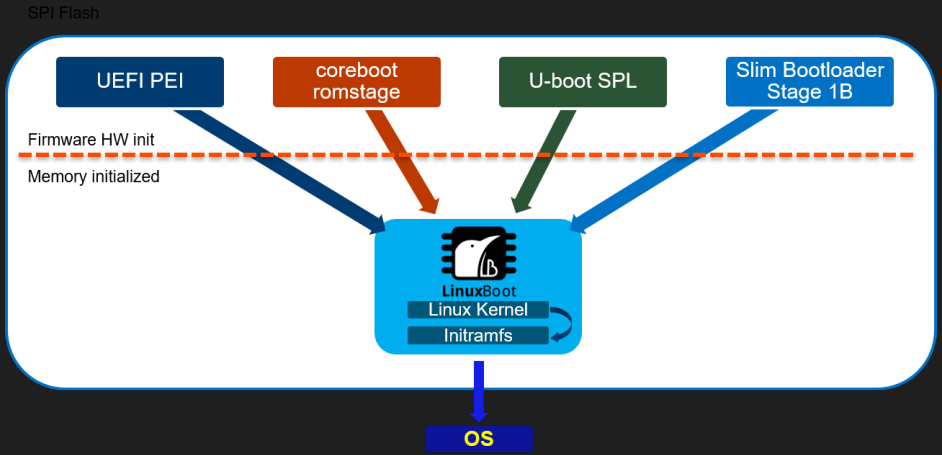
oreboot

oreboot is firmware written in Rust.



https://github.com/oreboot

LinuxBoot



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

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[3]

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

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

Sooooo many Operating Systems

Sooooo many Operating Systems

How many do you know?

Sooooo many Operating Systems

How many do you know?

- Windows
- Unix (Multics, … AIX, …)
- SunOS, Solaris, Illumos …
- {Free,Open,Net,DragonFly}BSD
- macOS (Darwin, MACH + FreeBSD)
- MINIX
- xv6
- Linux (many distros)
- Amoeba (where Python came from)
- Oberon (Niklaus Wirth et al)
- Plan 9 from Bell Labs, Inferno
- beOS, Haiku
- FreeRTOS
- Zephyr
- LiteOS
- … too many to name here :)

Platform Ownership

Let's talk!

Full ownership?

- locked bootloaders (phones), can sometimes be unlocked
- signed firmware (e.g., Intel BootGuard), sometimes misconfigured :-)
- projects (OpenWrt, OpenIPC, …) often replace vendor software partially
- control from start allows for more customization, easier development

Sustainability

What do we do with hardware solely made for a cloud based service?

- services are being turned off over time
- Google Stadia: offered Bluetooth upgrade for controller
- Magenta Smart Speaker: now only a Bluetooth speaker
- cheap TV boxes and tablets with unmaintained Android
- single board computers that rely on community (us!)

Thanks! :)

Follow Me

Daniel Maslowski

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

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

https://www.youtube.com/watch?v=-Ub8rCMrso0

https://metaspora.org/before-linux.pdf