ASAHI LINUX - PORTING LINUX TO APPLE SILICON

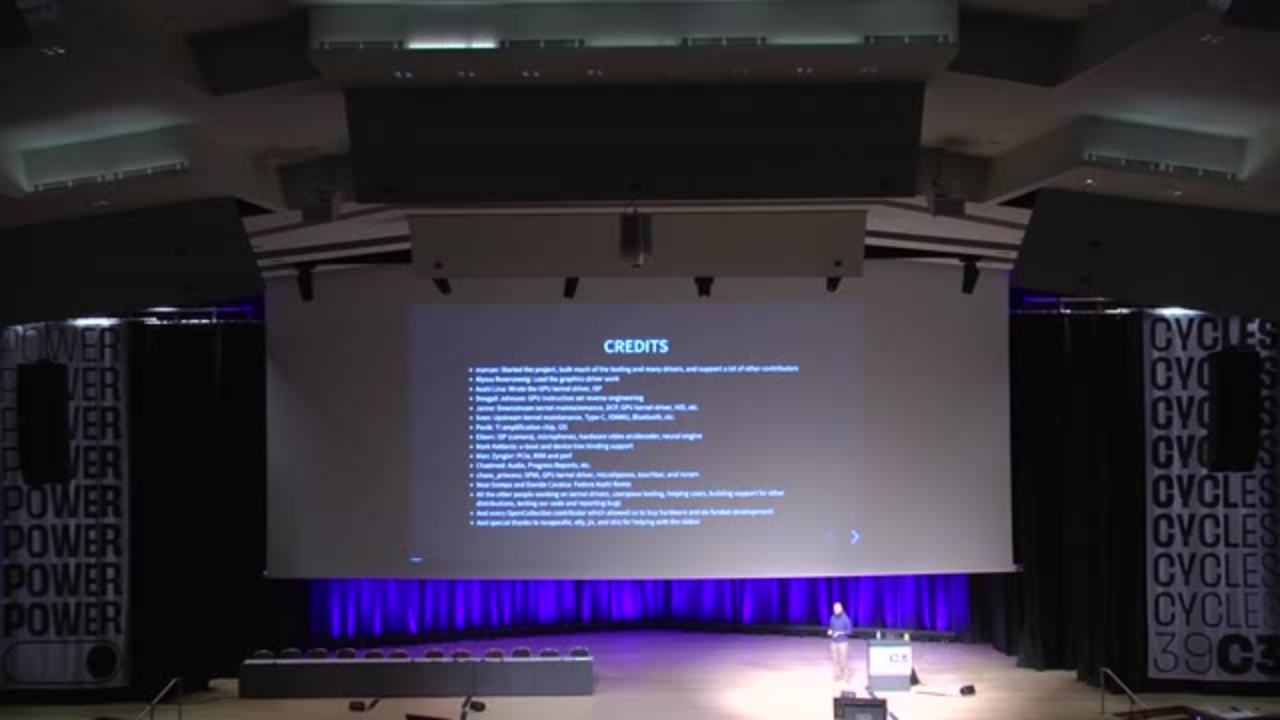
sven

CREDITS

- marcan: Started the project, built much of the tooling and many drivers, and support a lot of other contributors
- Alyssa Rosenzweig: Lead the graphics driver work
- Asahi Lina: Wrote the GPU kernel driver, ISP
- Dougall Johnson: GPU instruction set reverse engineering
- Janne: Downstream kernel maintainenance, DCP, GPU kernel driver, HID, etc.
- Sven: Upstream kernel maintanance, Type-C, IOMMU, Bluetooth, etc.
- Povik: TI amplification chip, I2S
- Eileen: ISP (camera), microphones, hardware video en/decoder, neural engine
- Mark Kettenis: u-boot and device tree binding support
- Marc Zyngier: PCIe, KVM and perf
- Chadmed: Audio, Progress Reports, etc.
- chaos_princess: SPMI, GPU kernel driver, microhpones, touchbar, and nvram
- Neal Gompa and Davide Cavalca: Fedora Asahi Remix
- All the other people working on kernel drivers, userspace tooling, helping users, building support for other distributions, testing our code and reporting bugs
- And every OpenCollective contributor which allowed us to buy hardware and do funded development!
- And special thanks to noopwafel, elly, jix, and shiz for helping with the slides!

CREDITS

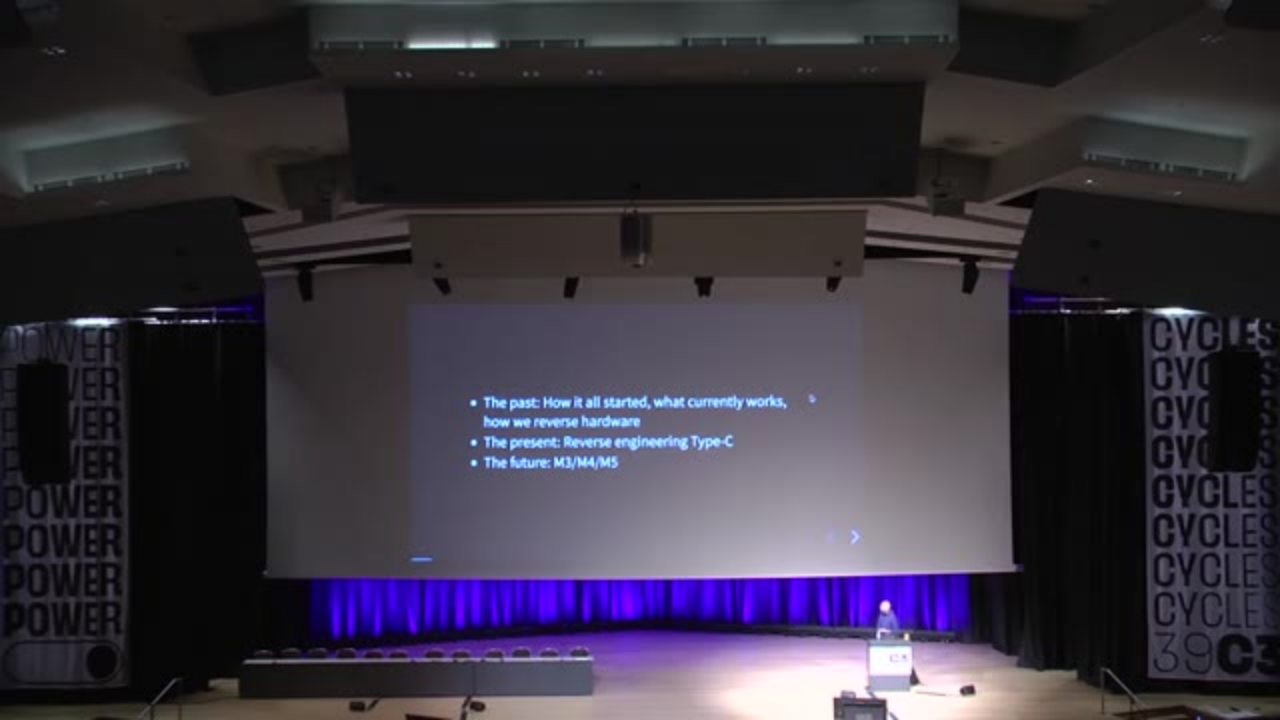
- marcan: Started the project, built much of the tooling and many drivers, and support a lot of other contributors
- Alyssa Rosenzweig: Lead the graphics driver work
- Asahi Lina: Wrote the GPU kernel driver, ISP
- Dougall Johnson: GPU instruction set reverse engineering
- Janne: Downstream kernel maintainenance, DCP, GPU kernel driver, HID, etc.
- Sven: Upstream kernel maintanance, Type-C, IOMMU, Bluetooth, etc.
- Povik: TI amplification chip, I2S
- Eileen: ISP (camera), microphones, hardware video en/decoder, neural engine
- Mark Kettenis: u-boot and device tree binding support
- Marc Zyngier: PCIe, KVM and perf
- Chadmed: Audio, Progress Reports, etc.
- chaos_princess: SPMI, GPU kernel driver, microhpones, touchbar, and nvram
- Neal Gompa and Davide Cavalca: Fedora Asahi Remix
- All the other people working on kernel drivers, userspace tooling, helping users, building support for other distributions, testing our code and reporting bugs
- And every OpenCollective contributor which allowed us to buy hardware and do funded development!
- And special thanks to noopwafel, elly, jix, and shiz for helping with the slides!


CREDITS
marcan: Started the project, built much of the tooling and many drivers, and support a lot of other contributors
Alyssa Rosenzweig: Lead the graphics driver work
Asahi Lina: Wrote the GPU kernel driver, DCP
Dougall Johnson: GPU instruction set reverse engineering
Jannau: Downstream kernel maintainenance, DCP, GPU kernel driver, HID, etc.
Sven: Upstream kernel maintainance, Type-C, USB3, Bluetooth, etc.
Povik: T2 amplification chip, I2S
Eileen: DP (camera), microphones, hardware video accelerator, neural engine
Mark Kettenis: u-boot and device tree binding support
Marc Zyngier: PCIe, RAM and perf
Chadmed: Audio, Progress Reports, etc.
chaos_princess: GPIO, GPU kernel driver, microphones, bootflow, and more!
Neal Gompa and Davide Cavalca: Fedora Asahi Remix
All the other people working on kernel drivers, userspace tooling, helping users, building support for other distributions, testing our code and reporting bugs
And every OpenCollective contributor which allowed us to buy hardware and do funded development
And special thanks to nosuspend, elly, jx, and stru for helping with the video!
POWER
CYCLES
39C3

CREDITS

- marcan: Started the project, built much of the tooling and many drivers, and support a lot of other contributors
- Alyssa Rosenzweig: Lead the graphics driver work
- Asahi Lina: Wrote the GPU kernel driver, ISP
- Dougall Johnson: GPU instruction set reverse engineering
- Janne: Downstream kernel maintainenance, DCP, GPU kernel driver, HID, etc.
- Sven: Upstream kernel maintanance, Type-C, IOMMU, Bluetooth, etc.
- Povik: TI amplification chip, I2S
- Eileen: ISP (camera), microphones, hardware video en/decoder, neural engine
- Mark Kettenis: u-boot and device tree binding support
- Marc Zyngier: PCIe, KVM and perf
- Chadmed: Audio, Progress Reports, etc.
- chaos_princess: SPMI, GPU kernel driver, microhpones, touchbar, and nvram
- Neal Gompa and Davide Cavalca: Fedora Asahi Remix
- All the other people working on kernel drivers, userspace tooling, helping users, building support for other distributions, testing our code and reporting bugs
- And every OpenCollective contributor which allowed us to buy hardware and do funded development!
- And special thanks to noopwafel, eily, jix, and shiz for helping with the slides!




• The past: How it all started, what currently works, how we reverse hardware
• The present: Reverse engineering Type-C
• The future: M3/M4/M5
POWER
CYCLES
39C3

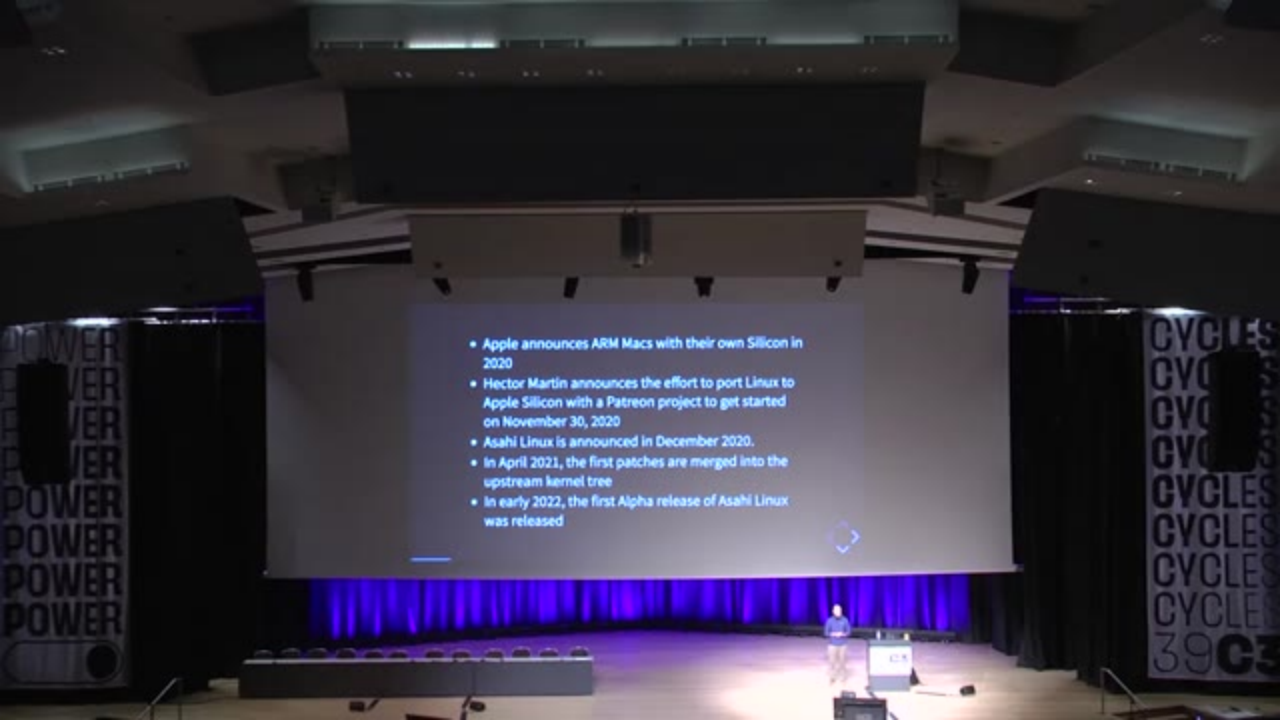
- Apple announces ARM Macs with their own Silicon in 2020
- Hector Martin announces the effort to port Linux to Apple Silicon with a Patreon project to get started on November 30, 2020
- Asahi Linux is announced in December 2020.
- In April 2021, the first patches are merged into the upstream kernel tree
- In early 2022, the first Alpha release of Asahi Linux was released

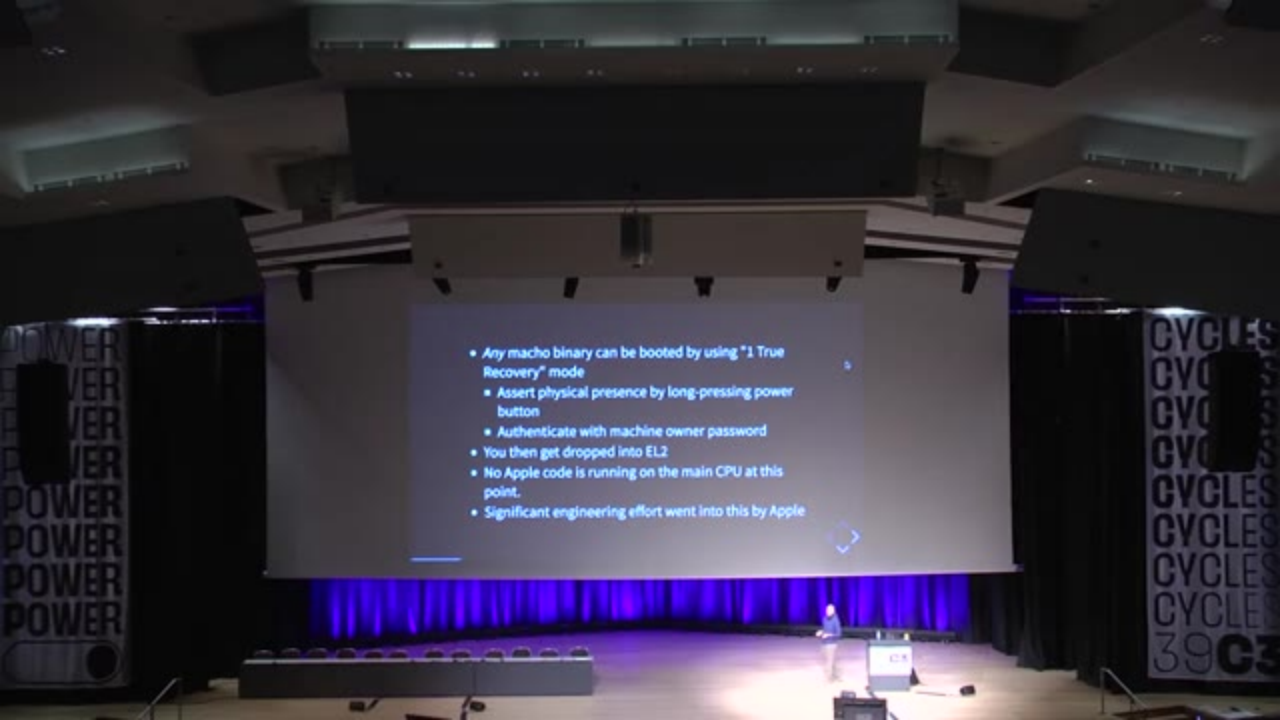
39C3
POWER CYCLES



- Apple announces ARM Macs with their own Silicon in 2020
- Hector Martin announces the effort to port Linux to Apple Silicon with a Patreon project to get started on November 30, 2020
- Asahi Linux is announced in December 2020.
- In April 2021, the first patches are merged into the upstream kernel tree
- In early 2022, the first Alpha release of Asahi Linux was released




Apple announces ARM Macs with their own Silicon in 2020
Hector Martin announces the effort to port Linux to Apple Silicon with a Patreon project to get started on November 30, 2020
Asahi Linux is announced in December 2020.
In April 2021, the first patches are merged into the upstream kernel tree
In early 2022, the first Alpha release of Asahi Linux was released
POWER
CYCLES
39C3


• Any macho binary can be booted by using "1 True Recovery" mode
◦ Assert physical presence by long-pressing power button
◦ Authenticate with machine owner password
• You then get dropped into EL2
• No Apple code is running on the main CPU at this point.
• Significant engineering effort went into this by Apple
POWER
CYCLES
39C3

- No Apple code is running on the main CPU at this point.
- Significant engineering effort went into this by Apple

No exploits or similar are required

This is all intentionally allowed by Apple



No exploits or similar are required

This is all intentionally allowed by Apple



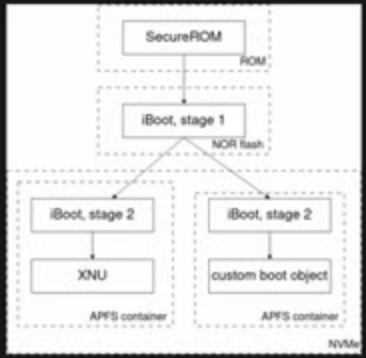


Xeno Kovah
@XenoKovah

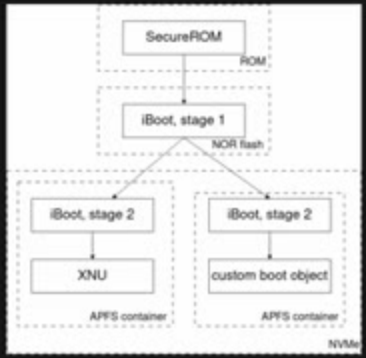
I purposely designed a mechanism so that M1 Macs would retain the capability to boot completely arbitrary code instead of XNU if users wanted. But you have to 1) reboot to recoveryOS with a physical power button press and 2) put in your SEP-backed credentials.



POWER CYCLES







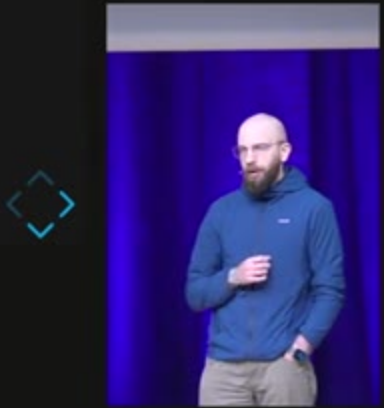
39C3

POWER CYCLES




macOS 12
Macintosh HD
Asahi Linux
Options
Continue

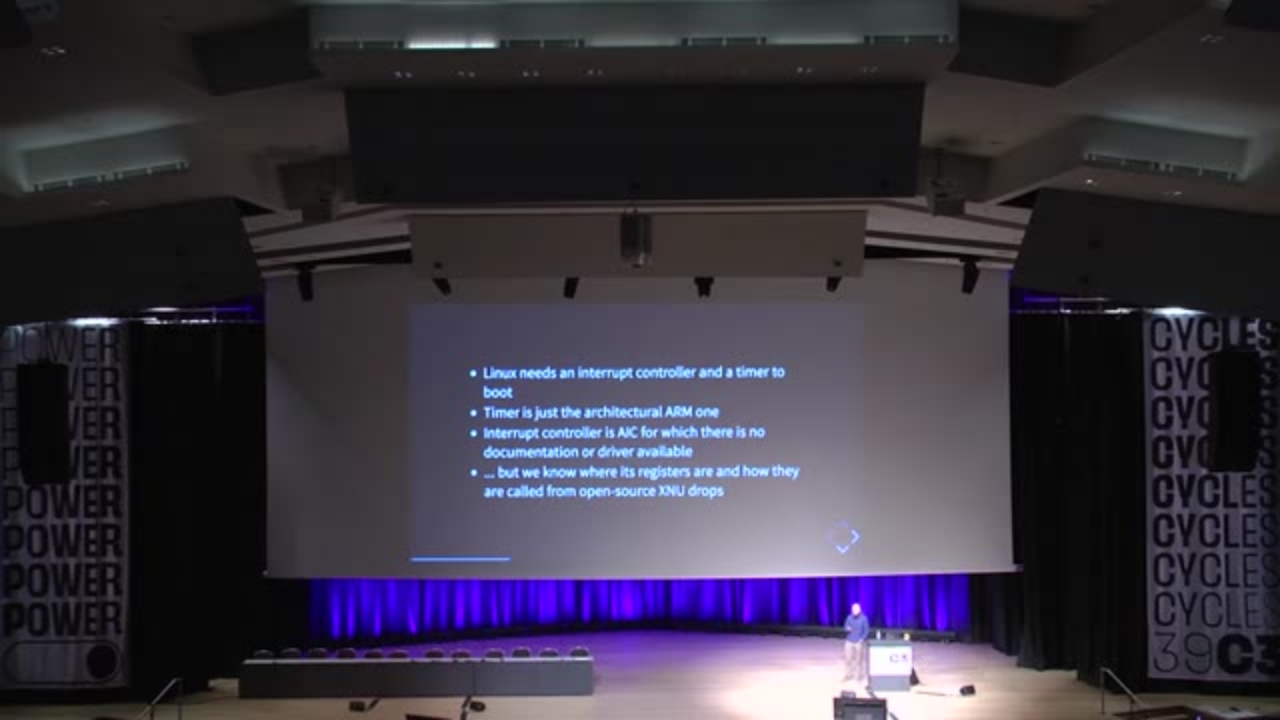
- Linux needs an interrupt controller and a timer to boot
- Timer is just the architectural ARM one
- Interrupt controller is AIC for which there is no documentation or driver available
- ... but we know where its registers are and how they are called from open-source XNU drops



39C3
POWER CYCLES

- Linux needs an interrupt controller and a timer to boot
- Timer is just the architectural ARM one
- Interrupt controller is AIC for which there is no documentation or driver available
- ... but we know where its registers are and how they are called from open-source XNU drops




Linux needs an interrupt controller and a timer to boot
Timer is just the architectural ARM one
Interrupt controller is AIC for which there is no documentation or driver available
... but we know where its registers are and how they are called from open-source XNU drops
POWER
CYCLES
39C3

- Linux needs an interrupt controller and a timer to boot
- Timer is just the architectural ARM one
- Interrupt controller is AIC for which there is no documentation or driver available
- ... but we know where its registers are and how they are called from open-source XNU drops



- Linux needs an interrupt controller and a timer to boot
- Timer is just the architectural ARM one
- Interrupt controller is AIC for which there is no documentation or driver available
- ... but we know where its registers are and how they are called from open-source XNU drops



M1N1

- Back when we reverse engineered the Wii we built mini which had a small RPC proxy
- Apple Silicon UART can be muxed out over USB
- Python shell on a separate machine which allows to poke hardware registers



39C3 POWER CYCLES

- python reboot.py to reboot the target
- python shell.py to drop to a python shell which allows MMIO access
  - read32(), write32(), ...
  - Allocate memory on the device under test
  - Monitor MMIO ranges for changes
- Allows prototyping drivers in python
- Very quick development cycle



```
def i2c_read_reg(addr, reg, reg_size):
    p.set32(base + REG_CTL, CTL_MTR | CTL_MRR)
    p.write32(base + REG_SMSTA, 0xffffffff)

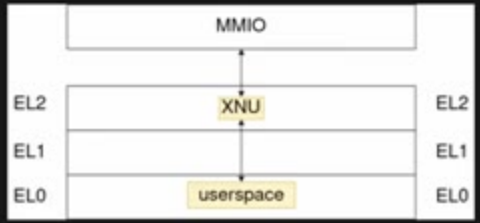
    p.write32(base + REG_MTXFIFO, MTXFIFO_START | (addr << 1))
    p.write32(base + REG_MTXFIFO, MTXFIFO_STOP | reg)

    while not (p.read32(base + REG_SMSTA) & SMSTA_XEN):
        pass

    p.write32(base + REG_MTXFIFO, MTXFIFO_START | (addr << 1) | 1)
    p.write32(base + REG_MTXFIFO, MTXFIFO_READ | MTXFIFO_STOP | reg_size + 1)

    res = []
    while len(res) < reg_size+1:
```




MMIO
EL2
XNU
EL2
EL1
EL1
EL0
userspace
EL0




39C3

```
class R_USB2PHY_MISCTUNE(Register32):
    APBCLK_GATE_OFF = 29
    REFCLK_GATE_OFF = 30

class Usb2PhyRegs(RegMap):
    USB2PHY_MISCTUNE = 0x1C, R_USB2PHY_MISCTUNE

class PhyTracer(ADTDevTracer):
    REGMAPS = [
        Usb2PhyRegs,
    # ...
    ]

PhyTracer(hv, "/arm-io/atc-phy1", verbose=2).start()
```

39C3

POWER CYCLES



39C3

POWER CYCLES

```
 1 # [cpu0] Pass: msr OSLAR_EL1, x31 = 0 (OK) (OSLAR_EL1)
 2 # [cpu0] Shadow: mrs x12, MDSCR_EL1 = 0
 3 # [cpu0] Skip: msr ACC_CFG_EL1, x13 = d
 4 # [cpu0] PMGR R 28e0801d8+0:32 = 0xf0000ff -> 0xf0000ff
 5 # [cpu0] PMGR R 28e0801f0+0:32 = 0xf0000ff -> 0xf0000ff
 6 ...
 7 # [cpu0] CPUSTART W 28e0d4000+c:32 = 0x0
 8 # [cpu0] CPUSTART W 28e0d4000+10:32 = 0x4
 9 # [cpu0] Starting guest secondary 0:2:2
10 ...
11 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd3c = 0xfc00000
12 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: W.4   0xb0228cd3c = 0xfc00fc0
13 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd40 = 0x3c20000c
14 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: W.4   0xb0228cd40 = 0x3c14060c
15 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd40 = 0x3c14060c
```



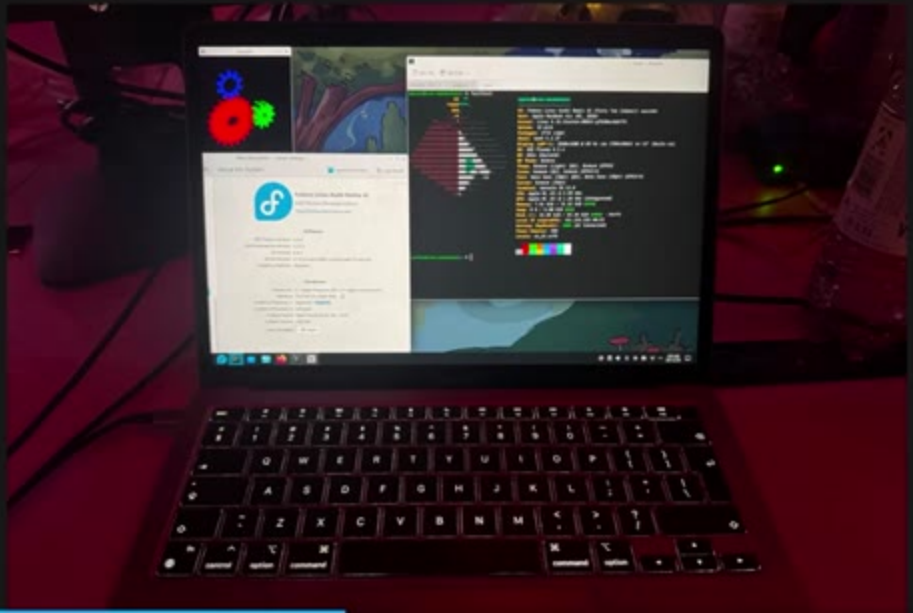
39C3

POWER CYCLES

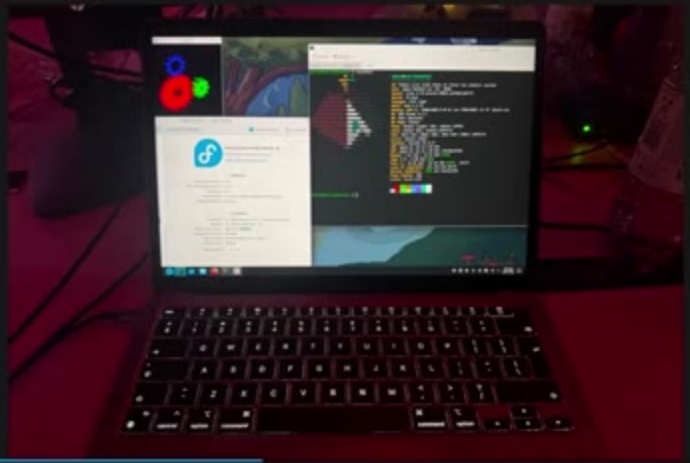
```
1  # [cpu0] Pass: msr OSLAR_EL1, x31 = 0 (OK) (OSLAR_EL1)
2  # [cpu0] Shadow: mrs x12, MDSCR_EL1 = 0
3  # [cpu0] Skip: msr ACC_CFG_EL1, x13 = d
4  # [cpu0] PMGR R 28e0801d8+0:32 = 0xf0000ff -> 0xf0000ff
5  # [cpu0] PMGR R 28e0801f0+0:32 = 0xf0000ff -> 0xf0000ff
6  ...
7  # [cpu0] CPUSTART W 28e0d4000+c:32 = 0x0
8  # [cpu0] CPUSTART W 28e0d4000+10:32 = 0x4
9  # [cpu0] Starting guest secondary 0:2:2
10 ...
11 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd3c = 0xfc00000
12 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: W.4   0xb0228cd3c = 0xfc00fc0
13 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd40 = 0x3c20000c
14 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: W.4   0xb0228cd40 = 0x3c14060c
15 # [Dwc3Tracer0/arm-io/usb-drd1] MMIO: R.4   0xb0228cd40 = 0x3c14060c
```



```
curl https://alx.sh | sh
```



```
curl https://alx.sh | sh
```



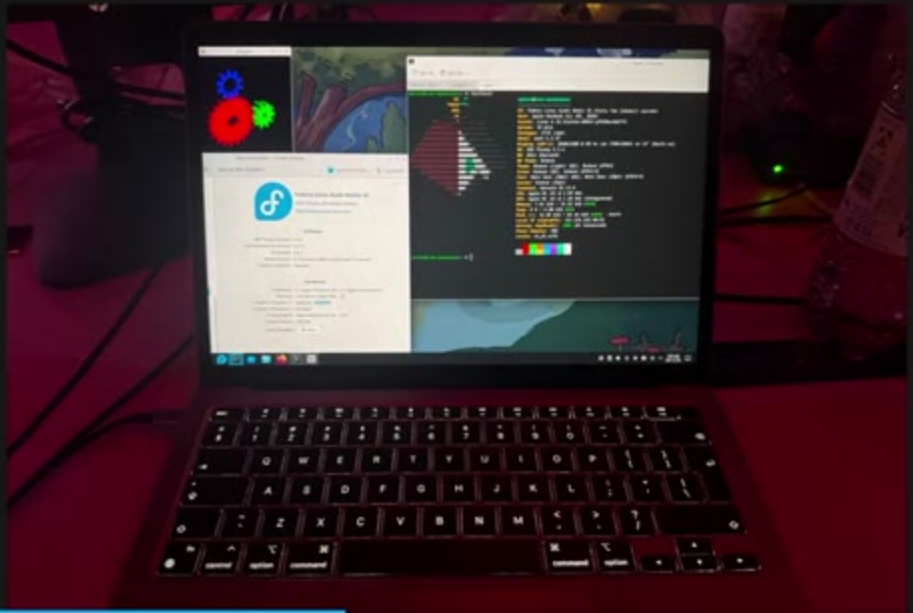
39C3

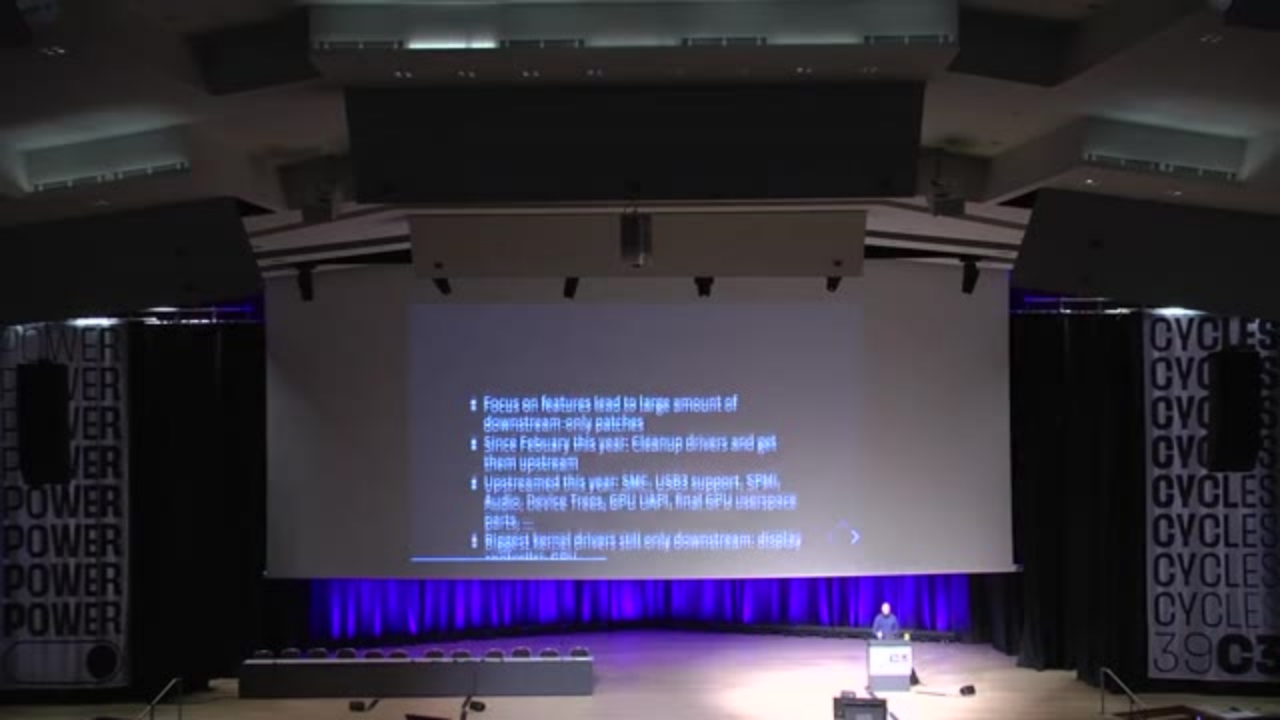
POWER CYCLES




curl https://ala.sh | sh
POWER
POWER
POWER
POWER
POWER
CYCLES
CYCLES
CYCLES
CYCLES
CYCLES
CYCLES
39C3

```
curl https://alx.sh | sh
```




Focus on features lead to large amount of downstream-only patches
Since February this year: Cleanup drivers and get them upstream
Upstreamed this year: SMC, USB3 support, SPMI, Audio, Device Trees, GPU UAPI, final GPU userspace parts, ...
Biggest kernel drivers still only downstream: display
POWER
CYCLES
39C3



- Focus on features lead to large amount of downstream-only patches
- Since Febuary this year: Cleanup drivers and get them upstream
- Upstreamed this year: SMC, USB3 support, SPMI, Audio, Device Trees, GPU UAPI, final GPU userspace parts, ...
- Biggest kernel drivers still only downstream: display controller, GPU





- 1996: USB1.0: 1.5 Mbit/s low-speed



- 1996: USB1.0: 1.5 Mbit/s low-speed
- 1998: USB1.1: 12 Mbit/s / full-speed

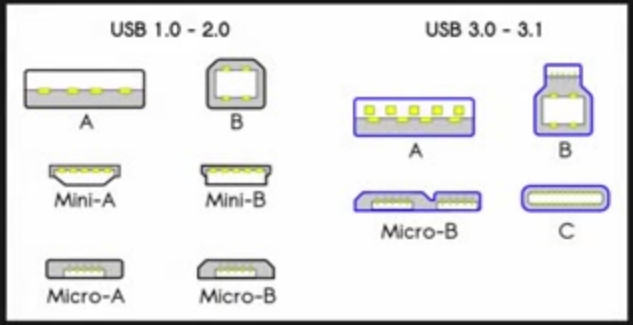


- 1996: USB1.0: 1.5 Mbit/s low-speed
- 1998: USB1.1: 12 Mbit/s / full-speed
- 2000: USB2.0: 480 Mbit/s / high-speed
- 2008: USB3.0: 5 Gbit/s / super-speed
- 2013: USB3.1: 10 Gbit/s / super-speed+



- 1996: USB1.0: 1.5 Mbit/s low-speed / D+/-
- 1998: USB1.1: 12 Mbit/s / full-speed
- 2000: USB2.0: 480 Mbit/s / high-speed
- 2008: USB3.0: 5 Gbit/s / USB 5Gbps
- 2013: USB3.1: 10 Gbit/s / USB 10Gbps
- 2017: USB3.2: 20 Gbit/s / USB 20Gbps
- 2019: USB4: 40 Gbit/s / USB 40Gbps



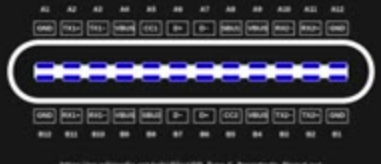


https://commons.wikimedia.org/wiki/File:USB_2.0_and_3.0_connectors.png

39C3
POWER CYCLES

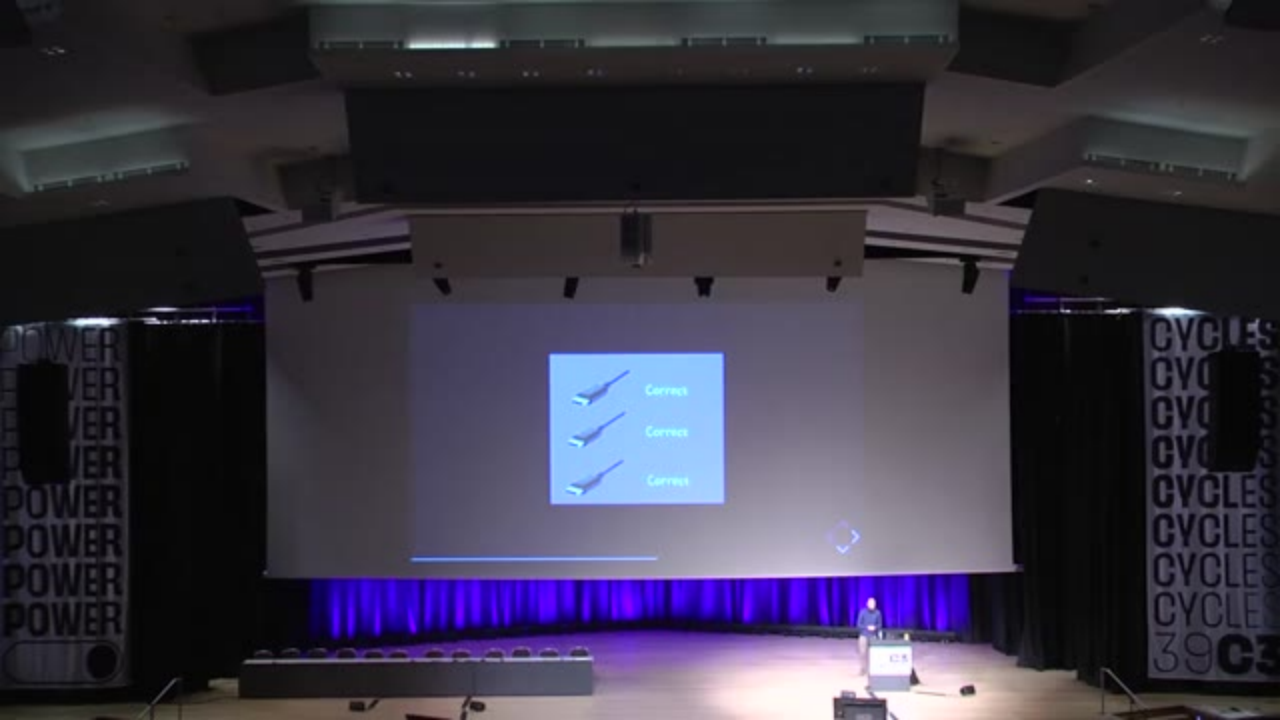


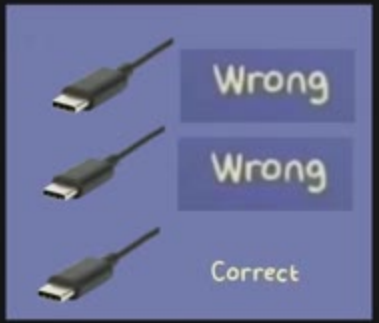
3903 POWER CYCLES



https://en.wikipedia.org/wiki/File:USB_Type-C_Receptacle_Pinout.svg




Correct
Correct
Correct
POWER
CYCLES
39C3



39C3
POWER CYCLES



39C3 POWER CYCLES

- USB1/2
- USB3
- Thunderbolt / USB4
- DisplayPort





- We know that iPhones used Synopsys Designware DWC2



- We know that iPhones used Synopsys Designware DWC2
- Poke MMIO with m1n1: It's Synopsys Designware DWC3 now
- Official datasheet not available, but registers, etc. published by e.g. Intel or Rockchip




39

- We know that iPhones used Synopsys Designware DWC2
- Poke MMIO with m1n1: It's Synopsys Designware DWC3 now
- Official datasheet not available, but registers, etc. published by e.g. Intel or Rockchip
- Linux has a driver which even used to be BSD/GPL dual-licensed years ago



39C3 POWER CYCLES

- DWC3 and CDC ACM (serial) implemented in m1n1
- ... but it only seems to work once



39C3
POWER CYCLES

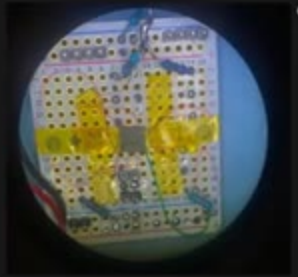
- DWC3 and CDC ACM (serial) implemented in m1n1
- ... but it only seems to work once
- Instantiate DWC3 in Linux
- ... but it also only seems to work once



39C3 POWER CYCLES

- DWC3 and CDC ACM (serial) implemented in m1n1
- ... but it only seems to work once
- Instantiate DWC3 in Linux
- ... but it also only seems to work once
- ... and it only works in device mode

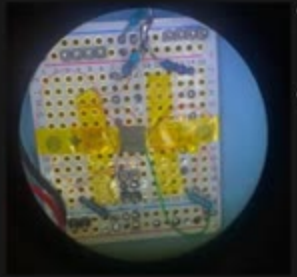




- Type-C can carry up to 240W these days

39C3
POWER CYCLES

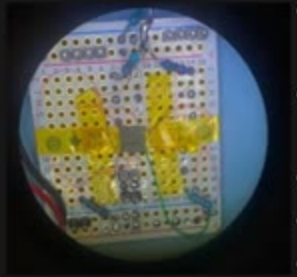




- Type-C can carry up to 240W these days
- Has to be negotiated to prevent smoke

39C3
POWER CYCLES

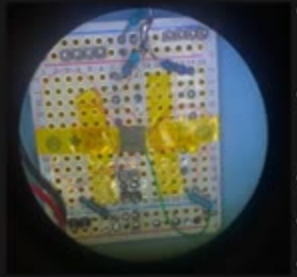




- Type-C can carry up to 240W these days
- Has to be negotiated to prevent smoke
- Weird protocol over CC lines
- Also handles alternate modes
- Handled by a TI-based USB-PD controller

39C3
POWER CYCLES

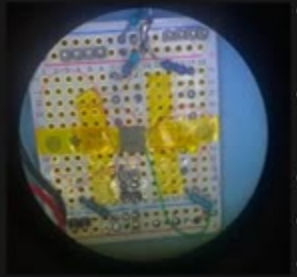




- Type-C can carry up to 240W these days
- Has to be negotiated to prevent smoke
- Weird protocol over CC lines
- Also handles alternate modes
- Handled by a TI-based USB-PD controller

39C3
POWER CYCLES





- Type-C can carry up to 240W these days
- Has to be negotiated to prevent smoke
- Weird protocol over CC lines
- Also handles alternate modes
- Handled by a TI-based USB-PD controller
- Linux driver available!
- Host mode works as well



```
1 MMIO: R.4   PIPEHANDLER_AON_GEN = 0x1 (DWC3_FORCE_CLAMP_EN=0, DWC3_RESET_N=1)
2 MMIO: W.4   PIPEHANDLER_AON_GEN = 0x10 (DWC3_FORCE_CLAMP_EN=1, DWC3_RESET_N=0)
3 MMIO: R.4   USB2PHY_CTL = 0x4 (RESET=0, PORT_RESET=0, APB_RESETN=1, SIDDQ=0)
4 MMIO: W.4   USB2PHY_CTL = 0xb (RESET=1, PORT_RESET=1, APB_RESETN=0, SIDDQ=1)
5 MMIO: R.4   USB2PHY_MISCTUNE = 0x8c0013 (APBCLK_GATE_OFF=0, REFCLK_GATE_OFF=0)
6 MMIO: W.4   USB2PHY_MISCTUNE = 0x608c0013 (APBCLK_GATE_OFF=1, REFCLK_GATE_OFF=1)
```

```
1 MMIO: R.4   PIPEHANDLER_AON_GEN = 0x1 (DWC3_FORCE_CLAMP_EN=0, DWC3_RESET_N=1)
2 MMIO: W.4   PIPEHANDLER_AON_GEN = 0x10 (DWC3_FORCE_CLAMP_EN=1, DWC3_RESET_N=0)
3 MMIO: R.4   USB2PHY_CTL = 0x4 (SUSPM=0, PORT_RESET=0, APB_RESETN=1, SIDDQ=0)
4 MMIO: W.4   USB2PHY_CTL = 0xb (SUSPM=1, PORT_RESET=1, APB_RESETN=0, SIDDQ=1)
5 MMIO: R.4   USB2PHY_MISCTUNE = 0x8c0013 (APBCLK_GATE_OFF=0, REFCLK_GATE_OFF=0)
6 MMIO: W.4   USB2PHY_MISCTUNE = 0x408c0013 (APBCLK_GATE_OFF=1, REFCLK_GATE_OFF=1)
```

39C3

POWER CYCLES



```
1 MMIO: R.4   PIPEHANDLER_AON_GEN = 0x1 (DWC3_FORCE_CLAMP_EN=0, DWC3_RESET_N=1)
2 MMIO: W.4   PIPEHANDLER_AON_GEN = 0x10 (DWC3_FORCE_CLAMP_EN=1, DWC3_RESET_N=0)
3 MMIO: R.4   USB2PHY_CTL = 0x4 (RESET=0, PORT_RESET=0, APB_RESETN=1, SIDDQ=0)
4 MMIO: W.4   USB2PHY_CTL = 0xb (RESET=1, PORT_RESET=1, APB_RESETN=0, SIDDQ=1)
5 MMIO: R.4   USB2PHY_MISCTUNE = 0x8c0013 (APBCLK_GATE_OFF=0, REFCLK_GATE_OFF=0)
6 MMIO: W.4   USB2PHY_MISCTUNE = 0x608c0013 (APBCLK_GATE_OFF=1, REFCLK_GATE_OFF=1)
```


• USB-PD chip triggers hotplug interrupt with target role
• Power-up and bring up DWC3 and XHCI
• USB device probably works!
• USB-PD chip trigger unplug interrupt
• Tear down XHCI and DWC3 and assert hard reset



- USB1/2 ✅
- USB3
- DisplayPort
- Thunderbolt / USB4



39C3
POWER CYCLES

- USB1/2 ✅
- USB3 ✅
- DisplayPort
- Thunderbolt / USB4



39C3
POWER CYCLES

- USB1/2 ✅
- USB3 ✅
- DisplayPort
- Thunderbolt / USB4



39C3

POWER CYCLES

[redacted] just got my #asahilinux in #mac up and running ;-) Installation worked like a charm. I forgot a major to do on my expectation list:
- use my usb-c 4k monitor which seems to become a challenge...



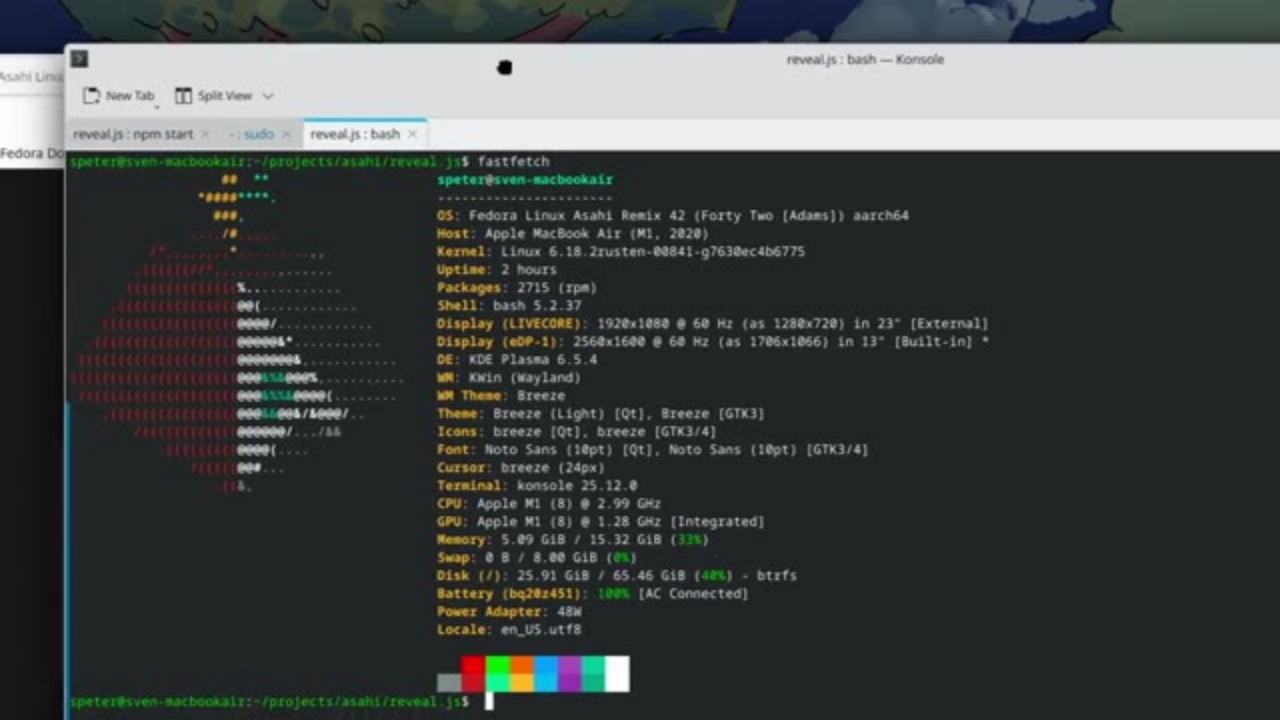


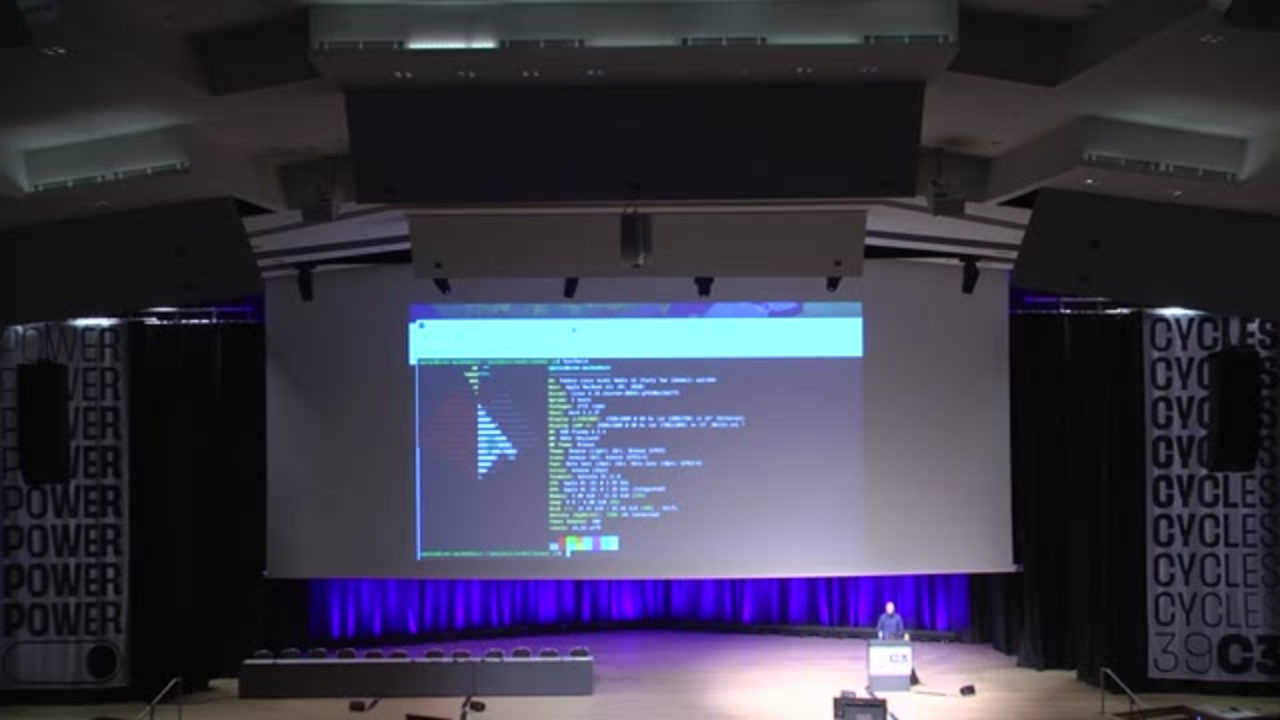
Does Asahi Linux support HDMI or DisplayPort output yet? tried to connect my M1 Mac to my 1440p monitor using an HDMI adapter and a USB-C to DisplayPort adapter, but nothing is working. [redacted]

submitted 10 hours ago by [redacted]

33 comments share save hide report




reveal.js : bash — Konsole
New Tab
Split View
reveal.js : npm start
sudo
reveal.js : bash
speter@sven-macbookair:~/projects/asahi/reveal.js$ fastfetch
speter@sven-macbookair
OS: Fedora Linux Asahi Remix 42 (Forty Two [Adams]) aarch64
Host: Apple MacBook Air (M1, 2020)
Kernel: Linux 6.18.2rusten-00841-g7630ec4b6775
Uptime: 2 hours
Packages: 2715 (rpm)
Shell: bash 5.2.37
Display (LIVECORE): 1920x1080 @ 60 Hz (as 1280x720) in 23" [External]
Display (eDP-1): 2560x1600 @ 60 Hz (as 1706x1066) in 13" [Built-in] *
DE: KDE Plasma 6.5.4
WM: KWin (Wayland)
WM Theme: Breeze
Theme: Breeze (Light) [Qt], Breeze [GTK3]
Icons: breeze [Qt], breeze [GTK3/4]
Font: Noto Sans (10pt) [Qt], Noto Sans (10pt) [GTK3/4]
Cursor: breeze (24px)
Terminal: konsole 25.12.0
CPU: Apple M1 (8) @ 2.99 GHz
GPU: Apple M1 (8) @ 1.28 GHz [Integrated]
Memory: 5.09 GiB / 15.32 GiB (33%)
Swap: 0 B / 8.00 GiB (0%)
Disk (/): 25.91 GiB / 65.46 GiB (40%) - btrfs
Battery (bq20z451): 100% [AC Connected]
Power Adapter: 48W
Locale: en_US.utf8
speter@sven-macbookair:~/projects/asahi/reveal.js$


POWER
POWER
POWER
POWER
POWER
CYCLES
CYCLES
CYCLES
CYCLES
CYCLES
CYCLES
CYCLES
39C3



39C3
POWER CYCLES

DISPLAYPORT

https://github.com/AsahiLinux/linux/tree/fairydust
meant for developers to help us iron out the last bugs



39C3
POWER CYCLES

DISPLAYPORT

https://github.com/AsahiLinux/linux/tree/fairydust
meant for developers to help us iron out the last bugs



39C3
POWER CYCLES

DISPLAYPORT

https://github.com/AsahiLinux/linux/tree/fairydust
meant for developers to help us iron out the last bugs



THE FUTURE: M3/M4/ M5



39C3
POWER CYCLES

M3



- A few changes:
  - New chicken bits initialization sequence
  - PD controller moved from I2C to SPMI
  - Minor interrupt controller and NVMe changes
  - Co-processor communication changes
  - ...



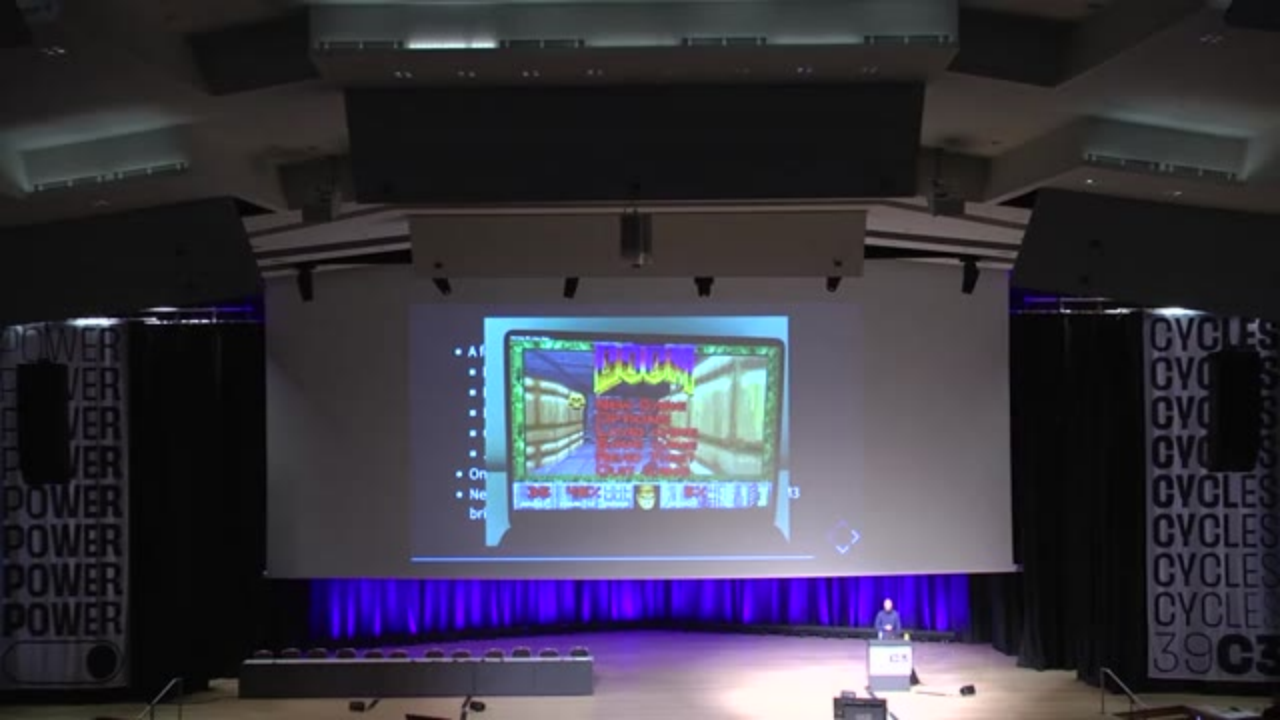
- A f
  - I
  - I
  - I
  - C
  - J
- On
- Ne
  bri

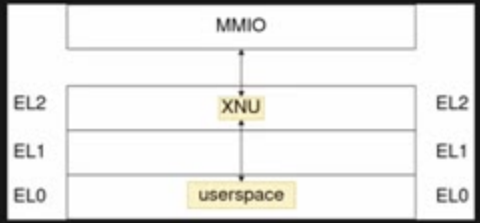


- A f
  - I
  - I
  - I
  - C
  - J
- On
- Ne
  bri

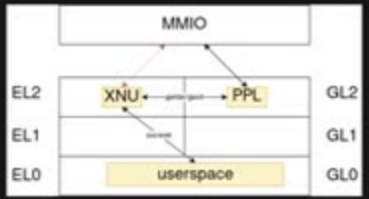


13


DOOM
POWER
CYCLES
39C3


MMIO
EL2
XNU
EL2
EL1
EL1
EL0
userspace
EL0





- GL: guarded levels
- same pagetables but different permissions
- Only GL2 can modify pagestables
- Custom ARM extension



- On M1/M2/M3: Custom extension is available and we reverse engineered and enabled it to virtualize XNU under m1n1
- M4/M5 with macho boot objects (i.e. XNU): EL2 with MMU enabled and Apple Code in GL2
- M4/M5 with raw boot objects (i.e. Linux): EL2 without MMU enabled, all ISA extensions disabled
- Fine for Linux, but breaks our reverse engineering tools :-(

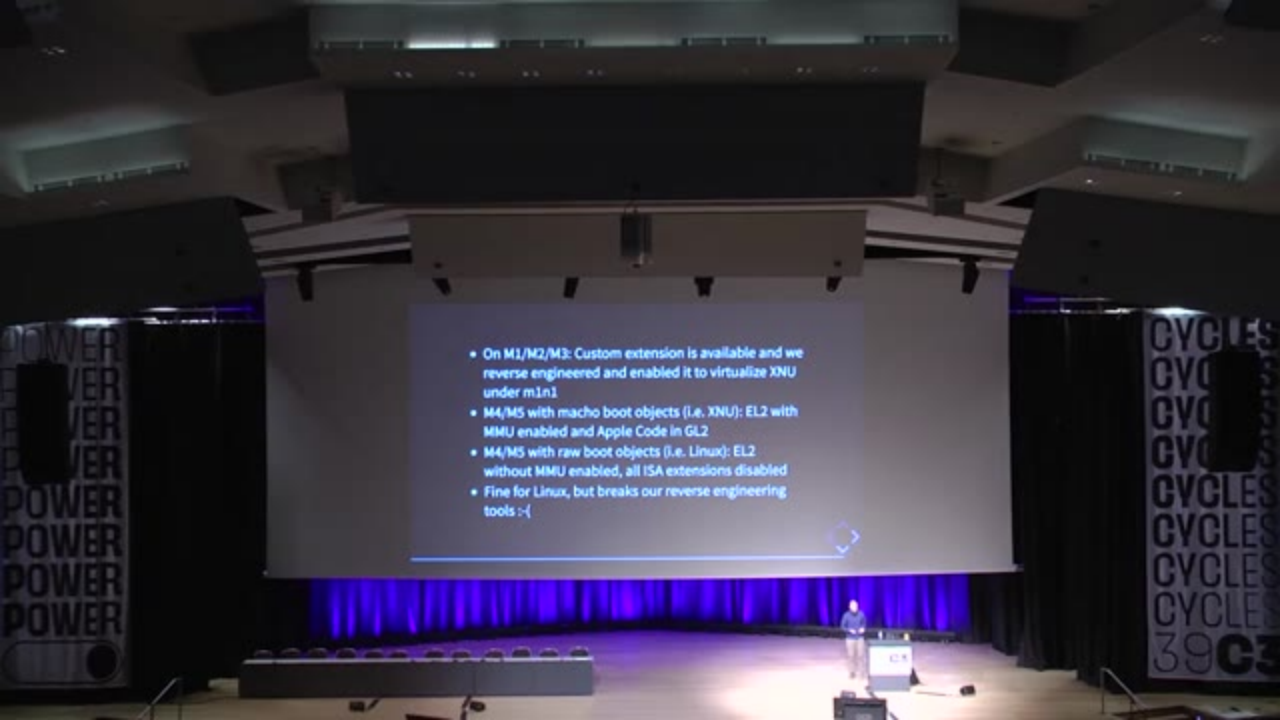


- On M1/M2/M3: Custom extension is available and we reverse engineered and enabled it to virtualize XNU under m1n1
- M4/M5 with macho boot objects (i.e. XNU): EL2 with MMU enabled and Apple Code in GL2
- M4/M5 with raw boot objects (i.e. Linux): EL2 without MMU enabled, all ISA extensions disabled
- Fine for Linux, but breaks our reverse engineering tools :-(



- On M1/M2/M3: Custom extension is available and we reverse engineered and enabled it to virtualize XNU under m1n1
- M4/M5 with macho boot objects (i.e. XNU): EL2 with MMU enabled and Apple Code in GL2
- M4/M5 with raw boot objects (i.e. Linux): EL2 without MMU enabled, all ISA extensions disabled
- Fine for Linux, but breaks our reverse engineering tools :-(

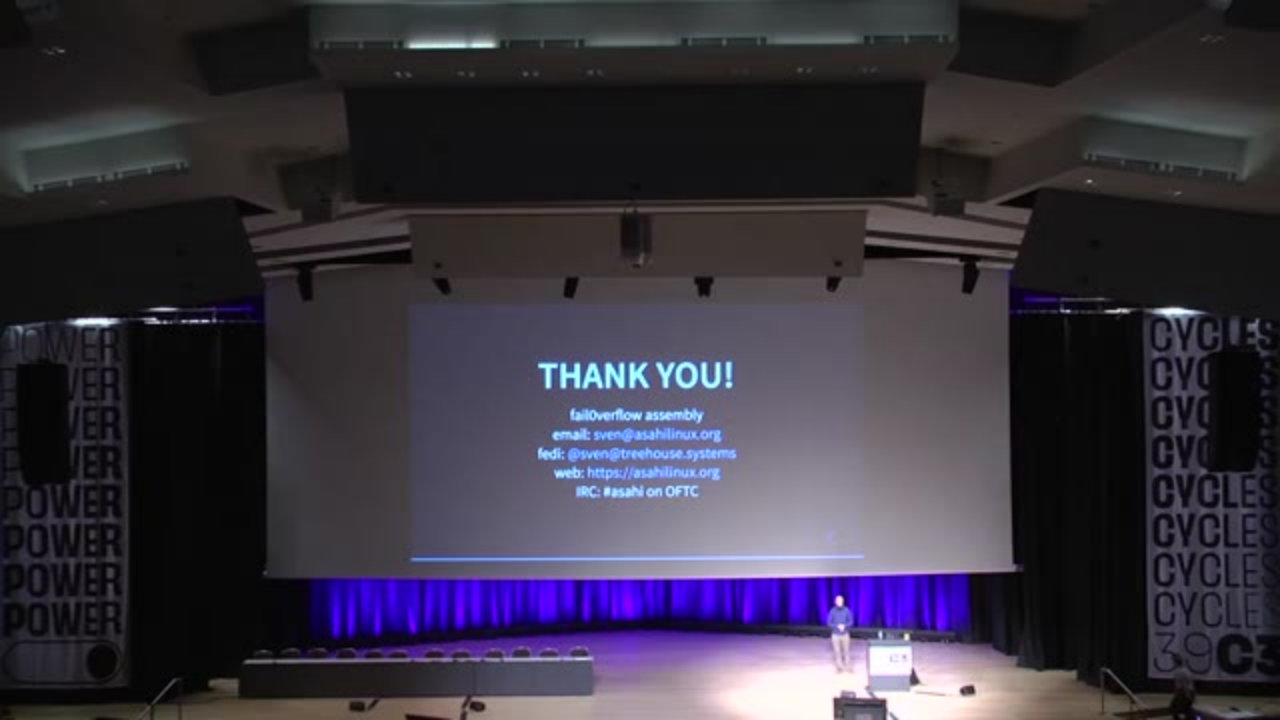



• On M1/M2/M3: Custom extension is available and we reverse engineered and enabled it to virtualize XNU under m1n1
• M4/M5 with macho boot objects (i.e. XNU): EL2 with MMU enabled and Apple Code in GL2
• M4/M5 with raw boot objects (i.e. Linux): EL2 without MMU enabled, all ISA extensions disabled
• Fine for Linux, but breaks our reverse engineering tools :-(
POWER
CYCLES
39C3

THANK YOU!

fail0verflow assembly
email: sven@asahilinux.org
fedi: @sven@treehouse.systems
web: https://asahilinux.org
IRC: #asahi on OFTC



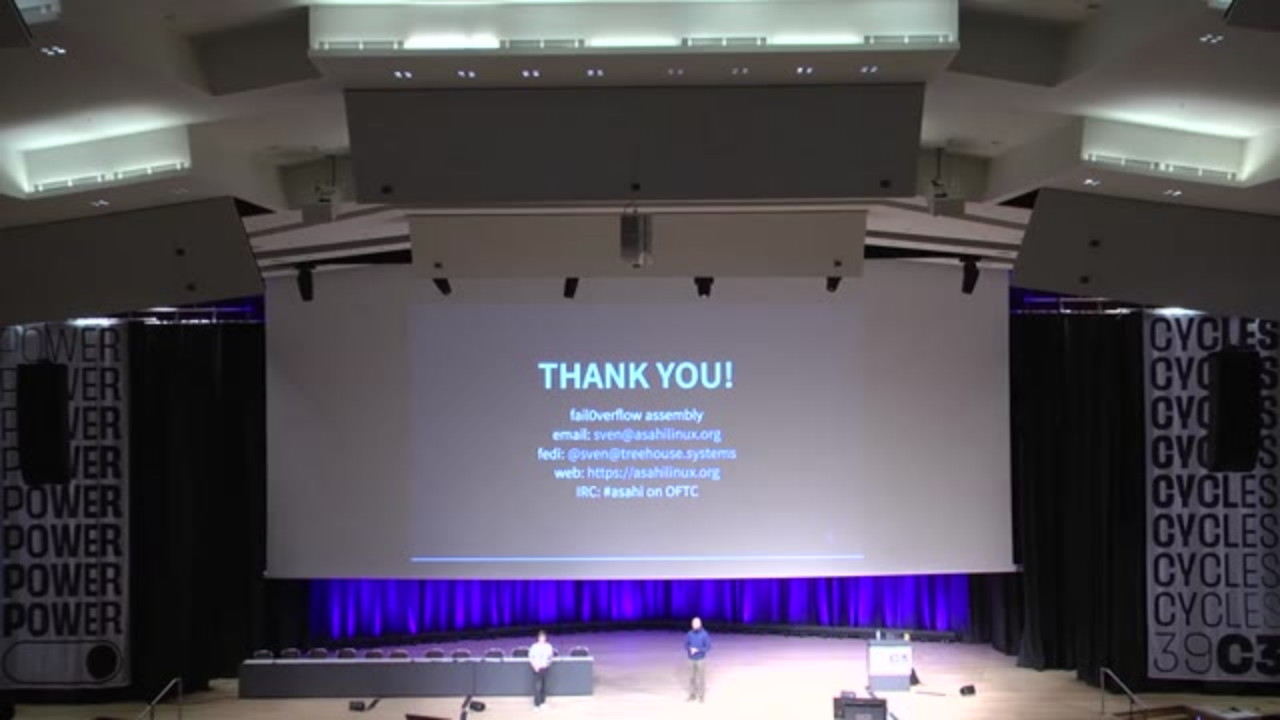

THANK YOU!
fail0verflow assembly
email: sven@asahilinux.org
fedi: @sven@treehouse.systems
web: https://asahilinux.org
IRC: #asahi on OFTC


THANK YOU!
fail0verflow assembly
email: sven@asahilinux.org
fedi: @sven@treehouse.systems
web: https://asahilinux.org
IRC: #asahi on OFTC
POWER
CYCLES
39C3








THANK YOU!
fail0verflow assembly
email: sven@asahilinux.org
fedi: @sven@treehouse.systems
web: https://asahilinux.org
IRC: #asahi on OFTC
POWER
CYCLES
39C3



More on
media.ccc.de

Recorded by VOC
together with







Attribution

