

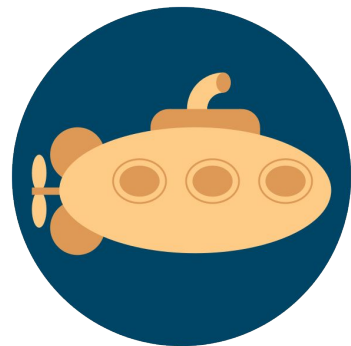


Linaro
Developer Services

U-Boot for modern Qualcomm phones


*About the lowest common denominator and empowering
users*

Caleb Connolly <caleb.connolly@linaro.org>



U-Boot

About me

- Caleb (they/them)
- FOSS enjoyer since 2018
- Kernel engineer @ Linaro (Qualcomm Ecosystem Team)
 - Happy to hack on pretty much anything (as long as it's not userspace)
 - Especially partial to things that improve UX
- postmarketOS core team member
 - Maintain support for Snapdragon 845 devices
 - Plotting new ways to keep your embedded devices out of landfill
- Maintainer of Qualcomm platform support in U-Boot
-  @cas@social.treehouse.systems

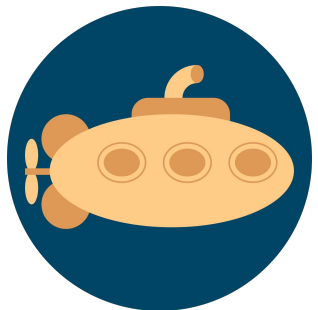
In this talk

1. Demonstrate why running Linux on Android phones suuucks
2. The magic of UEFI
3. U-Boot as a UEFI bootloader
4. The “how” of U-Boot on Qualcomm devices
5. State of Qualcomm support in U-Boot
6. Demo
7. Upstreaming status
8. Supporting a new SoC



Let's play... Odd one out

[● ◀]
systemd
boot



U-Boot



LinuxBoot

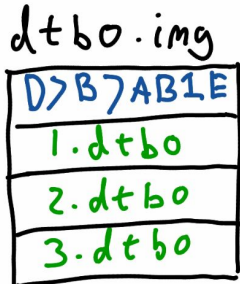
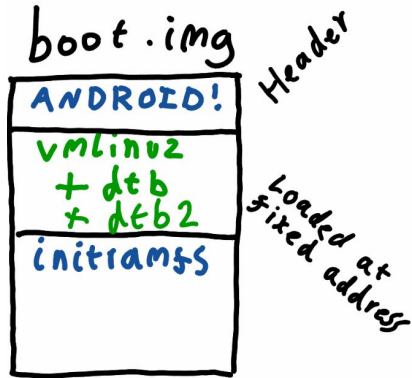


Bootloader

The software(s) responsible for loading the kernel, initramfs, and devicetree, and then jumping to the kernel.



Booting on Android (Qualcomm edition)



ABL

- Load boot.img
- find matching dtb
- Load dtbo.img
- Apply overlay
- JUMP to kernel

"Features"

- No Multiboot
- only custom A/B rollback support
- heavy integration cost for distros
- No Post-boot services



Booting on Android (Qualcomm edition)

boot.img

ANDROID!
vmlinux + dtb + dtb2
initramfs

- 5 versions so far
- Different untested codepaths for each!

dtbo.img

D>B7AB1E
1. dtbo
2. dtbo
3. dtbo

- Doesn't work on upstream DT
- Sometimes can be bypassed
- Sometimes need a custom empty DTBO image (Device specific!)

ABL

- Load boot.img
- ↙ find matching dtb

Custom non-standard DT properties that many OEMs don't even follow

- No feedback on failure - just drops to fastboot unless UART

"Features"

- No multiboot
- only custom A/B rollback support
- heavy integration cost for distros
- No post-boot services

- will softbrick your device without userspace intervention
- no way to disable!

What's hiding...

```
if [ "${deviceinfo_bootimg_append_seandroidenforce}" = "true" ]; then  
  log_arrow "initramfs: appending 'SEANDROIDENFORCE' to boot.img"  
  printf "SEANDROIDENFORCE" >> "$_bootimg"  
fi
```

[Firmware Bug]: Kernel image misaligned at boot, please fix your bootloader!

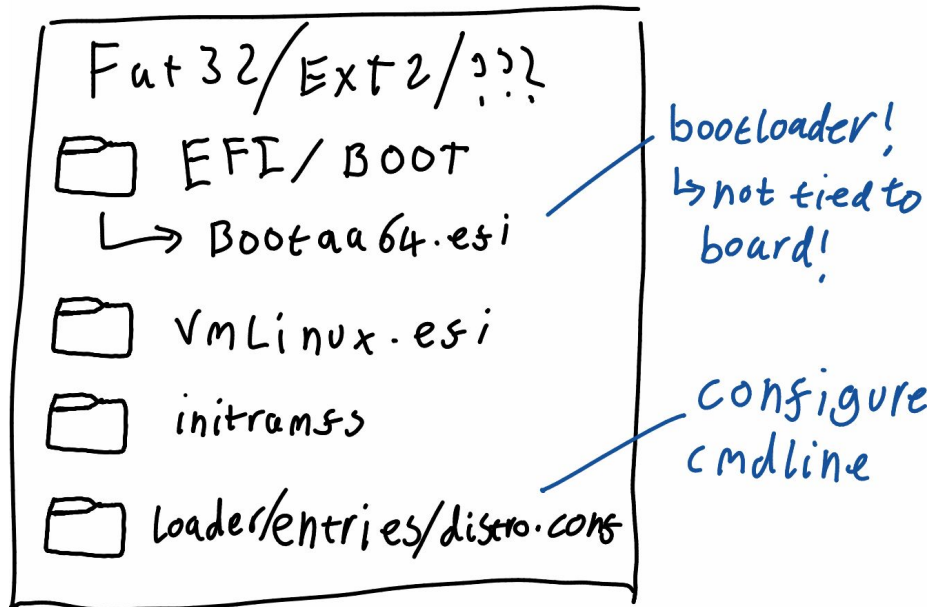
```
[1210] read from dtbo, 0x800000 bytes at Offset 0x0, partition size 406847488  
[1220] data abort, halting  
[1220] r0 0x8f900000 r1 0x00800000 r2 0x8fed8138 r3 0x00800000  
[1220] r4 0x00800000 r5 0x8f6d8138 r6 0x18400000 r7 0x00000000  
[1220] r8 0x8f628d60 r9 0x00000200 r10 0x8f6d9b98 r11 0x00000000  
[1220] r12 0x0000000d usp 0x00000000 ulr 0x00000000 pc 0x8f630e74  
[1220] spsr 0x80000153  
[1220] fiq r13 0x8f6b8328 r14 0x00000000  
[1220] irq r13 0x8f6bd2e0 r14 0x8f613244  
[1220] *svc r13 0x8f6d7fe8 r14 0x8f611a94  
[1220] und r13 0x8f6b8328 r14 0x00000000  
[1220] sys r13 0x00000000 r14 0x00000000  
[1220] bottom of stack at 0x8f6d7fe8:  
0x8f6d7fe8: 00000000 00800000 00000000 8f66b748 |.....H.f.|  
0x8f6d7ff8: 00000000 8f6ff700 00800000 18400000 |.....o.....|  
0x8f6d8008: 00000000 00000000 00000000 00800000 |.....|  
0x8f6d8018: 00000000 8f628d60 00000000 00000000 |.....b.....|  
0x8f6d8028: 18400000 00000000 00000000 00000000 |.....|  
0x8f6d8038: 18400000 00000000 00000000 00000000 |.....|  
0x8f6d8048: 8f6d8174 00000200 00000000 91b33359 |t.m.....Y...|  
0x8f6d8058: 00004000 00000000 00800000 00000000 |.....|  
[1220] HALT: reboot into dLoad mode...
```

```
"${_mkbootimg}" \  
  --kernel "${_kernelfile}" \  
  --ramdisk "$_ramdisk" \  
  --base "${_base}" \  
  --second_offset "${deviceinfo_flash_offset_second}" \  
  --cmdline "$(get_cmdline)" \  
  --kernel_offset "${deviceinfo_flash_offset_kernel}" \  
  --ramdisk_offset "${deviceinfo_flash_offset_ramdisk}" \  
  --tags_offset "${deviceinfo_flash_offset_tags}" \  
  --pagesize "${deviceinfo_flash_pagesize}" \  
  ${_second} \  
  ${_dt_img} \  
  ${_header_v2} \  
  ${deviceinfo_bootimg_custom_args} \  
  -o "$_bootimg" || exit 1
```




Booting with UEFI

EFI System Partition



Any number of ESP's

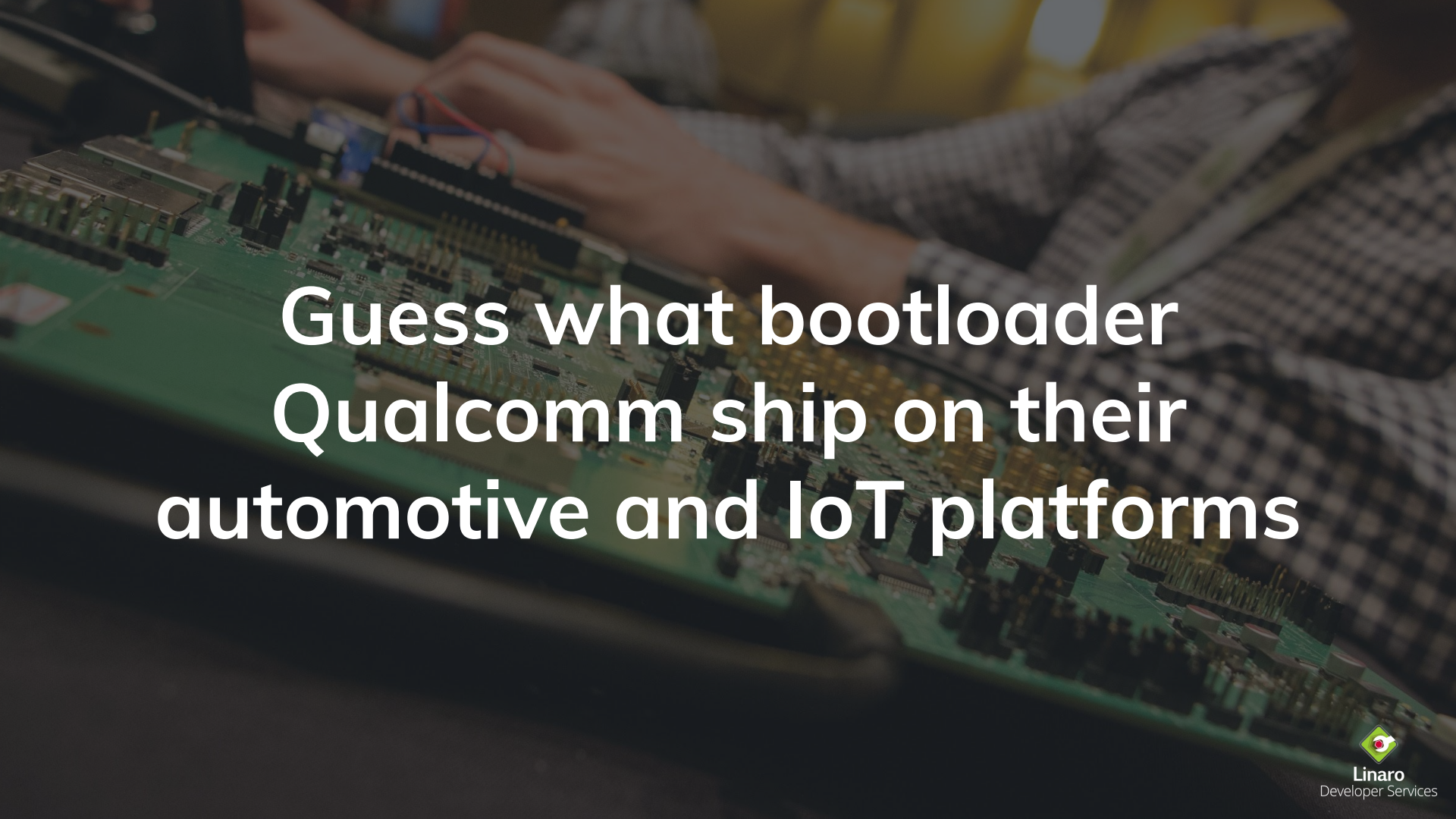
- Already universally supported
- The distro has full control over the bootloader!



Booting with UEFI

- Many x86 vendors (and those doing UEFI in the Arm space) still f*ck it up
- Still some limitations (e.g. SetVariable() doesn't work on devices that can't dedicate SPI flash to UEFI)
- If done right, an obvious winner

USE FOSS
BOOTLOADERS !!!

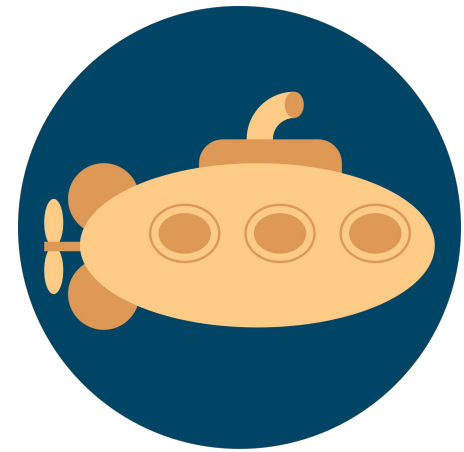


**Guess what bootloader
Qualcomm ship on their
automotive and IoT platforms**



U-Boot

- A very cool, and GPL-2 licensed bootloader
- Supports many architectures and devices
- Linux driver compatibility
- Devicetree!!!
- Highly adaptable
- UEFI - passes SystemReady
- Not so great Qualcomm support
- ... Until now!



U-Boot



Booting U-Boot

- We can't replace the Android bootloader :(
- ... most of the time anyway
- Code is hashed and the hash is signed by a private key owned by the vendor - the public key is burnt into the SoC at the factory.
- Exploits? Possibly...
- But we can chainload!
 - And glean a bunch of useful info from ABL
 - Build with `CONFIG_LINUX_KERNEL_IMAGE_HEADER=y`



Qualcomm support in U-Boot

- Provided in arch/arm/mach-snapdragon
 - MSM8916, MSM8996, SDM845, SM6115, QCM2290, SM8250, SC7280, SM8550 WIP
 - And more?
- Almost compatible with upstream DT
 - Deviations are tracked separately in \$DEVICE-u-boot.dtsi include files
 - Runtime DT fixup for USB
- No board specific code for platforms added after 2017
 - Read memory map from DT
 - Dynamically allocate load buffers
 - One build target for all supported platforms and devices!
- Support for USB, UFS, and newer MMC is making its way upstream
- And much more!
 - Button support, capsule updates, etc.



IoT dev boards

- RB1 (low-end - QCM2290)
 - 4 cores, 2GHz
 - 1/2GB RAM, 8/16GB eMMC
 - \$199
- RB2 (low/mid range - SM6115)
 - 8 cores, 2GHz
 - 2GB RAM, 16GB eMMC
 - \$249
- RB3 (mid range - SDM845)
 - 8 cores, up to 2.8GHz
 - 4GB RAM, 128GB UFS
 - \$400 (no longer sold)
- RB5 (high end - SM8250)
 - 8 cores, up to 2.84GHz
 - 8GB RAM
 - \$545
- All capable of booting from internal storage or USB
- No secureboot!
- Capable of running U-Boot as the primary bootloader
- PoC highly promising
- More work needed in this area



Initial release for SDM845 phones

<https://gitlab.com/sdm845-mainline/u-boot/-/releases/sdm845-phones-v0.1.0>

sdm845-phones-v0.1.0



Assets 9

- Source code (zip) ↓
- Source code (tar.gz) ↓
- Source code (tar.bz2) ↓
- Source code (tar) ↓

Other

- u-boot-fajita.img ↗
- u-boot-axotl.img ↗
- u-boot-enchilada.img ↗
- u-boot-beryllium-tianma.img ↗
- u-boot-beryllium-ebbg.img ↗

Evidence collection

sdm845-phones-v0.1.0-evidences-7444700.json ... 791d4b2b

Collected 7 Jan 2024, 22:55

This initial release provides support for booting from internal UFS storage when a valid EFI System Partition is found (containing `/EFI/B00T/bootaa64.efi`).

The volume and power buttons can be used to navigate menus (including systemd-boot and GRUB).

The U-Boot Console can be accessed by booting with a USB cable attached, interrupting autoboot with the power buttons, and choosing "Enable serial gadget console" from the menu, this will expose a serial device to your computer which can be accessed with any standard console program (picocom, minicom, etc).

Demo!!!

- https://gitlab.com/postmarketOS/pmaports/-/merge_requests/4599
- [https://wiki.postmarketos.org/wiki/Qualcomm_SDM845_\(generic-sdm845\)](https://wiki.postmarketos.org/wiki/Qualcomm_SDM845_(generic-sdm845))
- <https://gitlab.com/sdm845-mainline/u-boot>



Upstreaming status

- Initial cleanup mostly done
- Big refactor and migration to upstream DT on the lists
- USB support for SDM845 on the lists
- UFS in progress
- Separate effort by Sumit Garg to make all upstream DT available by importing devicetree-rebasing repo




Integrating U-Boot - future plans

- “Recent Advancements in U-Boot - Simon Glass” [1]
- Most “standard” devices should “just work” with Linux DT (assuming SoC support)
- Enable support for handling DTB variants (e.g. different display panels)
- Let’s make upstream Qualcomm defconfig friendly to phones!
 - Enable a mobile-friendly boot menu based on chassis type
 - Remap volume/power button keycodes to up/down/enter

[1]: <https://www.youtube.com/watch?v=YlJBsVZjkDI>

Adding a new SoC

- Clock and pinctrl drivers
 - Can be just stubs initially!
- UFS phy configuration data
- Compatible strings for PMIC, SMMU
- Congrats, you can probably boot from UFS now :D

 mastodon.social/@LinaroLtd

Thank you

slides: calebs.dev/fosdem24.pdf
contact: caleb.connolly@linaro.org
website: connolly.tech/



Linaro
Developer Services