WHY DISTRIBUTIONS DO NOT SUPPORT MY DEVICE?

Marcin Juszkiewicz
Software Engineer
2016.03.12

WHAT IS MY DEVICE?

Let’s limit to ARM architecture only

What user/developer can have:

- Developer board
- Consumer electronics
- Chromebook
- Server
- Android powered phone/tablet

WHICH THINGS DISTRIBUTIONS DISLIKE?

I am thinking of Debian, Fedora etc

There are few things:

- Non-standard bootloader
- Partitions layout requirements
- Strange kernel versions
- Strange bootloader versions
- Binary blobs

WHY BOOTLOADER?

Standard bootloader == less work

How ARM device boots:

- 1st stage bootloader starts from in-CPU ROM
- 2nd stage bootloader is loaded from SD (U-Boot SPL)
- 3rd stage bootloader is loaded from SD (U-Boot)
- U-Boot loads configuration from SD
- U-Boot loads kernel, initramfs, dtb from SD
- U-Boot runs kernel

U-BOOT

One bootloader on nearly every ARM device

Benefits for distribution:

- Active development
- Developers opens for suggestions
- Easy to buy set for all supported devices
- One config file for all devices
- Console available in case something goes wrong
- Lot of ways to boot system
- GPL

Partitions layout matters?

One image for all devices

Standard installation image for ARM architecture:

- Partitions in MBR
- First one is ext4 mounted as /boot/ (starts at 4MB from beginning of card)
- Next is swap (size differ)
- Ext4 partition mounted as / (often expanded on first boot to fill card)
- U-Boot SPL is in /boot/ or stored in sectors of first 4MB
- U-Boot and it’s configuration are in /boot/
- Kernel, initramfs and devicetree are in /boot

Raspberry/Pi

PARTITION LAYOUT FOR RASPBERRY/PI

Why one image for all devices does not fit

Raspberry/Pi’s GPU enforces own requirements:

- Partition table in MBR
- First one is vfat mounted in Raspbian as /boot/
- Ext4 partition mounted as /
- Bootloaders and their configurations are in /boot/
- Kernel is in /boot/

STARTING RASPBERRY/PI

How GPU starts device and why vfat partition is required

How Raspberry/Pi boots:

- 1st stage bootloader starts from GPU’s ROM
- 2nd stage bootloader is read from vfat partition of SD (bootcode.bin)
- bootcode.bin loads start.elf
- start.efl loads config.txt, cmdline.txt and kernel.img
- start.efl starts CPU
- start.efl runs kernel

HOW TO HELP WITH RASPBERRY/PI

Booting is weird but can be improved

We can make it a bit closed to standard way one:

- Instead of Linux kernel image we can use U-Boot as kernel.img
- U-Boot knows how to read from ext4
- U-Boot loads configuration from SD
- U-Boot loads kernel, initramfs, dtb from SD
- U-Boot runs kernel

BUT WHAT TO DO WITH VFAT FOR R/PI?

Standard image has only ext4

Changes need to be done:

- Let’s add vfat partition as first one
- Ext4 for /boot/ will be second so we need to change U-Boot configuration
- Swap and rootfs will move but we call them by UUID so nothing to change
- Mount vfat one as /boot/rpi/

Chromebook

PARTITION LAYOUT FOR CHROMEBOOK

We are different and what you will do with it?

Here everything is different:

- Partition table as GPT
- Kernel has own partition type (7f00)
- There are flags for partition priority
- There are flags telling about kernel use

LET’S BOOT CHROMEBOOK

Signed kernels, extra flags and other ideas

How Chromebook boots:

- 1st stage bootloader starts from CPU’s ROM
- 2nd stage bootloader is loaded from SPI flash memory
- Partition table (GPT) is read to find kernel partitions
- Kernel is read from partition with highest priority and proper other flags
- Signature is checked
- Kernel is run

HOW TO HELP WITH CHROMEBOOK?

Booting is weird but can be changed

We can make it a bit closed to standard way one:

- Instead of Linux kernel image we can use U-Boot
- U-Boot knows how to read from ext4
- U-Boot loads configuration from SD
- U-Boot loads kernel, initramfs, dtb from SD
- U-Boot runs kernel

Roseapple/Pi

TOTAL FAILURE

Or how to fail interesting device

What went wrong:

- Linux kernel in 3.10 version
- U-Boot is old too
- No mainline activity
- Kernel repository in “one big commit” style
- Typical “runs Debian” on website plus modified Debian image

What can be done?

KERNEL

Things not in mainline do not exist

What to avoid:

- Kernel older than 2 releases (which means 4.3 – 4.5 now)
- Hundreds of patches to add support for device
- Own solutions for existing subsystems
- Changes altering code for other devices

CHANGING DISTRIBUTION’S KERNEL

Or how to get support

Some suggestions:

- Send own patches to proper kernel mailing lists
- Improve, rewrite according to suggestions
- Publish your repository in public
- Base your kernel on Torvalds’ repository
- Build own kernels with distribution’s configuration
- Provide set of ready to use patches for in-distro kernel maintainers
- Integrate generation of own config files with distribution’s tools

BINARY BLOBS

Bleh

How to help:

- Take care of license (there has to be right for re-distribution)
- Send blobs to linux-firmware repository (all distributions package it)
- Suggest opening blobs source code and tools to build it

redhat.

THANK YOU

plus.google.com/+MarcinJuszkiewicz

facebook.com/marcin.juszkiewicz

linkedin.com/in/marcinjuszkiewicz