redhat.

DLACZEGO DYSTRYBUCJE NIE OBSŁUGUJĄ MOJEGO URZĄDZENIA?

Marcin Juszkiewicz
Software Engineer
2016.03.12

CO TO JEST MOJE URZĄDZENIE?

Ograniczmy się do architektury ARM

Co użytkownik/developer może mieć:

- Developer board
- Elektronika użytkowa
- Chromebook
- Serwer
- Telefon/tablet z Androidem

CZEGO DYSTRYBUCJE NIE LUBIĄ?

Mam na myśli Debiana, Fedorę itp.

Jest kilka rzeczy:

- Niestandardowy bootloader
- Wymagania co do układu partycji
- Dziwne wersje kernela
- Dziwne wersje bootloadera
- Binarne bloby

DLACZEGO BOOTLOADER?

Standardowy bootloader == mniej roboty

Przebieg startu na urządzeniu ARM:

- 1st stage bootloader startuje z ROM w procesorze
- 2nd stage bootloader wczytywany jest z karty SD (U-Boot SPL)
- 3rd stage bootloader wczytywany jest z karty SD (U-Boot)
- U-Boot ładuje konfigurację z SD
- U-Boot ładuje kernel, initramfs, dtb z SD
- U-Boot uruchamia kernel

U-BOOT

Jeden bootloader na prawie każde urządzenie ARM

Korzyści dla dystrybucji:

- Stały, aktywny rozwój
- Programiści otwarci na propozycje
- Łatwo zbudować zestaw dla wszystkich wspieranych urządzeń
- Jeden plik konfiguracyjny dla wszystkich urządzeń
- Konsola dostępna jak coś idzie źle przy uruchamianiu
- Wiele metod uruchamiania systemu
- GPL

UKŁAD PARTYCJI MA ZNACZENIE?

Jeden obraz dla wszystkich urządzeń

Standardowy obraz instalacyjny dla architektury ARM:

- Partycje w układzie MBR
- Pierwsza to ext4 montowany jako /boot/ (zaczyna się na 4MB od początku karty)
- Potem jest swap (różny rozmiar)
- Partycja ext4 montowana jako / (często rozszerzana na całą kartę przy pierwszym starcie)
- U-Boot SPL jest albo w /boot/ albo zapisany sektorami w pierwszych 4MB
- U-Boot i jego konfiguracja jest w /boot/
- Kernel, initramfs i devicetree są w /boot

Raspberry/Pi

UKŁAD PARTYCJI DLA RASPBERRY/PI

Czyli dlaczego jeden obraz dla wszystkich urządzeń nie pasuje

GPU z Raspberry/Pi narzuca swoje wymagania:

- Partycje w układzie MBR
- Pierwsza to vfat montowana w Raspbianie jako /boot/
- Partycja ext4 montowana jako /
- Bootloadery i ich konfiguracja są w /boot/
- Kernel jest w /boot/

ODPALAMY RASPBERRY/PI

Czyli jak GPU uruchamia urządzenie i dlaczego partycja vfat jest wymagana

Przebieg startu na urządzeniu Raspberry/Pi:

- 1st stage bootloader startuje z ROM w GPU
- 2nd stage bootloader wczytywany jest z partycji vfat karty SD (bootcode.bin)
- bootcode.bin wczytuje start.elf
- start.efl wczytuje config.txt, cmdline.txt i kernel.img
- start.efl uruchamia CPU
- start.efl uruchamia kernel

JAK POMÓC Z RASPBERRY/PI

Bootowanie jest dziwne ale można coś zmienić

Możemy zbliżyć bootowanie do normalnego:

- Zamiast jądra Linux podkładamy U-Boota jako kernel.img
- U-Boot wie jak odczytać z ext4
- U-Boot ładuje konfigurację z SD
- U-Boot ładuje kernel, initramfs, dtb z SD
- U-Boot uruchamia kernel

ALE CO Z TYM VFAT DLA R/PI?

Standardowy obraz ma tylko ext4

Trzeba wprowadzić zmiany:

- Dodać partycję vfat jako pierwszą
- Ext4 na /boot/ będzie drugi więc musimy zmienić konfigurację U-Boota
- Swap i rootfs się przesuną ale odwołujemy się po UUID więc nic się nie zmienia
- Vfat montujemy jako /boot/rpi/

Chromebook

UKŁAD PARTYCJI DLA CHROMEBOOKA

Jesteśmy inni i co nam zrobicie?

Tu jest całkiem inaczej

- Partycje w układzie GPT
- Jądro ma własny typ partycji (7f00)
- Są flagi określające priorytet partycji
- Są flagi określające czy używać danego jądra

PODPALAMY CHROMEBOOKA

Podpisane jądra, extra flagi i inne pomysły

Przebieg startu na chromebooku:

- 1st stage bootloader startuje z ROM w CPU
- 2nd stage bootloader wczytywany jest z pamięci SPI flash
- Odczytywana jest tablica partycji (GPT) w poszukiwaniu partycji z jądrem
- Odczytywane jest jądro z partycji z najwyższym priorytetem i odpowiednimi flagami
- Sprawdzany jest podpis
- Uruchamiany jest kernel

JAK POMÓC Z CHROMEBOOKIEM?

Bootowanie jest dziwne ale można coś zmienić

Możemy zbliżyć bootowanie do normalnego:

- Zamiast jądra Linux podkładamy U-Boota
- U-Boot wie jak odczytać z ext4
- U-Boot ładuje konfigurację z SD
- U-Boot ładuje kernel, initramfs, dtb z SD
- U-Boot uruchamia kernel

Roseapple/Pi

TOTALNA PORAŻKA

Czyli jak można zepsuć ciekawy produkt

Co zrobiono źle:

- Jądro w wersji 3.10
- U-Boot też dość leciwy
- Zero łatek wysłanych do mainline
- Repozytorium z kodem jądra w stylu “one big commit”
- Zwyczajowe “runs Debian” na stronie a obraz własny

Co można zrobić?

KERNEL

Czego nie ma w mainline to nie istnieje

Czego unikać:

- Kernel starszy niż dwa ostatnie wydania (czyli poniżej 4.3 – 4.5-rc)
- Setki łatek dodających obsługę urządzenia
- Własne rozwiązania istniejących podsystemów
- Zmiany dotykające kodu innych urządzeń

ZMIENIAMY KERNEL DYSTRYBUCJI

Czyli jak uzyskać wsparcie

Kilka sugestii:

- Wysyłamy swoje łatki na listy kernelowe
- Poprawiamy, przepisujemy wedlug uzyskanych sugestii
- Publikujemy swoje repozytorium w miejscu publicznym
- Rozwijamy jądro na bazie repozytorium Torvaldsa
- Budujemy własne jądra z konfiguracją z dystrybucji
- Opiekunom paczki z jądrem prezentujemy zestaw gotowych łatek
- Integrujemy generowanie swoich plików konfiguracyjnych z narzędziami z dystrybucji

BINARNE BLOBY

Bleh

Jak tu pomóc:

- Uporządkować licencję (musi być prawo do dowolnego dystrybuowania)
- Przesłać do włączenia do linux-firmware (wszystkie dystrybucje to paczkują)
- Zasugerować twórcom otwarcie źródeł oraz narzędzi do zbudowania

redhat.

DZIĘKUJĘ

plus.google.com/+MarcinJuszkiewicz

facebook.com/marcin.juszkiewicz

linkedin.com/in/marcinjuszkiewicz