Kexec

Ready for Embedded Linux

Magnus Damm
magnus.damm@gmail.com

Renesas Electronics Corp.

April 2010
Outline

Bootloader Basics
- Overview & Use Cases
- Inside the Bootloader
- Four Examples

Kexec-based Bootloaders
- Introduction & Motivation
- Hardware Setup Code
- Kernel Configuration & Tuning
- Optimizing User Space

Real World Examples
- SH-Based Boards
- ARM-Based Boards
Outline

Bootloader Basics
- Overview & Use Cases
- Inside the Bootloader
- Four Examples

Kexec-based Bootloaders
- Introduction & Motivation
- Hardware Setup Code
- Kernel Configuration & Tuning
- Optimizing User Space

Real World Examples
- SH-Based Boards
- ARM-Based Boards
Outline

Bootloader Basics
Overview & Use Cases
   Inside the Bootloader
   Four Examples

Kexec-based Bootloaders
   Introduction & Motivation
   Hardware Setup Code
   Kernel Configuration & Tuning
   Optimizing User Space

Real World Examples
   SH-Based Boards
   ARM-Based Boards
Overview

Setup Hardware

Load Kernel

Start Kernel
Use Cases

Development:
- Load Kernel Over Ethernet/USB

Standalone:
- Load Kernel From NAND/NOR/MMC/USB
Outline

Bootloader Basics
Overview & Use Cases
Inside the Bootloader
Four Examples

Kexec-based Bootloaders
Introduction & Motivation
Hardware Setup Code
Kernel Configuration & Tuning
Optimizing User Space

Real World Examples
SH-Based Boards
ARM-Based Boards
Inside the Bootloader

Hardware Setup (Assembly + C):
- Setup CS Memory Windows
- Initialize Caches & MMU
- Configure Clocks & GPIOs
- Setup System RAM
- Configure I/O Devices
Inside the Bootloader

Load Kernel (C only):
- “Driver Model”
- Network Stack / Filesystem
- Kernel File Format Parser

Start Kernel (C + Assembly):
- Prepare Kernel Parameters
- Execute Kernel
Outline

Bootloader Basics
  Overview & Use Cases
  Inside the Bootloader
  Four Examples

Kexec-based Bootloaders
  Introduction & Motivation
  Hardware Setup Code
  Kernel Configuration & Tuning
  Optimizing User Space

Real World Examples
  SH-Based Boards
  ARM-Based Boards
Example 1: Good Old PC

BIOS:
- Located in ROM / NOR Flash
- Performs Hardware Setup
- Loads Bootloader from HDD

Bootloader:
- Executes from RAM
- Loads Kernel from HDD
Example 2: Embedded NOR Flash

[SoC + RAM + NOR Flash + I/O]

Bootloader:

- Located in NOR Flash
- Performs Hardware Setup
- Loads Kernel from NOR Flash
Example 3: Embedded NOR Flash + NAND / MMC

[SoC + RAM + NOR Flash + NAND Flash / MMC + I/O]

Bootloader:
- Located in NOR Flash
- Performs Hardware Setup
- Loads Kernel from NAND Flash / MMC
Example 4: Embedded NAND Flash / MMC

[SoC + RAM + NAND Flash / MMC + I/O]

Mask ROM:
- Located inside SoC
- Loads Bootloader from NAND Flash / MMC

Bootloader:
- Executes from On-chip RAM
- Performs Hardware Setup
- Loads Kernel from NAND Flash / MMC
Outline

Bootloader Basics
  Overview & Use Cases
  Inside the Bootloader
  Four Examples

Kexec-based Bootloaders
  Introduction & Motivation
  Hardware Setup Code
  Kernel Configuration & Tuning
  Optimizing User Space

Real World Examples
  SH-Based Boards
  ARM-Based Boards
Outline

Bootloader Basics
  Overview & Use Cases
  Inside the Bootloader
  Four Examples

Kexec-based Bootloaders
  Introduction & Motivation
  Hardware Setup Code
  Kernel Configuration & Tuning
  Optimizing User Space

Real World Examples
  SH-Based Boards
  ARM-Based Boards
What is Kexec?

"kexec is a system call that implements the ability to shutdown your current kernel, and to start another kernel. It is like a reboot but it is independent of the system firmware..."

Configuration help text in Linux-2.6.17
What is Kexec?

Kexec is a combination of kernel code and user space code:

- Linux kernel support available through `CONFIG_KEXEC`.
- `kexec-tools` provides the user space tool `kexec`.

Simple Kexec example to reboot into “zImage”:

```
# kexec -l zImage -append="console=ttySC0"
# kexec -e
```

Many thanks to:

- Eric W. Biederman - Kexec and `kexec-tools` author.
- Simon Horman - `kexec-tools` maintainer.
- Tony Lindgren - Kexec fixes for ARM.
So what is a Kexec-based Bootloader?

A Kexec-based bootloader is a combination of:

- Hardware setup code (not mandatory).
- A Linux kernel configured with `CONFIG_KEXEC=y`.
- User space with `kexec-tools`. 
Motivation

SH-Mobile board - sh7724-based “kfr2r09”

- Handset prototype
- USB Gadget-only hardware design
- Micro-SD slot
- Upstream kernel driver for USB-Gadget
- No U-boot port, no U-boot drivers
Motivation - U-boot vs the Kernel

Question: Add U-boot support or extend the kernel?

U-boot and Barebox:
- Established
- Low barrier of entry
- Small

The Linux Kernel:
- Good hardware support
- Release management
Outline

Bootloader Basics
- Overview & Use Cases
- Inside the Bootloader
- Four Examples

Kexec-based Bootloaders
- Introduction & Motivation
- Hardware Setup Code
- Kernel Configuration & Tuning
- Optimizing User Space

Real World Examples
- SH-Based Boards
- ARM-Based Boards
Hardware Setup Code for SH - Overview

“romImage” for SH:

- Includes all setup code needed to boot the system
- Supported by the boards kfr2r09 and ecovec24
- Use “make romImage” to build image
- Burn to the NOR flash at the reset vector
- Merged in linux-2.6.31
Hardware Setup Code for SH - Files

“romImage” files:
- arch/sh/include/mach-xxx/mach/partner-jet-setup.txt - debug script
- arch/sh/include/mach-xxx/mach/romimage.h - board setup code
- arch/sh/boot/romimage/ - romImage wrapper for zImage
- arch/sh/boot/compressed/ - standard zImage
- arch/sh/configs/ - romImage configurations
Outline

Bootloader Basics
Overview & Use Cases
Inside the Bootloader
Four Examples

Kexec-based Bootloaders
Introduction & Motivation
Hardware Setup Code
Kernel Configuration & Tuning
Optimizing User Space

Real World Examples
SH-Based Boards
ARM-Based Boards
Kernel Configuration & Tuning - Iteration 1 - L-size

Start with an unoptimized kernel:

- Use the defconfig for your board, compile-in drivers
- Compile-in the kernel cmdline using CONFIG_CMDLINE_OVERWRITE=y
- Pass “quiet” on the kernel cmdline to silence the kernel
- Point out user space with CONFIG_INITRAMFS_SOURCE
- Set CONFIG_INITRAMFS_COMPRESSION_NONE=y
- Play around with CONFIG_KERNEL_GZIP/BZIP2/LZMA/LZO
Base on top of Iteration 1 and...

- Tune the kernel configuration for your use case
  - Remove unused subsystems, filesystems and drivers
  - Only one timer driver is needed
- CONFIG_SLOB=y, CONFIG_TINY_RCU=y
- Remember to keep CONFIG_KEXEC=y
Kernel Configuration & Tuning - Iteration 3 - S-size

Base on top of Iteration 2 and...

- Disable module support
- Disable even more kernel features (warning!)
  - CONFIG_EMBEDDED, CONFIG_BUG, CONFIG_PRINTK
Outline

Bootloader Basics
Overview & Use Cases
Inside the Bootloader
Four Examples

Kexec-based Bootloaders
Introduction & Motivation
Hardware Setup Code
Kernel Configuration & Tuning
Optimizing User Space

Real World Examples
SH-Based Boards
ARM-Based Boards
Optimizing User Space - Iteration 1

Start with an unoptimized initramfs:

- Use an existing cross toolchain (perhaps glibc to keep it simple)
- Build static `busybox` binary using `allyesconfig`
- Build a static `kexec` binary from `kexec-tools`
- Hack up configuration files and scripts
- Combine with the kernel using `CONFIG_INITRAMFS_SOURCE`

Uncompressed initramfs size: ~2.6 MiB (~300 Apps, sh4)

With L-size romImage, compressed with LZMA: ~2.9 MiB (~2s)

With L-size romImage, compressed with LZO: ~3.7 MiB (~1.5s)
Optimizing User Space - Iteration 2

Base on top of Iteration 1 and...

▶ Trim the `busybox` configuration to save space
▶ Remember to keep `udhcpc` and `tftp` if you netboot
▶ For speedup, replace `mdev` with kernel option
  `CONFIG.DevTmpfs=y`

Uncompressed initramfs size: ~1.6 MiB (~60 Apps, sh4)
With M-size romImage, compressed with LZMA: ~2.0 MiB (~1.5s)
With M-size romImage, compressed with LZO: ~2.6 MiB (~1s)
Base on top of Iteration 2 and...

- Disable further `busybox` applets
- Hack `busybox` and `kexec-tools` into a single static binary

Uncompressed initramfs size: ~1 MiB (~30 Apps, sh4)
With S-size `romImage`, compressed with LZMA: ~1.0 MiB (~1s)
With S-size `romImage`, compressed with LZO: ~1.4 MiB (~0.5s)
Optimizing User Space

Keep on iterating...

- Switch to a smaller libc
- Remove dead code

For quicker turn around time, use zImage to test user space
Outline

Bootloader Basics
  Overview & Use Cases
  Inside the Bootloader
  Four Examples

Kexec-based Bootloaders
  Introduction & Motivation
  Hardware Setup Code
  Kernel Configuration & Tuning
  Optimizing User Space

Real World Examples
  SH-Based Boards
  ARM-Based Boards
Outline

Bootloader Basics
  Overview & Use Cases
  Inside the Bootloader
  Four Examples

Kexec-based Bootloaders
  Introduction & Motivation
  Hardware Setup Code
  Kernel Configuration & Tuning
  Optimizing User Space

Real World Examples
  SH-Based Boards
  ARM-Based Boards
SH-Based Board MS7724

SH-Mobile board "MS7724" aka "Ecovec24":

- romImage burned to reset vector in NOR Flash
- Loads kernel over Ethernet using CONFIG_SH_ETH=y
- USB Storage support with CONFIG_USB_R8A66597_HCD=y
- LCD splash screen using CONFIG_FB_SH_MOBILE_LCDC=y
SH-Based Board kfr2r09

SH-Mobile board - sh7724-based “kfr2r09”:

- romImage burned to reset vector in NOR Flash
- Loads kernel over USB Gadget using
  `CONFIG_USB_CDC_COMPOSITE=y`
- Loads kernel from Micro-SD using
  `CONFIG_MFD_SH_MOBILE_SDHI=y`
Outline

Bootloader Basics
- Overview & Use Cases
- Inside the Bootloader
- Four Examples

Kexec-based Bootloaders
- Introduction & Motivation
- Hardware Setup Code
- Kernel Configuration & Tuning
- Optimizing User Space

Real World Examples
- SH-Based Boards
- ARM-Based Boards
ARM-Based Boards G3EVM & G4EVM

SH-Mobile ARM Boards G3EVM & G4EVM:

- U-boot burned to reset vector in NOR Flash
- U-boot starts uImage kernel from NOR Flash
- uImage has USB Host using CONFIG_USB_R8A66597_HCD=y
- uImage supports boot over USB Ethernet adapter
Summary

- From power-on to shell prompt in about a second.
- Unoptimized romImage needs ~4 MiB Flash
- Optimized romImage needs ~1 MiB Flash
- Size not an issue with NAND Flash / MMC Boot.