One zImage to Rule them All

Presented at LinuxCon Japan 2012

June 8, 2012

Deepak Saxena – dsaxena@linaro.org

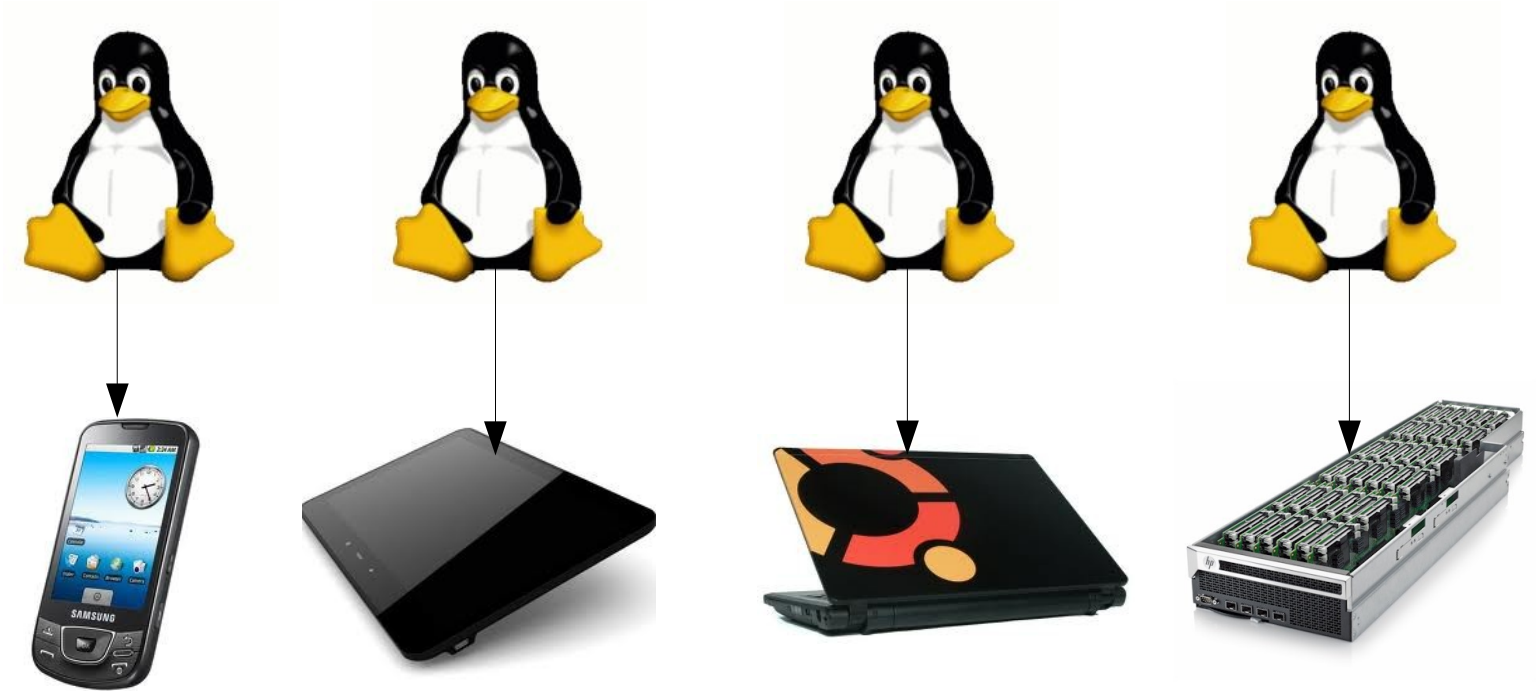
Linaro CONNECT

Who Am I?

- Kernel Tech Lead @ Linaro
  - Mostly meetings, slides, presentations
- Before that, I did write code:
  - 12 years kernel experience
    - Worked on storage drivers at Intel
    - Developed and maintained IXP* Xscale NPU ports
    - Kernel maintainer at MontaVista for several years
      - Reviewed all patches for distro kernel
      - Reviewed thousands of lines of vendor BSP code
  - OLPC kernel maintainer for about 1.5 years

Problem Statement

- Problem:
  - Every ARM platform requires a different kernel
    - Even new revision of the same platform



Why is this a Problem? (1)

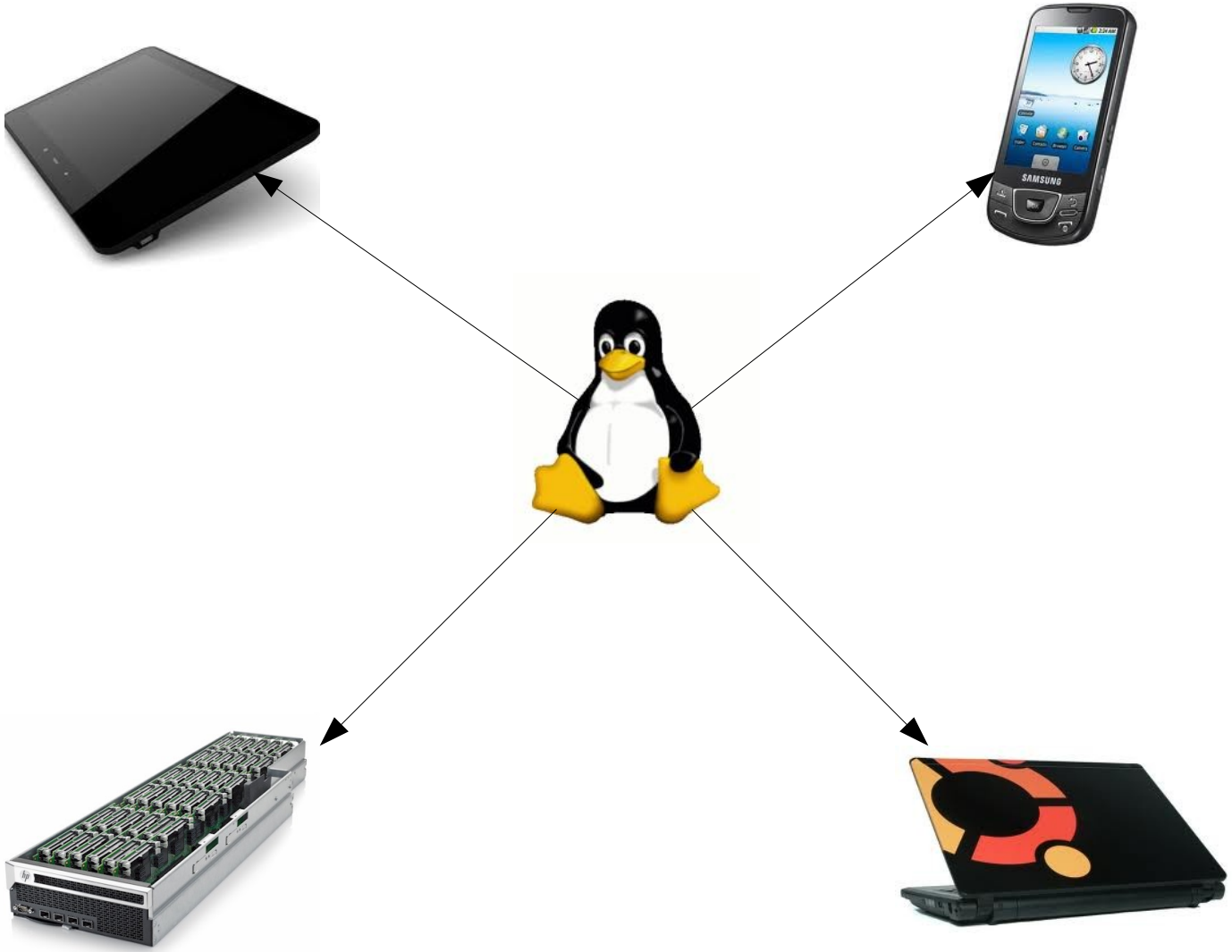
- ARM-Based Servers (“Enterprise”)
  - Vastly different use model from mobile
    - Purchase HW + deploy anywhere
    - HW often purchased separately from SW
    - Want to run new versions of SW on older HW
    - Distros have spoken:
      - Must have one kernel image to make this feasible:
        - Reduce test matrix
        - Provide a run-anywhere install image
    - Cloud/Hyperscale Computing:
      - Servers with thousands of nodes
      - Heterogenous compute environment
        - Will have mix of old and new hardware
        - Need simplified deployment and management model

Why is this a problem? (2)

- Similar problems everywhere:
  - Consumer Electronics
    - Increased test matrix
      - Updating kernel for
  - Mobile:
    - Every new phone rev needs kernel changes
      - Even just trivial changes (re-route IRQS for example)
- “Traditional” Embedded
  -

Solution Statement

- One kernel that boots on any ARM platform!



How did we get here?

- ARM is a very diverse ecosystem
  - Both a great benefit and a challenge
    - Allows for innovation and differentiation
    - Vendors with slightly different IP blocks for same problems
    - Code ends up being duplicated due to closed-door development
  - Maintainer overload
    - Many vendor communities pushing code upstream
    - Too much for Russell King to handle
    - Code start getting pushed directly to Linus
      - (I might have been the first one to do this...sorry for the mess!)
    - Nobody really actively reviewing and driving direction

What's Being Done About This

- Multi-faceted problem
  - Header Cleanups and Consolidation
  - Driver Subsystem Consolidation
  - Device Tree
  - Active Maintenance of ARM-SOC Tree
- Each solves independent set of issues
  - All together lead towards our final goal

The ARM Kernel Tree

- ARM tree layout
  - arch/arm/mach-*
    - Different machine types
  - arch/arm/mach-*/include/mach/*.h
    - Contains machine/SoC specific headers
    - Maps to <mach/*.h> at build time
  - arch/arm/include/asm/*.h
    - Maps to <asm/*.h>

Some Numbers
(From 3.0, when this effort started)

- 64 mach-* directories under arch/arm/
  - Each with separate set of header file
  - Lots of overlapping symbols
    - system.h, io.h, timex.h, hardware.h, vmalloc.h, memory.h, irqs.h, gpio.h, etc
  - Goal is to get rid of many of these
    - Make them ARM generic
    - Move platform specific symbols to non-generic named headers
- 577 occurrences of “#include <mach/*>” in drivers/
  - Again, lots of overlap
  - Creates more maintenance burden
  - Ultimate goal: Move all driver-specific symbols next to drivers
    - arch/arm/mach-foo/include/mach/foo-gpio.h → drivers/gpio/foo-gpio.h (or directly into foo-gpio.c)

Header Cleanups

- system.h
  - Deleted and functions moved to different locations
- io.h
  - Many macros made generic (override with `CONFIG_NEED_MACH_IO_H`)
- vmalloc.h
  - Deleted
- timex.h
  - Work in progress (see next slide)
- irq.s
  - Needed for NR_IRQS
  - Requires moving to sparse IRQ
  - Used by drivers that hardcode this...

Why This Is Difficult
(i.e. <mach/timex.h>: Kernel Archaeology 101)

- timex.h defines `CLOCK_TICK_RATE`
  - Used in *legacy* code to manage system tick timer
    - Value is pretty much meaningless in modern systems
  - Should be easy to delete...
    - Not so much...
    - `CLOCK_TICK_RATE` drives `LATCH`
      - `LATCH` is magic to “latch” into timer trigger register
    - `LATCH` is used in odd places in the kernel
      - Old joystick driver
      - x86 timer code
      - Audio
    - It's required for global macros....

Why This is Difficult (2)

- Macros are somewhat magical, understood by few
- Maintainers don't agree if we can change them

```
/* LATCH is used in the interval timer and ftape setup. */
#define LATCH  ((CLOCK_TICK_RATE + HZ/2) / HZ)  /* For divider */

/* Suppose we want to divide two numbers NOM and DEN: NOM/DEN, then we can
 * improve accuracy by shifting LSH bits, hence calculating:
 *     (NOM << LSH) / DEN
 * This however means trouble for large NOM, because (NOM << LSH) may no
 * longer fit in 32 bits. The following way of calculating this gives us
 * some slack, under the following conditions:
 *   - (NOM / DEN) fits in (32 - LSH) bits.
 *   - (NOM % DEN) fits in (32 - LSH) bits.
 */
#define SH_DIV(NOM,DEN,LSH) (   (((NOM) / (DEN)) << (LSH))              \
                             + ((((NOM) % (DEN)) << (LSH)) + (DEN) / 2) / (DEN))

/* HZ is the requested value. ACTHZ is actual HZ ("<< 8" is for accuracy) */
#define ACTHZ (SH_DIV (CLOCK_TICK_RATE, LATCH, 8))

/* TICK_NSEC is the time between ticks in nsec assuming real ACTHZ */
#define TICK_NSEC (SH_DIV (1000000UL * 1000, ACTHZ, 8))

/* TICK_USEC is the time between ticks in usec assuming fake USER_HZ */
#define TICK_USEC ((1000000UL + USER_HZ/2) / USER_HZ)

/* TICK_USEC_TO_NSEC is the time between ticks in nsec assuming real ACTHZ and  */
/* a value TUSEC for TICK_USEC (can be set bij adjtimex)                */
#define TICK_USEC_TO_NSEC(TUSEC) (SH_DIV (TUSEC * USER_HZ * 1000, ACTHZ, 8))
```

Why This is Difficult (3)

- Changes we are making require coordination
  - Multiple subsystems
  - Undocumented dependencies
  - Multiple maintainers
    - With differing POVs/requirements

Driver Cleanup/Consolidation

- Many different implementations of same code
  - Differing APIs
  - Overlapping symbols
  - Code bloat
  - Some Examples:
    - Pinmux
    - Clock Management

struct clk

- The epitome of code duplication and fragmentation
  - include/linux/clk.h added in 2006 (2.16!!)
    - Declared “struct clk;”
    - Declared various functions that act on it
    - Did not *define* the structure, left to each user
  - 2011: 27 different struct clk definitions in arch/arm!
    - Each with different set of semantics
- Has taken 2+ years to develop a common definition
  - Still needs some discussion on certain areas
  - See documentation/clk.txt

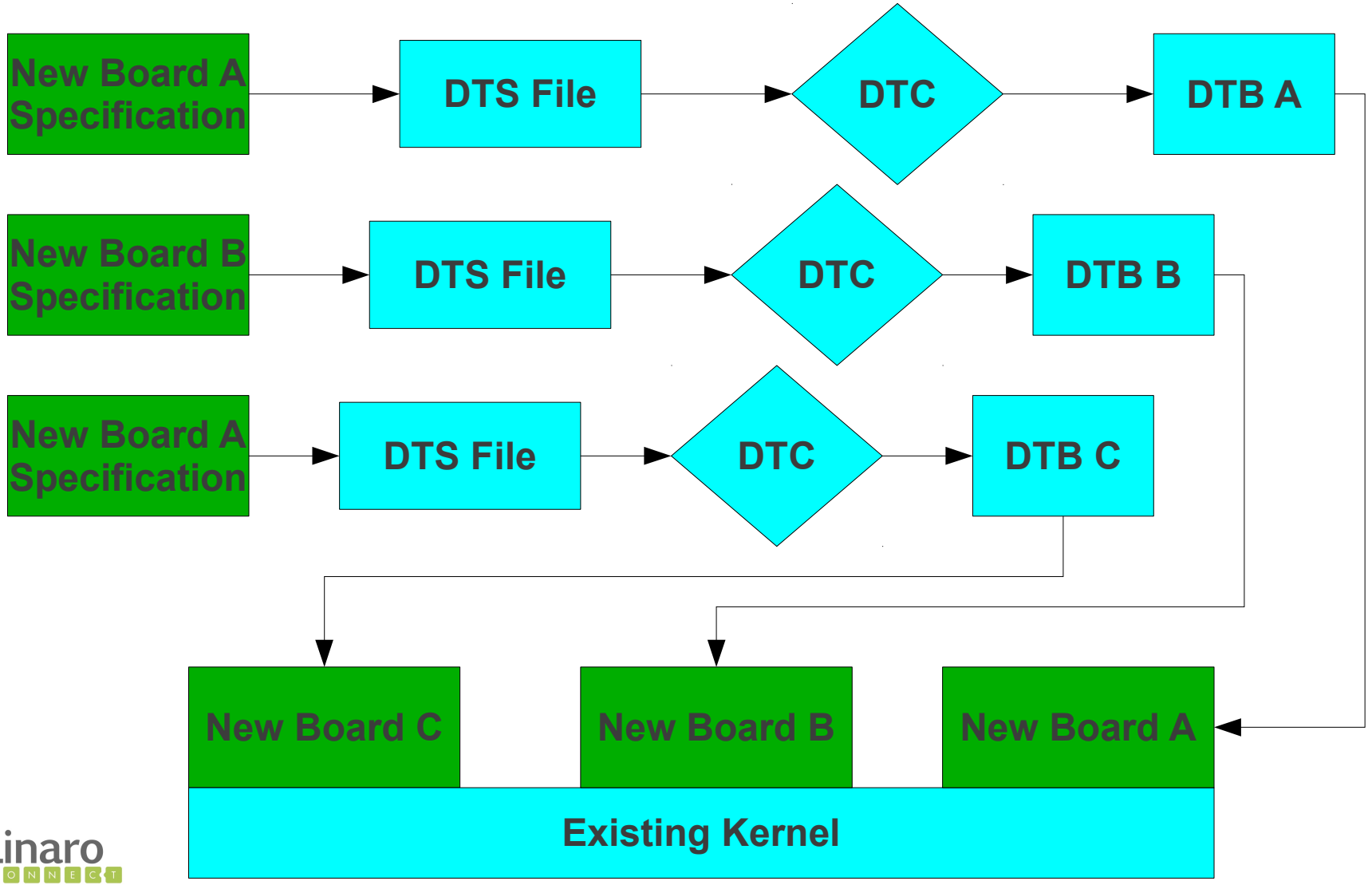
Pinmux

- Subsystem to manage pins on modern SOCs
  - Assign pins to different on-chip IP
  - Run-time re-assignment based on device use
  - Manage pull up and pull down states
  - 6+ months of work to complete this
    - Still discussing some areas...

Device Tree

- Legacy ARM platforms had static device tables
  - Requires rebuild for simple changes:
    - IRQ line change
    - Memory map move
  - Requires new static tables for new board even if no other driver changes
- Device Tree moves much of this data to be probed at boot time instead.

Device Tree Based Workflow



DT Code Demo

- Origin Board, based on Exynos Platform

Where We Are Now

- Work done by many hands
  - ARM SOC Tree has been of great help
    - git://git.kernel.org/pub/scm/linux/kernel/git/arm/arm-soc.git
    - See Arnd's talk (http://goo.gl/CBKaC)
  - Active involvement by Linaro, vendors, and community
- Arnd has 4 linaro platforms building
  - ARM Versatile Express boots!

What's Left To Do

- USB driver consolidation
  - Can currently only build one USB host per type
- Finish DT conversion
  - Lots of drivers still left
  - Lots of dts files to be written
- Real cleanup of driver #include madness
- DEBUG_LL & early_console

A Reality Check

- Several zImages to rule them all...
  - ARM v6/7 bit non-LPAE
  - ARM v6/7 bit LPAE
  - ARM v5
  - ARM v8 (down the road...)
- This is still better than today!

How you Can Help

- Convert your platforms to new common bits:
  - pinmux, sparse irq, generic gpio, device tree, etc
  - Get your code upstream!
    - Life will be much harder otherwise in this new world
  - Help out with one of the areas from previous slide
  - Grab arm-soc tree
  - Rebase your code to this
  - Add your platform to arch/arm/mach-multi and test
    - Send fixes to Arnd and linux-arm-kernel lists

Questions? Comments?