Android/Linux kernel: Lessons learned

Matthew Garrett
<mjg@redhat.com>

What is Linux?

What is Linux?

(You may be in the wrong presentation)

What is Android?

- Mobile operating system
- Targeted at embedded platforms
- Linux kernel
- ARM, x86, mips

Where is Android?

- 13% of the total smartphone market
- 27% of the smartphone market first half 2010
- Spreading into the tablet market

Where is Android

Around 5 million Android devices sold per month

Android and Linux

- Many vendors targeting Android
  - We'd like to get them targeting mainline...

So, what's the problem?

- Android requires certain kernel functionality
- Some of that functionality has to be provided by drivers
- Drivers can end up depending on that functionality...

Wakelocks

An awkward solution to a difficult problem...

A phone without power is unhelpful

- Applications will do their utmost to ensure that your phone looks pretty and dies young
- Letting untrusted applications land on your platform is like giving an SMG to a toddler
  - (except with fewer grubby handprints everywhere)
- If you can control the behaviour of applications, you will have better battery life

(Now for the technical stuff)

Android's power management

- Android will opportunistically enter a full system suspend state
- This forcibly stops processes from running
- A halted process gathers no electrons
- Your phone doesn't melt in your pocket

But...

What if you hit a button at the precise moment that your phone decides to go to sleep?

Wakelocks

Wakelocks exist to prevent you from losing wakeup events

How they work

- User puts phone down
- System decides to suspend
- Incoming phone call wakes system
- Baseband driver takes wakelock, preventing the system from immediately going back to sleep
- Phone UI takes userspace wakelock and acknowledges the baseband, which releases its wakelock
- Phone call ends, wakelock released, back to sleep

Getting them into the kernel

- January 2009 – initial posting
  - Wakelocks and early suspend
- February 2009 – followup
  - Minor cleanups
- February 2009 – third posting
  - Answered most of the implementation questions
- April 2010 – fourth posting
  - Reworked based on discussion with kernel developers

Problems

- Requires modification of any drivers that generate wakeups
- Perceived as working around application bugs
- Doesn't benefit platforms that aren't wakelock aware

What went wrong?

- Lack of understanding of the goals
- Frequent derailing of the discussion
-

The community is hardly blameless...

- People will adopt entrenched positions
- Interests are influenced by corporate design
  - People who've implemented a technical policy and feel it works will tend to be in favour of that policy...
- Lack of interest in a use case can end up as failing to believe in the validity of a use case

Avoiding trouble

- Be clear about what a patchset aims to achieve
- Separate functional code as much as possible
- Getting code into the kernel is always easier if you have a recognised name associated with it
- When people bring up red herrings, make it clear that they're red herrings