What Android and Embedded Linux can learn from each other

Bernhard "Bero" Rosenkränzer
Android Engineer, Linaro

Linaro

DRAFT

Common misconception

Android = Linux + Java

~~Android = Linux + Java~~

Android = Linux + ?

- Kernel patches:
  - Binder IPC + remote method invocation
  - Ashmem/pmem (shared memory system)
  - Logger
  - USB gadget driver
  - Wakelocks
  - OOM handling
  - Alarm timers
  - Paranoid Network Security
  - Timed Output/Timed GPIO
  - RAM_CONSOLE

Android = Linux + ?

- Userland:
  - Bionic libc
  - Graphics system: OpenGL ES, SKIA, SurfaceFlinger, gralloc
  - Debugging tools (adb, ...)
  - Init
  - Package manager (apk)
  - Various libraries (many also commonly found in “normal” Embedded Linux)
  - Build system
  - Dalvik VM

Anything useful for Embedded Linux?

Binder IPC/remote method invocation

- By order of magnitudes faster than D-Bus
- More memory efficient (avoids copies where possible)
- Suitable for passing larger amounts of data (e. g. used by Android's sound system)
- C and C++ interfaces available, no Java required
- Interesting alternative to D-Bus outside the Android world

ashmem/pmem shared memory

- ashmem is similar to POSIX Shared Memory, but with a simpler API
- Can discard shared memory units under memory pressure
- pmem is similar to ashmem, but provides physically contiguous memory: can be shared between userspace and kernel (dsp, gpu, ...)
- C API - no Java required.

Logger

- Write-path optimized logging device – to avoid logging taking too much overhead
- Normal read()/write() API with IOCTLs to change buffer size, flush logs, etc.
- No Java required

USB Gadget driver

- Android's USB gadget driver allows mode switching – allowing the same device to be used as a USB storage device, network device and custom (adb) device without rebooting or switching kernel modules
- Implementation is not very clean (which is why this likely won't make it into Linux soon), but works well.
- Obviously, no Java required

Wakelocks

- Locks that keep the machine from suspending until they're released
- Simple C and sysfs based APIs available, no Java required
- It's easy to "forget" about a wakelock and prevent the machine from ever suspending. There may be better options.
- Still useful if used right.

OOM Handling

- Android's optional OOM (Out-of-memory) handler simply kills processes as available memory becomes low.
- Suitable for some types of device – what you want worst for other types of device
- Whether or not this behavior is wanted depends on a device's purpose – not so much on whether it's running Android or traditional Linux

Alarm Timers

- Provides a simple interface (character device based – no Java required) for running a timer – even if the device is otherwise suspended.
- POSIX Alarm Timers – providing similar functionality with a different API – have made it into Linux 3.0, so Android Alarm Timers will likely lose significance. Use POSIX Alarm Timers if they get the job done.

Paranoid Network Security

- Allows restricting network functionalities by the group of the calling process:
  - Can create RFCOMM, SCO or L2CAPP Bluetooth socket
  - Can create a Bluetooth socket
  - Can create IPv4 and IPv6 sockets
  - Can create raw sockets
  - Allow CAP_NET_ADMIN permissions

Timed Output/Timed GPIO

- Allows charging a GPIO pin and restoring it automatically after a specified timeout. Can be useful for e. g. controlling a vibrator from user space

RAM_CONSOLE

- Allows saving the kernel's printk() messages to a buffer in RAM that can be viewed in the next kernel invocation – making it a lot easier to work with kernel panics

Bionic libc

- Bionic is “yet another libc implementation”, alongside (e)glibc, uClibc, dietlibc, newlib, ...
- Small (even smaller than uClibc), designed to work well with low-powered CPUs
- Good support for Linux specific features
- BSD licensed
- Some room for optimizations left

Bionic libc, part 2

- Missing some standard libc functions, but depending on the rest of your stack, you may not need them – e. g. Bionic is missing white character support, but if you're using e. g. Qt (QString, QChar) or ICU for your text processing needs, you won't notice
- No binary compatibility with glibc – BUT binary compatibility with Android...

Graphics system

- Android’s graphics system and its main components OpenGL ES, Skia, gralloc, SurfaceFlinger can be an interesting alternative to X11 and Wayland even on “normal” Linux:
  - Less overhead
  - Designed for relatively modern GPUs
  - Supported fairly well by hardware makers
  - Accessible from C, C++, Java

Graphics system

- A Qt port to Android exists (Necessitas) – including an output plugin for the Android graphics system with hardware accelerated OpenGL ES support.
  Using the Android graphics system for traditional Linux may be more feasible than most people are thinking.

Debugging tools

- “Normal” Linux has, for the most part, better, more mature debugging tools for most purposes, however, one thing deserves attention:
  - “debuggerd” provides a nice way to log backtraces automatically when an application crashes. This could be useful functionality for “normal” Linux.

Init

- Many Linux distributions are currently trying to replace good old SysVInit with something better – Android is no exception. Its init replacement takes a flexible but quick to parse init.rc file describing startup.
- Due to lack of support for per-package init scripts not suitable for Desktop Linux without extra work, but Android's init could be interesting to some Embedded Linux devices.

Let’s look at the other side: what can Android learn from Embedded Linux?

There is a world outside of Android

- And often, there's more than one way to do things right. Android makes it very hard to integrate things not specifically written for it, and should make a better effort at opening up to:
  - Applications using a pre-existing build system (cmake, autoconf, ...) - see also Botao's talk about build system integration!
  - Programming languages other than Java

Developer tools

- The first Android based notebooks have come out – and (short of “build it yourself”), there’s no compiler that can compile an application on the notebook to run on itself.
- The fact that Android decided against having an equivalent of /usr/include makes things worse – you have to pick the headers from the Android source tree or SDK.

Developer tools

- It doesn’t make too much sense to have development tools on your typical mobile phone, but it was foreseeable (and has happened now) that Android would move into bigger devices with proper keyboards etc.
- Numerous Linux developer tools would be nice to have (shameless plug: Linaro’s Android builds include busybox, strace and more!)

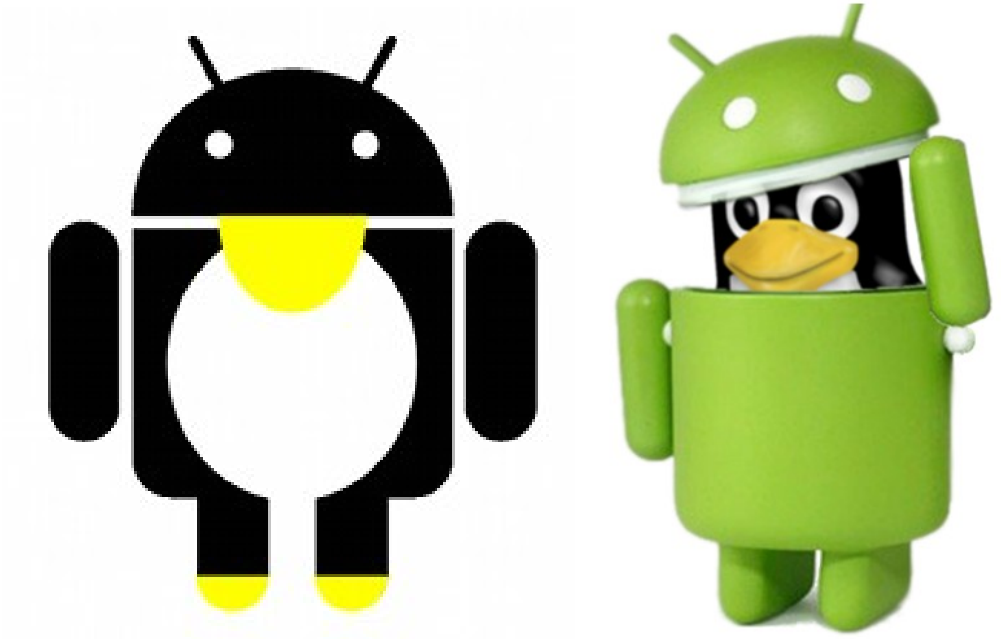
Graphics system

- This works both ways – while there are reasons for a Linux developer to consider using Android's Graphics system, it may also make sense to have a mostly-Android system using X11 if, for example, network transparency is a major feature for the device being built, or if you're stuck with a graphics chipset for which only an X11 driver exists
- Having the whole stack (Dalvik etc.) running well on top of X11 would be interesting for running Android applications on Desktop Linux as well...

Package management

- Android's (and BSD's) approach of keeping the entire source in one tree and building it all in one go can become very slow as the system grows
- Having binary packages (akin to rpm/deb) makes it easier to pull in another application quickly without having to rebuild
- Also practically forces the concept of development packages (badly needed on Android)

In an ideal world...



Linaro

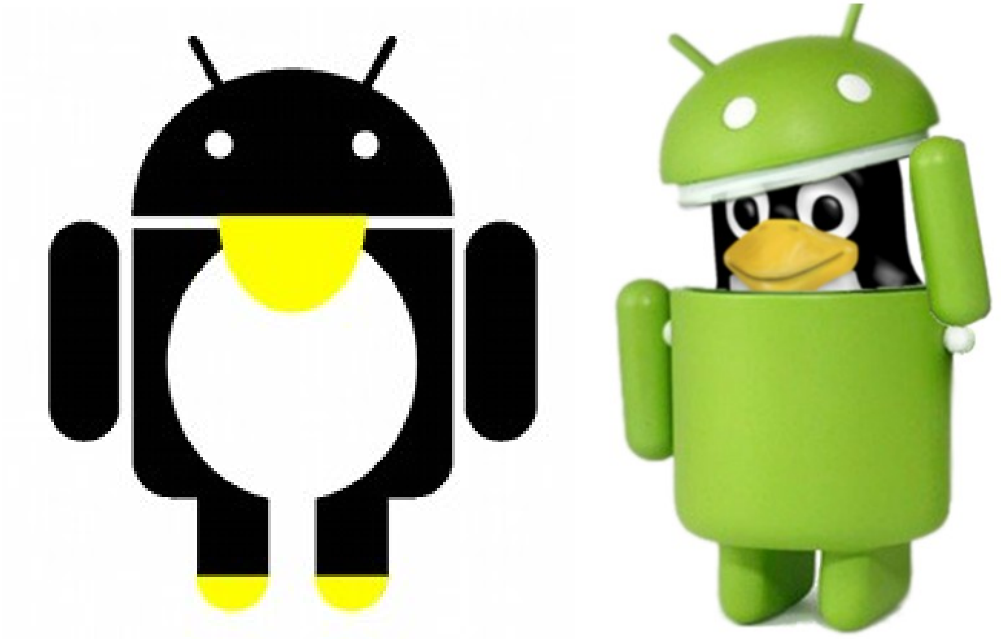
In an ideal world...

- There's not much of a point in having two separate communities working separately.
- There's no immediate need to be binary compatible (keep the different libcs for different purposes – and we won't have binary compatibility between arm and x86 anyway) – but let's aim at maximum source level compatibility

In an ideal world...

- Let's build the best of both worlds – make good Android technologies available (and actually use them) on regular Linux without forcing its more questionable "features" on anyone.

Questions?



Linaro