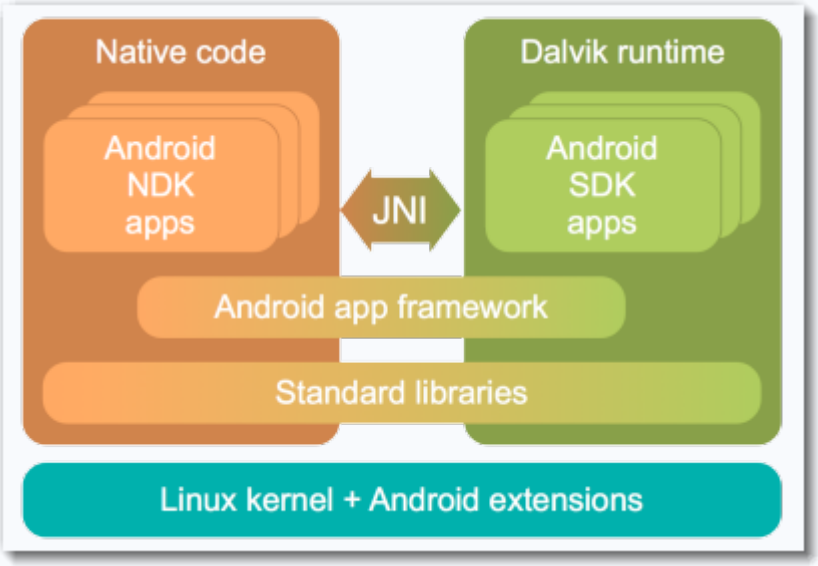
谷歌工程师多图详解Android系统架构

近日，Google 的一位工程师 Sans Serif 发布了一篇博文非常清楚地描述了 Android 系统架构，中国移动通信研究院院长黄晓庆在新浪微博上推荐了该文，并认为文中对 Android 的介绍很好，您可以看一下 Google 工程师眼中的 Android 系统架构是什么样的。以下为 Sans Serif 博文的译文。

Andriod 是什么？

首先，就像 Android 开源和兼容性技术负责人 Dan Morrill 在 Android 开发手册兼容性部分所解释的，“Android 并不是传统的 Linux 风格的一个规范或分发版本，也不是一系列可重用的组件集成，Android 是一个用于连接设备的软件块。”



Linux：

所有东西的底层是一个稳定的保持更新的 Linux 内核（我现在用的 Nexus 手机所用的就是 2.6.32 版的内核），以及我们精心打造的能源管理组件；当然还有将它们整合至上层 Linux 代码的扩展和公共组件。

Dalvik：

Android 另一个重要的部分，包括虚拟机和一组重要的运行环境。它的设计非常巧妙，是个很好的一个手机终端的底层应用。

代码如何生成？

Dalvik 虚拟机只执行.dex 的可执行文件。当 Java 程序通过编译，最后还需要通过 SDK 中的工具转化成.dex 格式才能在虚拟机上执行。

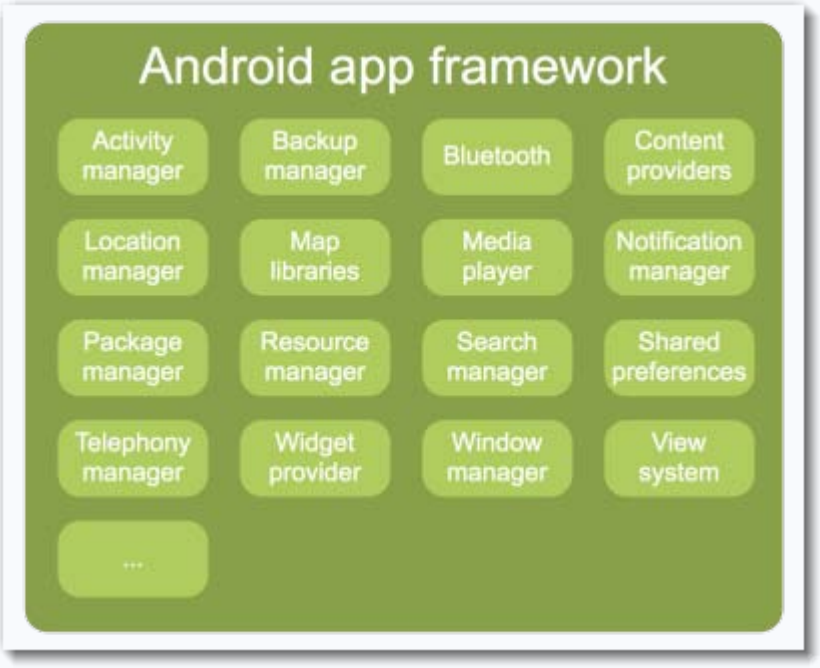
我需要强调的是，Android 应用本身就可视作可在平台上运行并调用 APIs 的代码，所以对代码如何生成不需特别看重。

特别的 Apps：

在图中有些基于 Dalvik 虚拟机的 Apps 看起来像是 Android 的一部分，其实是由 Google 提供，这些应用包括 Dialer、Contact、Calendar、Gmail 和 Chat 等。它们中的绝大部分是开源并可复用的。只有少部分例外，比如 Google Maps 和 Android Market.

开源那些事：

在下面的图中，绿色的大部分组件是基于 Apache 许可证开源，其余基于 GPL、LGPL 和 BSD。



Android 框架

在 Android 开发者网（developer.android.com）上已有不少篇幅来帮助你使用它，在此就不再累述。

标准库

在这里“标准”是指“开发者在开源环境中一般可以使用的”。

App 里面是什么

一个 Android App 包含在一个我们称之为 APK 的压缩文件夹中,APK 并没有什么可说的,需要注意的是 Android Manifest——介于 App 和 Android System 的接口。

App 里面是什么(1)

App 里面是什么(2)

其他

大多数应用是基于 Dalvik 的，我指的是除了游戏之外的应用。游戏开发者通常希望用 C/C++来编写，排斥使用虚拟机，所以他们可以通过 Andriod NDK 来开发。

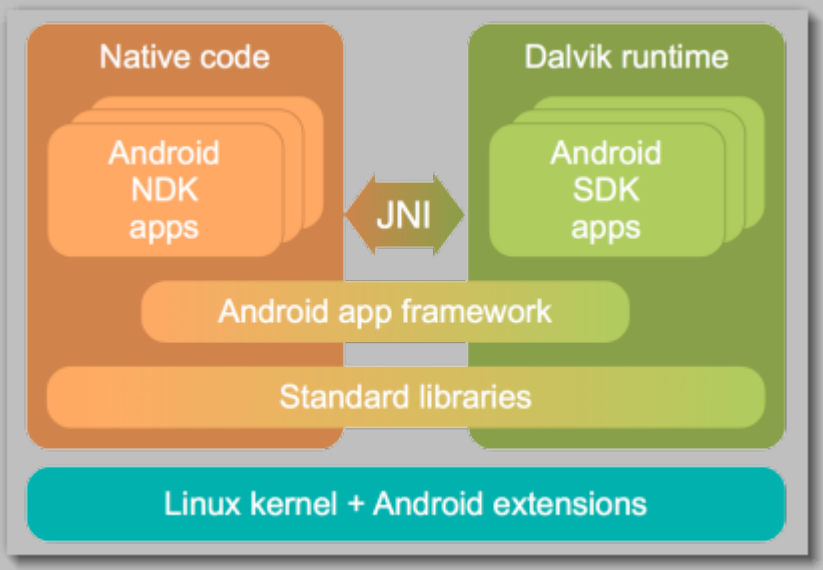
What Android Is

Being an illustrated run through the basics.

What happened was, for our recent South American tour I wanted an Android architecture overview graphic. I ran across, among the Android SDK documentation, a page entitled What is Android?, and it's perfectly OK. Except for, I really disliked the picture — on purely aesthetic grounds, just not my kind of lettering and gradients and layouts — so I decided to make another one.

I thought I'd run it here and, since I've been spending a lot of time recently explaining What Android Is to people, I thought I'd provide my version of that as well, in narrative rather than point form.

First of all, as Dan Morrill memorably explained in On Android Compatibility, "Android is not a specification, or a distribution in the traditional Linux sense. It's not a collection of replaceable components. Android is a chunk of software that you port to a device."



Linux • Underneath everything is a reasonably up-to-date Linux kernel (2.6.32 in my current Nexus One running Froyo), with some power-saving extensions we cooked up; the process of trying to merge this stuff into upstream Linux has been extended and public and is by no means over.

Android runs on Linux, but I'd be nervous about calling it a distro because it leaves out so much that people expect in one of those: libraries

and shells and editors and GUIs and programming frameworks. It's a pretty naked kernel, which becomes obvious the first time you find yourself using a shell on an Android device.

If it were a distro it'd be one of the higher-volume ones, shipping at 200K units a day in late 2010. But nobody counts these things, and then there are a ton of embedded flavors of Linux shipping in unremarkable pieces of consumer electronics, so there's a refreshing absence of anyone claiming to be "the most popular Linux". I like that.

Dalvik • The next big piece is Dalvik, comprising the VM and a whole bunch of basic runtime essentials. Its design is fairly unique, and judging by recent history, seems to be working out pretty well as a mobile-device app substrate.

All the standard APIs that you use to create Android apps are defined in terms of Dalvik classes and interfaces and objects and methods. In fact, some of them are thin layers of Dalvik code over native implementations.

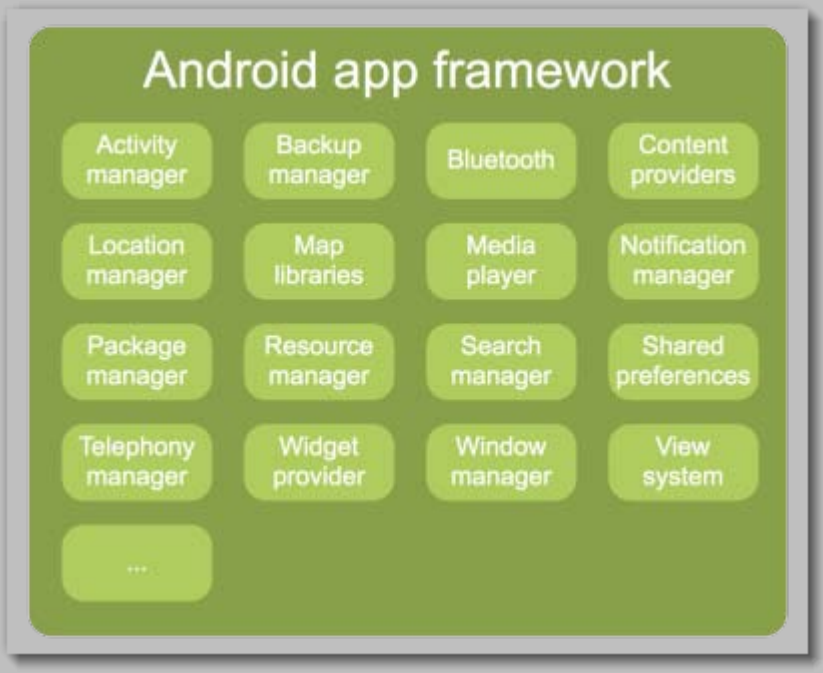
It's possible, and common practice, to call back and forth between Dalvik and native code using the JNI protocol, which is a neat trick since what's running on Dalvik isn't anything like Java bytecodes on a Java VM.

How It's Generated • Native code is currently produced more or less exclusively by compiling C or C++ code; but there's no reason it has to be that way. Dalvik code is currently produced by generating Java bytecodes and translating them; but there's no reason it has to be that way.

I want to emphasize this point a little. Android apps are defined as code that runs on the platform and uses the APIs. As long as an app does these things properly, it's really nobody's concern how it got generated.

Special Apps • The picture is a little misleading, because some of those Dalvik-based apps are provided by Google and sometimes are seen as "part of Android". I'm talking about the Dialer and Contacts and Calendar and Gmail and Chat and so on. Most of them are open-source and replaceable (and have been replaced by handset makers); a few are closed-source and proprietary, like Google Maps and Android Market.

That Open-Source Thing • In the big picture above, most of the stuff in green is Apache-licensed. The rest is a mixture of GPL and LGPL and BSD, with some Apache in there too. This excludes some low-level device drivers and of course the majority of non-Google apps, which are closed-source.



The Framework • This is the stuff that uniquely defines Android; more or less everything that Google wrote and you wouldn't expect to find on a reasonably-configured GNU/Linux box. Its proper use is the subject of all the many pages on display at developer.android.com and of endless mailing lists, sample sites, and a growing number of books.

I like it; but you already knew that.

Libraries • The word "standard" here means "generally available to programmers working in an open-source environment". The picture isn't comprehensive.

Quite a few people, including me, have over-emphasized the role of the Harmony libraries. To start with, the Android selection excludes lots of stuff, for example AWT and Swing and OMG CORBA; all superfluous for apps using the Android framework.

Also, just counting roughly by code bulk of all the sort of stuff in this picture, the Harmony code comprises less than half the total. I don't want to diss Harmony, they're a wonderful project and I'm a huge fan; but it's inaccurate to give the impression that Android is just Dalvik plus Harmony.

What's In an App? • An Android app lives in what's called an APK, which is simply a ZIP file, with a particular internal file layout that allows it to be run in place, without unpacking. There's nothing magic about

them, you can email them around and drop them on USB keys and extract pieces with unzip.

The Android Manifest is the interface between an app and the Android system, and that's all I'm going to say here because it's a key piece of the puzzle and deserves lengthy discussion if any at all.

The resource bundle contains your audio and video and graphics and so on, the pieces that come with the app as opposed to being fetched over the network.

Native or Not • Most apps these days are written for Dalvik. When I say "most apps" I mean "everything that isn't a game"; game developers typically want to code in C/C++ and that's it. Dalvik offers a nice fast gateway to OpenGL and all the phone's hardware, but game devs just don't want to hear about virtual machines, so they use the Android NDK.

If you're writing code in the Java programming language you can use Eclipse and a pretty nice toolchain that makes the barrier to entry remarkably low. If you're coding to the NDK, you're going to be doing a lot of the build-time machinery yourself and living without some of the nice debugging and profiling candy, not to mention signing up to port your code to other CPU architectures if they run Android and have lots of users. But game developers revel in pain.

And that's what Android is • Hope you liked the pictures.