Get Started with Android

Colin Walls, Mentor Graphics

The Android operating system simplifies application development.

There’s no question Android’s popularity has made it the star software of the mobile handset industry and it has become the preferred operating system for many developers. So what exactly is Android and should programmers know before they start a project?

You might think of Android only as an operating system (OS) for high-end mobile phones, but it extends beyond that type of device. Android provides an open-source operating system developed by Google for the Open Handset Alliance (OHA). So this complete and highly optimized software platform has the potential for use in a wide range of applications. Android operates as an application framework atop Linux, which has facilitated its rapid adoption by many product designers. In most cases, open-source licenses require that developers share their modifications and additions with the over-arching programmer community. But an Apache license (ver. 2) applies to the majority of Android code, which means developers can add proprietary code to the operating system without having to share it.

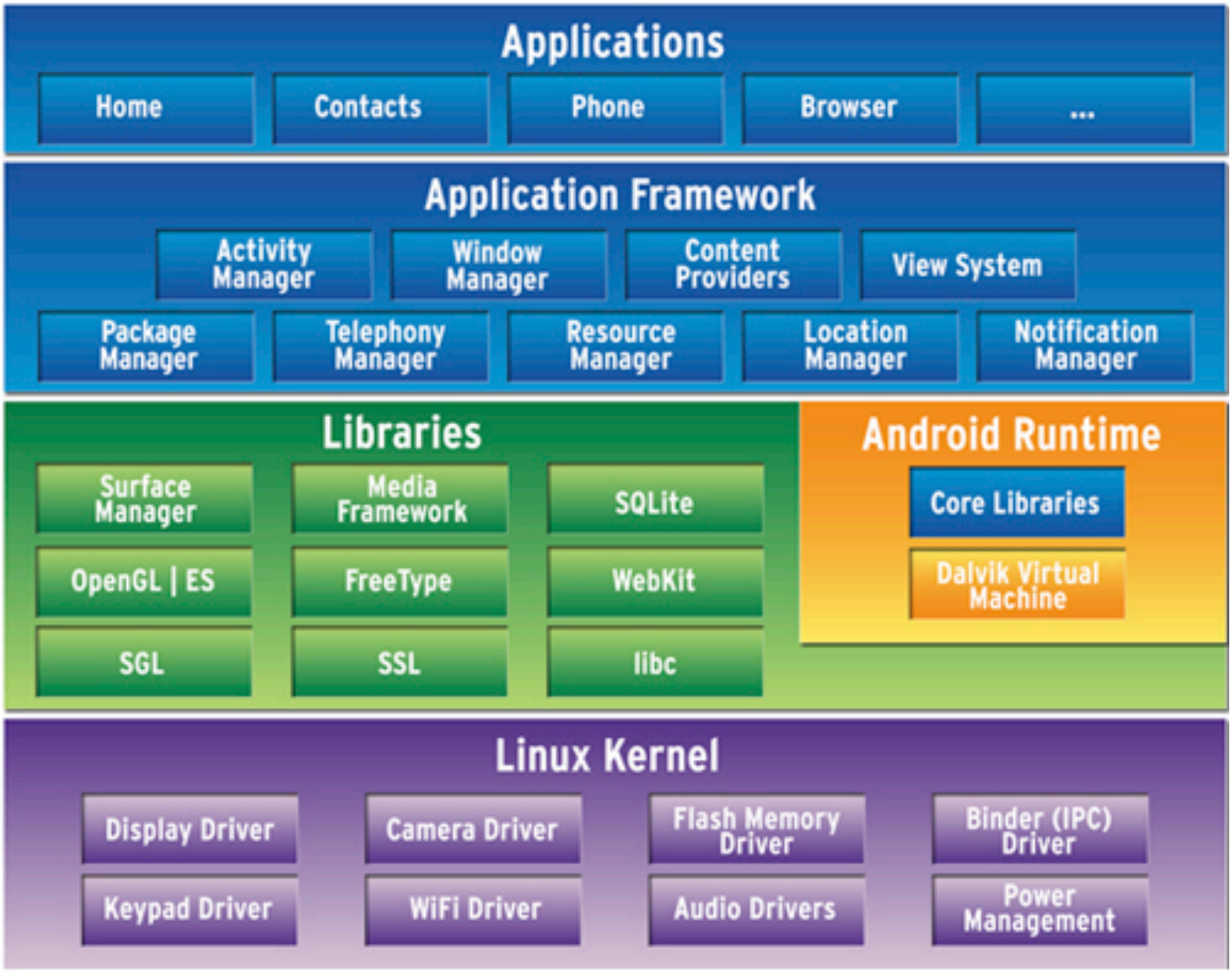

Fig 1. The Android system architecture uses a standard software-stack approach with Linux as its base.

An Android system acts like a stack of software components built upon the Linux 2.6 kernel. Linux provides basic system functions such as process and memory management, multithreading, and security. Also, the Linux kernel includes device drivers that take the pain out of interfacing to peripheral hardware.

The Android runtime kernel operates above the Linux kernel, as shown in Figure 1, and it contains both runtime libraries and the Dalvik virtual machine (VM). The runtime libraries comprise open-source building blocks such as the WebKit browser, the SQLite database, and the FreeType font engine.

The Dalvik VM--designed specifically for Android--provides two key capabilities: First, programmers can instantiate it, or create it, as needed so each application has its own private copy running in a Linux process, which protects its code and data from other applications.

Second, the Dalvik VM uses registers rather than a memory-based stack to improve memory-use efficiency. The Dalvik VM implements bytecode, designed for efficient execution. It’s important to realize that Android is not a Java virtual machine, but it does use the Java language. Java classes within Android's Application Framework

layer provide higher level services for applications.

The top applications layer comprises standard applications, such as a Web browser, email, and so on, within Android. Each application may let other applications use some of its functions. The Android short-message service (SMS), for example, can let other applications send text messages. This form of function "exposure" lets programmers reuse software and helps provides a consistent "user experience." Applications also can transfer, exchange, and share data.

Although programmers have other options, they usually create applications in Java. Not only does the Dalvik VM provide an efficient bytecode interpreter, it also lets programmers create portable code. Programmers who have C/C++ applications can use the Android Native Development Kit (NDK). A Java output file is processed, using a Google-supplied tool, to generate specific bytecode for the Dalvik VM.

An Android application includes resources bundled into an archive called an "Android package." And an application includes four components: activities, services, broadcast receivers, and content providers, which the application instantiates and runs as required.

1. The primary code component of an Android app is an *Activity*, which is simply some executable code which has a user interface. The developer may incorporate a number of activities into the app, but must nominate one of them as the default, which runs when the user invokes the app. Each activity can invoke other activities and services.

2. A *Service* operates like an activity, except it runs in the background without a user interface; for example, a media player that plays music while a user performs other tasks. Services often remain active for long periods.

3. *Broadcast Receivers* simply respond to broadcast messages from other applications or from the system. When an Android-based product takes a picture, for example, other applications might need to know this event occurred.

4. A *Content Provider* transfers data from one application to other applications on request. (Methods within the ContentResolver class handle data-transfer requests.)

The standard Android development environment runs within Eclipse and by specifying an Android Virtual Device you define your target configuration. You can then execute code on the host-based emulator or on a real device, normally connected via a USB cable. This environment only supports Android development for ARM-based target devices, but recently Mentor Graphics and others have ported Android to processor architectures such as MIPS.

About the author

Colin Walls has over twenty-five years experience in the electronics industry, largely dedicated to embedded software. A frequent presenter at conferences and seminars and author of numerous technical articles and two books on embedded

software. Colin is an embedded software technologist with the Mentor Graphics Embedded Software Division, based in the UK. Visit his blog is located at: http://blogs.mentor.com/colinwalls [1].

For further reading about Android Developers please visit, http://developer.android.com/index.html [2].

Lee, Wei Meng, "Beginning Android Application Development," Wrox. ISBN: 978-1118017111.

Rogers, Rick, et al., "Android Application Development: Programming with the Google SDK," O'Reilly Media. ISBN: 978-0596521479

Source URL (retrieved on *09/01/2014 - 7:47pm*):
http://www.ecnmag.com/articles/2011/07/get-started-android

Links:
[1] http://blogs.mentor.com/colinwalls
[2] http://developer.android.com/index.html