2015

Android Software Stack & Build Process

V1.0

LEE KELLY

Table of Contents

1. The Android Operating System

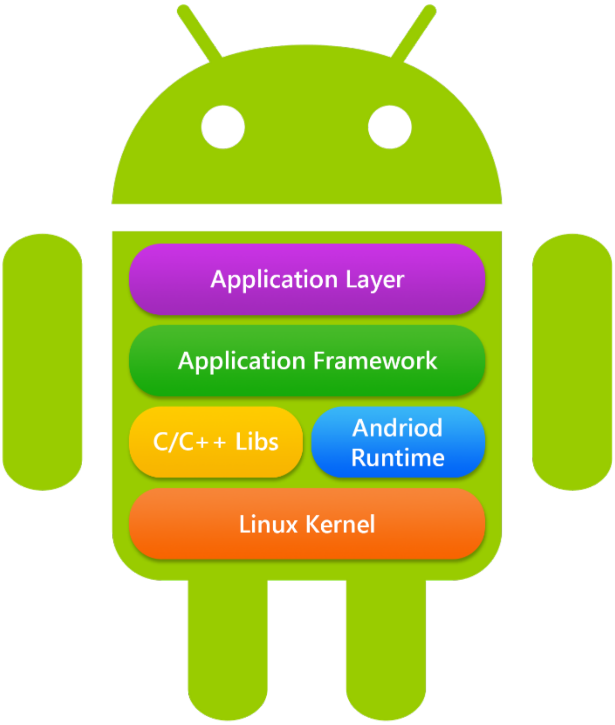
Android is an operating system based on the Linux kernel. The project responsible for developing the Android system is called the *Android Open Source Project* (AOSP) and is primarily lead by Google.

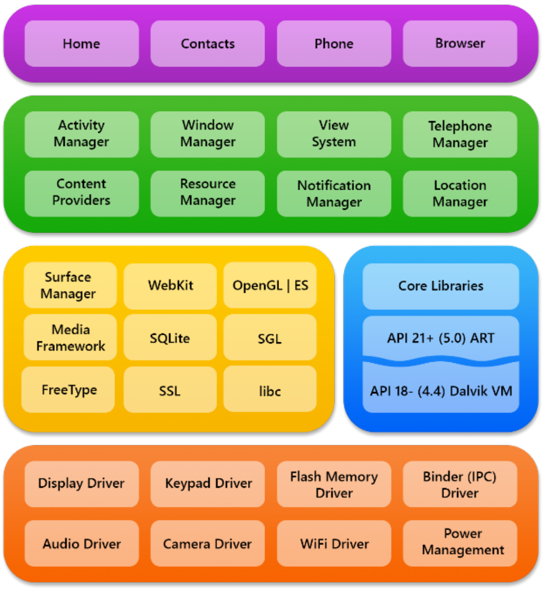
The Android system supports background processing, provides a rich user interface library, supports 2-D and 3-D graphics using the OpenGL-ES standard and grants access to the file system as well as an embedded SQLite database.

An Android application typically consists of different visual and non-visual components and can reuse components of other applications.

2. The Android Software Stack

The Android system is a full software stack, which is typically divided into four layers:





2.1 Linux Kernel Layer

At the base of the Android stack is the Linux kernel, the communication layer for the underlying hardware, it provides the following functions:

- Hardware Abstraction
- Memory Management Programs
- Security Settings
- Power Management Software
- Other Hardware Drivers (*Drivers are programs that control hardware devices.*)
- Support for Shared Libraries
- Network Stack

2.2 Libraries & Runtime Layer

Above the Linux kernel are native libraries for many common functions (*playback/recording of audio/video, graphic rendering, data storage, web browsing, etc.…*) of the Application Framework, as well as the Android Runtime and core Java libraries.

2.2.1 Libraries

- **Surface Manager** – composing windows on the screen
- **SGL** – 2D Graphics
- **Open GL|ES** – 3D Library
- **Media Framework** – Supports playback/recording of various audio/video formats.
- **FreeType** – Font Rendering
- **WebKit** – Browser Engine
- **libc** – System C libraries
- **SQLite** – Database Engine
- **OpenSSL** – SSL Network Library

2.2.2 Android Runtime

Core Java libraries and an application runtime environment for running Android applications.

Prior to Android 5.0 *Lollipop* (API 21), the runtime used was the Dalvik Virtual Machine that transformed the application's bytecode into native instructions using a *Just-In-Time* (JIT) compiler, which compiled an app every time it was executed.

Android RunTime (ART) has replaced Dalvik, and uses *Ahead-Of-Time* (AOT) compilation, which compiles the app upon install. It was written from the ground up to take advantage of modern smartphone hardware and provide smoother animations and longer battery life.

2.3 Application Framework Layer

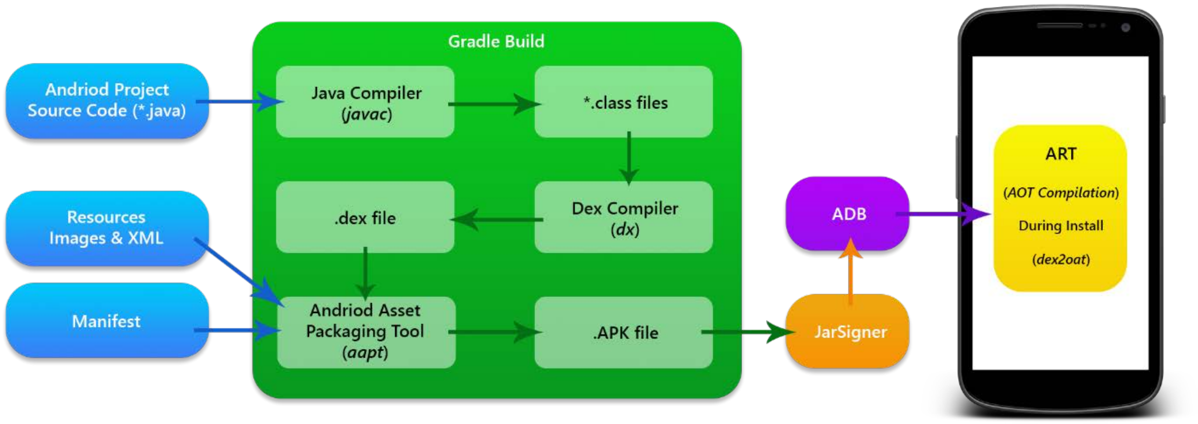
On the top of native libraries and android runtime, is the Android framework, which includes Android API's such as UI (*User Interface*), telephony, resources, locations, Content Providers (*data*) and package managers. It provides a lot of classes and interfaces for Android application development.

2.4 Application Layer

On the top of the Android framework, there are applications. All applications (*i.e. home, contact, settings, games, browsers, etc.…*) are using Android framework, which uses Android runtime and libraries. Android runtime and native libraries are using the Linux kernel.

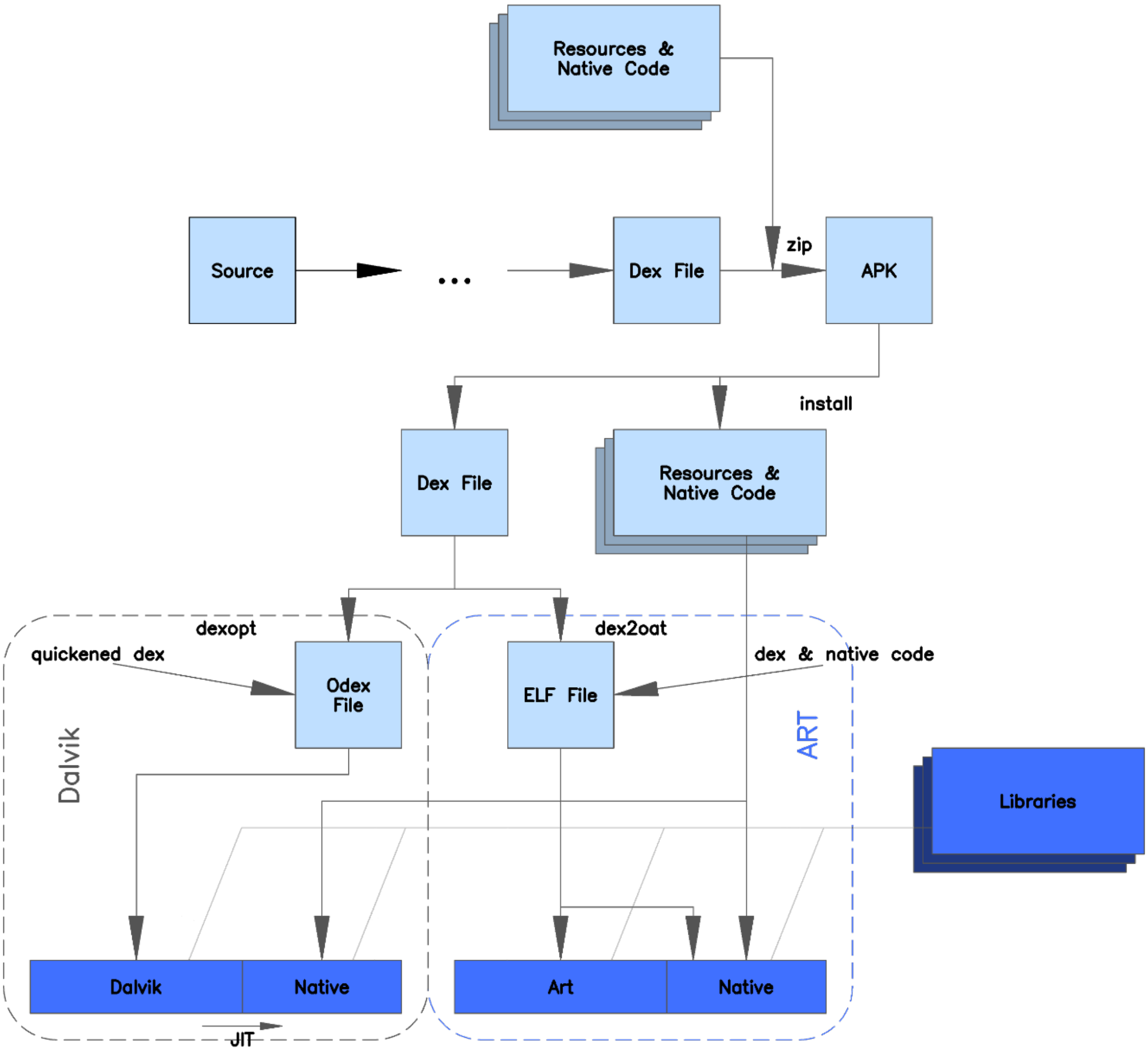
3. The Build Process

When deploying an app from Android Studio to a device (*physical or virtual*) the following compiling and packaging process occurs:



1. The Java source files are converted to Java class files by the ***javac*** (*Java Compiler*).
2. The Android SDK contains a tool called ***dx*** which converts Java class files into a *.dex* (*Dalvik Executable*) file. All class files of the application are placed in this *.dex* file. During this conversion process redundant information in the class files are optimized in the *.dex* file, therefore much smaller in size than the corresponding class files.
3. The *.dex* file, Resources of an Android project (*images and XML files*) and the Manifest, are packed into an *.apk* (*Android Package*) file, by the ***aapt*** (*Android Asset Packaging Tool*).
4. The *.apk* file is then signed by JarSigner.
5. The signed *.apk* file can then be deployed to a device via the ***adb*** (*Android Debug Bridge*) tool.
6. During the install process, **ART** uses the ***dex2oat*** tool to compile the *.dex* file into an Executable and Linkable Format (ELF file), and into native machine code.

4. A Comparison of Dalvik and ART Architectures


Resources & Native Code
Source
...
Dex File
zip
APK
install
Dex File
Resources & Native Code
dexopt
quickened dex
Odex File
Dalvik
dex2oat
dex & native code
ELF File
ART
Libraries
Dalvik
Native
Art
Native
JIT