引言

通过前面两篇：

- Android 开发之旅：环境搭建及HelloWorld
- Android 开发之旅：HelloWorld项目的目录结构

我们对 android 有了个大致的了解，知道如何搭建 android 的环境及简单地写一个 HelloWorld 程序，而且知道一个 android 项目包括哪些文件夹和文件及相应的作用。本篇将站在顶级的高度——架构，来看 android。我开篇就说了，这个系列适合 0 基础的人且我也是从 0 开始按照这个步骤来学的，谈架构是不是有点螳臂挡车，自不量力呢？我觉得其实不然，如果一开始就对整个 android 的架构了然于胸，就不会误入歧途，能够很好地把握全局。本文的主题如下：

1、架构图直观

下面这张图展示了 Android 系统的主要组成部分：

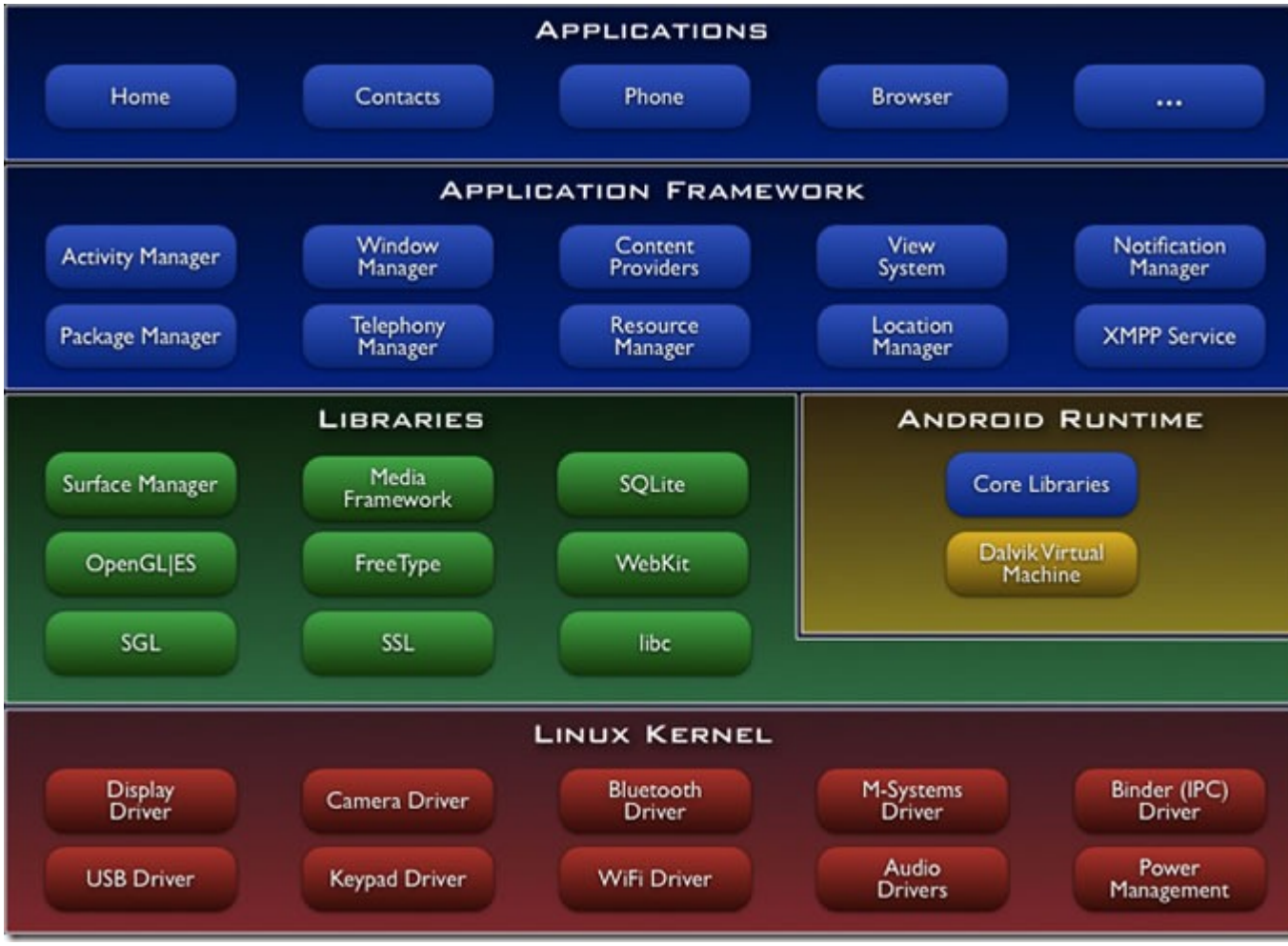

图 1、Android 系统架构（来源于：android sdk）

可以很明显看出，Android 系统架构由 5 部分组成，分别是：Linux Kernel、Android Runtime、Libraries、Application Framework、Applications。第二部分将详细介绍这 5 个部分。

2、架构详解

现在我们拿起手术刀来剖析各个部分。其实这部分 SDK 文档已经帮我们做得很好了，我们要做的就是拿来主义，然后再加上自己理解。下面自底向上分析各层。

2.1、Linux Kernel

Android 基于 Linux 2.6 提供核心系统服务，例如：安全、内存管理、进程管理、网络堆栈、驱动模型。Linux Kernel 也作为硬件和软件之间的抽象层，它隐藏具体硬件细节而为上层提供统一的服务。

如果你学过计算机网络知道 OSI/RM，就会知道分层的好处就是使用下层提供的服务而为上层提供统一的服务，屏蔽本层及以下层的差异，当本层及以下层发生了变化不会影响到上层。也就是说**各层各司其职，各层提供固定的 SAP（Service Access Point）**，专业点可以说是**高内聚、低耦合。**

如果你只是做应用开发，就不需要深入了解 Linux Kernel 层。

2.2、Android Runtime

Android 包含一个核心库的集合，提供大部分在 Java 编程语言核心类库中可用的功能。每一个 Android 应用程序是 Dalvik 虚拟机中的实例，运行在他们自己的进程中。Dalvik 虚拟机设计成，在一个设备可以高效地运行多个虚拟机。Dalvik 虚拟机可执行文件格式是.dex，dex 格式是专为 Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。

大多数虚拟机包括 JVM 都是基于栈的，而 Dalvik 虚拟机则是基于寄存器的。两种架构各有优劣，一般而言，基于栈的机器需要更多指令，而基于寄存器的机器指令更大。dx 是一套工具，可以将 Java .class 转换成 .dex 格式。一个 dex 文件通常会有多个.class。由于 dex 有時必须进行最佳化，会使文件大小增加 1-4 倍，以 ODEX 结尾。

Dalvik 虚拟机依赖于 Linux 内核提供基本功能，如线程和底层内存管理。

2.3、Libraries

Android 包含一个 C/C++库的集合，供 Android 系统的各个组件使用。这些功能通过 Android 的应用程序框架（application framework）暴露给开发者。下面列出一些核心库：

- **系统 C 库**——标准 C 系统库（libc）的 BSD 衍生，调整为基于嵌入式 Linux 设备
- **媒体库**——基于 PacketVideo 的 OpenCORE。这些库支持播放和录制许多流行的音频和视频格式，以及静态图像文件，包括 MPEG4、 H.264、 MP3、 AAC、 AMR、JPG、 PNG
- **界面管理**——管理访问显示子系统和无缝组合多个应用程序的二维和三维图形层
- **LibWebCore**——新式的 Web 浏览器引擎,驱动 Android 浏览器和内嵌的 web 视图
- **SGL**——基本的 2D 图形引擎
- **3D 库**——基于 OpenGL ES 1.0 APIs 的实现。库使用硬件 3D 加速或包含高度优化的 3D 软件光栅
- **FreeType** ——位图和矢量字体渲染
- **SQLite** ——所有应用程序都可以使用的强大而轻量级的关系数据库引擎

2.4、Application Framework

通过提供开放的开发平台，Android 使开发者能够编制极其丰富和新颖的应用程序。开发者可以自由地利用设备硬件优势、访问位置信息、运行后台服务、设置闹钟、向状态栏添加通知等等，很多很多。

开发者可以完全使用核心应用程序所使用的框架 APIs。应用程序的体系结构旨在简化组件的重用，任何应用程序都能发布他的功能且任何其他应用程序可以使用这些功能（需要服从框架执行的安全限制）。这一机制允许用户替换组件。

所有的应用程序其实是一组服务和系统，包括：

- **视图（View）**——丰富的、可扩展的视图集合，可用于构建一个应用程序。包括包括列表、网格、文本框、按钮，甚至是内嵌的网页浏览器
- **内容提供者（Content Providers）**——使应用程序能访问其他应用程序（如通讯录）的数据，或共享自己的数据

- **资源管理器（Resource Manager）**——提供访问非代码资源，如本地化字符串、图形和布局文件
- **通知管理器（Notification Manager）**——使所有的应用程序能够在状态栏显示自定义警告
- **活动管理器（Activity Manager）**——管理应用程序生命周期,提供通用的导航回退功能

2.5、Applications

Android 装配一个核心应用程序集合，包括电子邮件客户端、SMS 程序、日历、地图、浏览器、联系人和其他设置。所有应用程序都是用 Java 编程语言写的。更加丰富的应用程序有待我们去开发！

3、总结

从上面我们知道 Android 的架构是分层的，非常清晰，分工很明确。Android 本身是一套软件堆叠(Software Stack)，或称为「软件叠层架构」，叠层主要分成三层：操作系统、中间件、应用程序。从上面我们也看到了开源的力量，一个个熟悉的开源软件在这里贡献了自己的一份力量。

现在我们对 android 的系统架构有了一个整体上的了解，我将用一个例子来深入体会一下，但是考虑到此系列希望 0 基础的人也能看懂，在介绍例子之前将介绍一下 **Android 应用程序的原理及一些术语**，可能需要几篇来介绍。敬请关注！