

深入理解Android 5 源代码

作者：李骏

目录

[封面](#)

[扉页](#)

[版权](#)

[前言](#)

[第1章 Android系统介绍](#)

[1.1 Android系统成功的秘诀](#)

[1.1.1 获取了业界的广泛支持](#)

[1.1.2 研发阵容强大](#)

[1.1.3 为开发人员“精心定制”](#)

[1.1.4 开源](#)

[1.2 剖析Android系统架构](#)

[1.2.1 底层操作系统层（OS）](#)

[1.2.2 各种库（Libraries）和Android运行环境（RunTime）](#)

[1.2.3 ApplicationFramework（应用程序框架）](#)

[1.2.4 顶层应用程序（Application）](#)

[1.3 五大组件](#)

[1.3.1 Activity界面](#)

[1.3.2 Intent和IntentFilters切换](#)

[1.3.3 Service（服务）](#)

[1.3.4 BroadcastReceiver发送广播](#)

[1.3.5 用ContentProvider存储数据](#)

[1.4 进程和线程](#)

[1.4.1 什么是进程](#)

[1.4.2 什么是线程](#)

[第2章 获取并编译Android源代码](#)

[2.1 获取Android源代码](#)

[2.1.1 在Linux系统中获取Android源代码](#)

[2.1.2 在Windows平台获取Android源代码](#)

[2.2 分析Android源代码结构](#)

[2.2.1 总体结构](#)

[2.2.2 应用程序部分](#)

[2.2.3 应用程序框架部分](#)

[2.2.4 系统服务部分](#)

[2.2.5 系统程序库部分](#)

[2.2.6 硬件抽象层部分](#)

[2.3 Android源代码提供的接口](#)

[2.3.1 暴露接口和隐藏接口](#)

[2.3.2 调用隐藏接口](#)

[2.4 编译源代码](#)

[2.4.1 搭建编译环境](#)

[2.4.2 在模拟器中运行](#)

[2.5 编译源代码生成SDK](#)

[第3章 分析Java Native Interface系统](#)

[3.1 JNI基础](#)

[3.1.1 JNI的功能结构](#)

[3.1.2 JNI的调用层次](#)

[3.1.3 分析JNI的本质](#)

[3.2 分析MediaScanner](#)

[3.2.1 分析Java层](#)

[3.2.2 分析JNI层](#)

[3.2.3 分析Native（本地）层](#)

[3.3 分析Camera系统的JNI](#)

[3.3.1 Java层预览接口](#)

[3.3.2 注册预览的JNI函数](#)

[3.3.3 C/C++层的预览函数](#)

[第4章 分析HAL系统](#)

[4.1 HAL基础](#)

[4.1.1 推出HAL的背景](#)

[4.1.2 HAL的基本结构](#)

[4.2 分析HAL module架构](#)

[4.2.1 hw_module_t](#)

[4.2.2 结构hw_module_methods_t的定义](#)

[4.2.3 hw_device_t结构](#)

[4.3 分析文件hardware.c](#)

[4.3.1 寻找动态链接库的地址](#)

[4.3.2 数组variant_keys](#)

[4.3.3 载入相应的库](#)

[4.3.4 获得hw_module_t结构体](#)

[4.4 分析硬件抽象层的加载过程](#)

[4.5 分析硬件访问服务](#)

[4.5.1 定义硬件访问服务接口](#)

[4.5.2 具体实现](#)

[4.6 分析Android官方实例](#)

[4.6.1 获取实例工程源代码](#)

[4.6.2 直接调用Service方法的实现代码](#)

[4.6.3 通过Manager调用Service的实现代码](#)

[4.7 HAL和系统移植](#)

[4.7.1 移植各个Android部件的方式](#)

[4.7.2 设置设备权限](#)

[4.7.3 init.rc初始化](#)

[4.7.4 文件系统的属性](#)

[第5章 分析IPC通信机制](#)

[5.1 Binder机制概述](#)

[5.2 分析Binder驱动程序](#)

[5.2.1 分析数据结构](#)

[5.2.2 分析设备初始化](#)

[5.2.3 打开Binder设备文件](#)

[5.2.4 内存映射](#)

[5.2.5 释放物理页面](#)

[5.2.6 分配内核缓冲区](#)

- [5.2.7 释放内核缓冲区](#)
- [5.2.8 查询内核缓冲区](#)
- [5.3 Binder封装库](#)
 - [5.3.1 类BBinder](#)
 - [5.3.2 类BpRefBase](#)
 - [5.3.3 类IPCThreadState](#)
- [5.4 初始化Java层Binder框架](#)
- [5.5 分析MediaServer的通信机制](#)
 - [5.5.1 MediaServer的入口函数](#)
 - [5.5.2 ProcessState](#)
 - [5.5.3 defaultServiceManager](#)
 - [5.5.4 注册MediaPlayerService](#)
 - [5.5.5 分析StartThread_Pool和join_Thread_Pool](#)
- [第6章 分析Binder对象和Java接口](#)
 - [6.1 分析实体对象 \(binder_node\)](#)
 - [6.2 分析本地对象 \(BBinder\)](#)
 - [6.3 分析引用对象 \(binder_ref\)](#)
 - [6.4 分析代理对象 \(BpBinder\)](#)
 - [6.5 分析Java接口](#)
 - [6.5.1 获取Service_Manager](#)
 - [6.5.2 分析ActivityManagerService的Java层](#)
- [第7章 分析ServiceManager和MessageQueue](#)
 - [7.1 分析ServiceManager](#)
 - [7.1.1 分析主入口函数](#)
 - [7.1.2 打开Binder设备文件](#)
 - [7.1.3 注册处理](#)
 - [7.1.4 创建Binder实体对象](#)
 - [7.1.5 尽职的循环](#)
 - [7.1.6 将信息注册到ServiceManager](#)
 - [7.1.7 分析MediaPlayerService和Client](#)
 - [7.2 获得Service_Manager接口](#)
 - [7.3 分析MessageQueue](#)
 - [7.3.1 创建MessageQueue](#)
 - [7.3.2 提取消息](#)
 - [7.3.3 分析函数nativePollOnce](#)
- [第8章 init进程和Zygote进程](#)
 - [8.1 分析init进程](#)
 - [8.1.1 分析入口函数](#)
 - [8.1.2 分析配置文件](#)
 - [8.1.3 分析Service](#)
 - [8.1.4 解析on字段的內容](#)
 - [8.1.5 init控制Service](#)
 - [8.1.6 控制属性服务](#)
 - [8.2 分析Zygote \(孕育\) 进程](#)
 - [8.2.1 Zygote基础](#)
 - [8.2.2 分析Zygote的启动过程](#)
- [第9章 System进程和应用程序进程](#)
 - [9.1 分析System进程](#)
 - [9.1.1 启动System进程前的准备工作](#)
 - [9.1.2 分析SystemServer](#)
 - [9.1.3 分析EntropyService](#)
 - [9.1.4 分析DropBoxManagerService](#)
 - [9.1.5 分析DiskStatsService](#)
 - [9.1.6 分析DeviceStorageManagerService \(监测系统内存存储空间的状态\)](#)
 - [9.1.7 分析SamplingProfilerService](#)
 - [9.2 分析应用程序进程](#)
 - [9.2.1 创建应用程序](#)
 - [9.2.2 启动线程池](#)
 - [9.2.3 创建信息循环](#)
- [第10章 分析Activity组件](#)
 - [10.1 Activity基础](#)
 - [10.1.1 Activity状态](#)
 - [10.1.2 剖析Activity中的主要函数](#)
 - [10.2 分析Activity的启动源代码](#)
 - [10.2.1 Launcher启动应用程序](#)
 - [10.2.2 返回ActivityManagerService的远程接口](#)
 - [10.2.3 解析intent的内容](#)
 - [10.2.4 分析检查机制](#)
 - [10.2.5 执行Activity组件的操作](#)
 - [10.2.6 将Launcher推入Paused状态](#)
 - [10.2.7 处理消息](#)
 - [10.2.8 暂停完毕](#)
 - [10.2.9 建立双向连接](#)
 - [10.2.10 启动新的Activity](#)
 - [10.2.11 通知机制](#)
 - [10.2.12 发送消息](#)
- [第11章 应用程序管理服务——PackageManagerService分析](#)
 - [11.1 PackageManagerService概述](#)
 - [11.2 系统进程启动](#)
 - [11.3 开始运行](#)
 - [11.4 扫描APK文件](#)
 - [11.5 解析并安装文件](#)
 - [11.6 启动系统默认Home应用程序Launcher](#)
 - [11.6.1 设置系统进程](#)
 - [11.6.2 启动Home应用程序](#)
 - [11.6.3 启动com.android.launcher2.Launcher](#)
 - [11.6.4 加载应用程序](#)
 - [11.6.5 获得Activity](#)
- [第12章 Content Provider存储机制](#)
 - [12.1 Content Provider基础](#)
 - [12.1.1 ContentProvider在应用程序中的架构](#)
 - [12.1.2 ContentProvider的常用接口](#)
 - [12.2 启动Content Provider](#)
 - [12.2.1 获得对象接口](#)
 - [12.2.2 存在校验](#)
 - [12.2.3 启动Android应用程序](#)
 - [12.2.4 根据进程启动Content Provider](#)

- [12.2.5 处理消息](#)
- [12.2.6 具体启动](#)
- [12.3 Content Provider数据共享](#)
- [12.3.1 获取接口](#)
- [12.3.2 创建CursorWindow对象](#)
- [12.3.3 数据传递](#)
- [12.3.4 处理进程通信的请求](#)
- [12.3.5 数据操作](#)
- [第13章 分析广播机制源代码](#)
- [13.1 Broadcast基础](#)
- [13.2 发送广播信息](#)
- [13.2.1 intent描述指示](#)
- [13.2.2 传递广播信息](#)
- [13.2.3 封装传递](#)
- [13.2.4 处理发送请求](#)
- [13.2.5 查找广播接收者](#)
- [13.2.6 处理广播信息](#)
- [13.2.7 检查权限](#)
- [13.2.8 处理的进程通信请求](#)
- [13.3 分析BroadcastReceiver](#)
- [13.3.1 MainActivity的调用](#)
- [13.3.2 注册广播接收者](#)
- [13.3.3 获取接口对象](#)
- [13.3.4 处理进程间的通信请求](#)
- [第14章 分析电源管理系统](#)
- [14.1 Power Management架构基础](#)
- [14.2 分析Framework层](#)
- [14.2.1 文件PowerManager.java](#)
- [14.2.2 提供PowerManager功能](#)
- [14.3 JNI层架构分析](#)
- [14.3.1 定义了两层之间的接口函数](#)
- [14.3.2 与Linux Kernel层进行交互](#)
- [14.4 Kernel（内核）层架构分析](#)
- [14.4.1 文件power.c](#)
- [14.4.2 文件earlysuspend.c](#)
- [14.4.3 文件wakelock.c](#)
- [14.4.4 文件resume.c](#)
- [14.4.5 文件suspend.c](#)
- [14.4.6 文件main.c](#)
- [14.4.7 proc文件](#)
- [14.5 wakelock和early_suspend](#)
- [14.5.1 wakelock的原理](#)
- [14.5.2 early_suspend的原理](#)
- [14.5.3 Android休眠](#)
- [14.5.4 Android唤醒](#)
- [14.6 Battery电池系统架构和管理](#)
- [14.6.1 实现驱动程序](#)
- [14.6.2 实现JNI本地代码](#)
- [14.6.3 Java层代码](#)
- [14.6.4 实现Uevent部分](#)
- [14.7 JobScheduler节能调度机制](#)
- [14.7.1 JobScheduler机制的推出背景](#)
- [14.7.2 JobScheduler的实现](#)
- [14.7.3 实现操作调度](#)
- [14.7.4 封装调度任务](#)
- [第15章 分析WindowManagerService系统](#)
- [15.1 WindowManagerService基础](#)
- [15.2 计算Activity窗口的大小](#)
- [15.2.1 实现View遍历](#)
- [15.2.2 函数relayoutWindow](#)
- [15.2.3 函数relayoutWindow](#)
- [15.2.4 拦截消息的处理类](#)
- [15.2.5 判断是否计算过](#)
- [第16章 分析电话系统](#)
- [16.1 Android电话系统详解](#)
- [16.1.1 电话系统简介](#)
- [16.1.2 电话系统结构](#)
- [16.1.3 驱动程序介绍](#)
- [16.1.4 RIL接口](#)
- [16.1.5 分析电话系统的实现流程](#)
- [16.2 电话系统中的音频模块](#)
- [16.2.1 音频系统结构](#)
- [16.2.2 分析音频系统的层次](#)
- [16.3 分析拨号流程](#)
- [16.3.1 拨号界面](#)
- [16.3.2 实现Phone应用](#)
- [16.3.3 Call通话控制](#)
- [16.3.4 静态方法调用](#)
- [16.3.5 通话管理](#)
- [16.3.6 dial拨号](#)
- [16.3.7 状态跟踪](#)
- [16.3.8 RIL消息“出入”口](#)
- [16.3.9 显示通话主界面](#)
- [第17章 分析短信系统](#)
- [17.1 短信系统的主界面](#)
- [17.2 发送普通短信](#)
- [17.3 发送彩信](#)
- [17.4 接收短信](#)
- [17.4.1 Java应用层的接收流程](#)
- [17.4.2 Framework层的处理过程](#)
- [第18章 Sensor传感器系统详解](#)
- [18.1 Android传感器系统概述](#)
- [18.2 Java层详解](#)
- [18.3 Frameworks层详解](#)
- [18.3.1 监听传感器的变化](#)
- [18.3.2 注册监听](#)
- [18.4 JNI层详解](#)

- [18.4.1 实现Native（本地）函数](#)
- [18.4.2 处理客户端数据](#)
- [18.4.3 处理服务端数据](#)
- [18.4.4 封装HAL层的代码](#)
- [18.4.5 处理消息队列](#)
- [18.5 HAL层详解](#)
- [第19章 分析SEAndroid系统](#)
- [19.1 SEAndroid概述](#)
- [19.1.1 内核空间](#)
- [19.1.2 用户空间](#)
- [19.2 文件安全上下文](#)
- [19.2.1 设置打包在ROM里面的文件的安全上下文](#)
- [19.2.2 设置虚拟文件系统的安全上下文](#)
- [19.2.3 设置应用程序数据文件的安全上下文](#)
- [19.3 进程安全上下文](#)
- [19.3.1 为独立进程静态地设置安全上下文](#)
- [19.3.2 为应用程序进程设置安全上下文](#)
- [第20章 分析ART系统](#)
- [20.1 对比Dalvik VM和ART](#)
- [20.2 启动ART](#)
- [20.2.1 运行app_process进程](#)
- [20.2.2 准备启动](#)
- [20.2.3 创建运行实例](#)
- [20.2.4 注册本地JNI函数](#)
- [20.2.5 启动守护进程](#)
- [20.2.6 解析参数](#)
- [20.2.7 初始化类、方法和域](#)
- [20.3 分析主函数main](#)
- [20.4 查找目标类](#)
- [20.4.1 函数LookupClass\(\)](#)
- [20.4.2 函数DefineClass\(\)](#)
- [20.4.3 函数InsertClass\(\)](#)
- [20.4.4 函数LinkClass\(\)](#)
- [20.5 类操作](#)
- [20.6 实现托管操作](#)
- [20.7 加载OAT文件](#)
- [20.7.1 产生OAT](#)
- [20.7.2 创建ART虚拟机](#)
- [20.7.3 解析启动参数并创建堆](#)
- [20.7.4 生成指定目录文件](#)
- [20.7.5 加载OAT文件](#)
- [20.7.6 解析字段](#)

深入理解Android 5源代码

李骏◆编著

人民邮电出版社

北京

图书在版编目 (CIP) 数据

深入理解Android 5源代码/李骏编著.--北京:人民邮电出版社,2016.1

ISBN 978-7-115-40595-1

I.①深... II.①李... III.①移动终端—应用程序—程序设计 IV.①TN929.53

中国版本图书馆CIP数据核字 (2015) 第260754号

内容提要

本书共分20章,循序渐进地分析了Android系统的基本源代码,依次讲解了Android系统介绍,获取并编译Android源代码,分析Java Native Interface系统,分析HAL系统,分析IPC通信机制,分析Binder对象和Java接口,分析ServiceManager和MessageQueue,init进程和Zygote进程,System进程和应用程序进程,分析Activity组件,应用程序管理服务分析,Content Provider、Broadcast (广播)系统,电源管理系统分析,分析WindowManagerService系统、分析电话系统,分析短信系统、Sensor传感器系统详解、分析SEAndroid系统和分析ART系统等核心知识。本书内容言简意赅,讲解方法通俗易懂,不仅适合有一定基础的读者学习,也特别有利于初学者学习。

本书适合Android初学者、Android爱好者、Android底层开发人员、Android应用开发人员学习,也可以作为相关培训学校和大专院校相关专业师生的教学用书。

◆编著 李骏

责任编辑 张涛

责任印制 张佳莹 焦志炜

◆人民邮电出版社出版发行 北京市丰台区成寿寺路11号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京昌平百善印刷厂印刷

◆开本: 787×1092 1/16

印张: 42.75

字数: 1306千字 2016年1月第1版

印数: 1-2500册 2016年1月北京第1次印刷

定价: 99.00元

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第0021号

前言

Android是一款基于Linux平台的开源手机操作系统，该平台由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放的操作系统。

本书的内容

Android系统是Google研发团队集体智慧的结晶，拥有着海量的源代码。本书受篇幅的限制，只分析了Android系统中的一些主要模块和类，并对主要模块的细节进行了全面分析，而相似部分并没有进行详细阐述。这样可以确保读者在有限的篇幅中了解Android的内部结构和运行机制，同时避免读者陷入海量代码的云雾中而不得要领的情况发生。

另外，由于Android系统升级较快，有些代码变动很大。虽然Android系统自2008年9月发布第一个版本1.1以来，截至2014年10月发布最新版本5.0，一共存在十多个版本。在本书的讲解中，选择了本书写作时的最新版本Android 5.0系统，这样可以体验Android系统的最新功能。

全书共分20章，涵盖了Android系统主要的源代码，如HAL系统、IPC通信机制、Binder对象、init进程和Zygote进程、System进程和应用程序进程、Activity组件、Content Provider、Broadcast、电源管理、电话系统、短信系统、传感器、SEAndroid、ART等核心技术。为了帮助读者学以致用，对于重点的模块都会详细剖析其原理和实现的过程，以便读者在自己的项目开发中可以借鉴。

本书特色

本书内容丰富，分析细致，我们的目标是通过一本图书，提供多本图书的价值。在内容的编写上，本书具有以下特色。

(1) 结构合理

从用户的实际需要出发，科学安排知识结构，内容由浅入深。全书详细地讲解了和Android应用开发有关的源代码结构。

(2) 易学易懂

本书条理清晰、语言简洁，可帮助读者快速掌握每个知识点。读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行有针对性的学习。

(3) 实用性强

本书彻底摒弃枯燥的理论，注重实用性和可操作性，详细讲解了各个知识点的基本知识。

读者对象

初学Android编程的自学者；

大中专院校的老师和学生；

毕业设计的学生；

Android编程爱好者；

相关培训机构的老师和学员；

从事Android开发的程序员。

本书在编写过程中，得到了人民邮电出版社工作人员的大力支持，正是各位编辑的求实、耐心和效率，才使得本书在这么短的时间内出版。另外，也十分感谢我的家人，在我写作的时候给予的巨大支持。另外，本人水平毕竟有限，书中的纰漏和不尽如人意之处在所难免，诚请读者提出意见或建议，以便修订并使之更臻完善。另外，本书的答疑和技术交流网站为<http://www.toppr.net/>，读者如有疑问可以在此提出，一定会得到满意的答复。

作者

第1章 Android系统介绍

2007年，Google公司推出了一款移动智能设备系统——Android，这是一种建立在Linux基础之上的为手机、平板电脑等移动设备提供的软件解决方案。截止到2013年，根据知名IDC公司的统计，Android系统在世界智能手机发货量中占据75%的份额，已经成为了当今最受欢迎的智能设备系统之一，2014年更是达到了84.4%。在本章将引领读者一起来了解Android系统的发展历程，充分了解这款操作系统的成功之处。

1.1 Android系统成功的秘诀

Android系统为什么能够在这么短的时间内成为移动智能设备市场占有率的老大？在本节的内容中，将从4个方面来为读者解答这个问题。

1.1.1 获取了业界的广泛支持

Android系统基于Linux内核，是一款开源的手机操作系统。正是因为如此，在Android刚刚崭露头角之后，各大手机厂商和电信部门纷纷加入到了Android联盟当中。Android联盟由业界内的世界级企业组成，主要成员包括中国移动、摩托罗拉、高通、T-Mobile、三星、LG、HTC等在内的30多家技术和无线应用的领军企业。Android通过与运营商、设备制造商、开发商和其他有关各方结成深层次的合作伙伴关系，希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

1.1.2 研发阵容强大

Android的研发队伍阵容强大，包括摩托罗拉、Google、HTC（宏达电子）、PHILIPS、T-Mobile、高通、三星、LG及中国移动在内的34家企业，他们都将基于该平台开发手机的新型业务，应用之间的通用性和互联性将在最大程度上得到保持。

1.1.3 为开发人员“精心定制”

Google公司一直视程序员为前进动力的源泉，为了提高程序员们的开发积极性，不但为开发人员提供了一流的开发装备和软件服务，而且还提出了振奋人心的奖励机制。

- （1）保证开发人员可以迅速转型到Android应用开发

Android应用程序是通过Java语言开发的，只要具备Java开发基础，就能很快上手。作为单独的Android应用开发，对Java编程门槛的要求并不高，即使没有编程经验的初学者，也可以在突击学习Java之后平滑地过渡到Android开发上来。另外，Android完全支持2D、3D和数据库，并且和浏览器实现了集成。所以，通过Android平台，程序员可以迅速、高效地开发出绚丽多彩的应用。

- （2）定期召开奖金丰厚的Android大赛

为了吸引更多的用户使用Android开发，Google已经成功举办了奖金为数千万美元的开发者竞赛。鼓励开发人员创建出创意十足、十分有用的软件。这种大赛对于开发人员来说，不但能提高自己的开发水平，并且高额的奖金也是学员们学习的动力。

- （3）开发人员可以利用自己的应用赚钱

为了能让Android平台吸引更多的关注，Google提供了一个专门下载Android应用的商店：Android Market，地址是<https://play.google.com/store>。在这个商店里允许开发人员发布应用程序，也允许Android用户下载获取自己喜欢的程序。作为开发者，需要申请开发者账号，申请后才能将自己的程序上传到Android Market，并且可以对自己的软件进行定价。只要你的软件程序足够吸引人，你就可以获得很好的金钱回报。这样实现了程序员学习和赚钱两不误，所以吸引了更多开发人员加入到Android大军中来。

1.1.4 开源

Android是一款开源的系统，开源意味着对开发人员和手机厂商来说是完全无偿免费使用的。正是因为这一原因，吸引了全世界各地无数程序员的热情。于是很多手机厂商纷纷采用Android作为自己产品的系统。因为免费所以降低了成本，提高了利润。而对于开发人员来说，因为Android深受众多移动设备产品所采用，所以这方面的人才也变得愈发得到重视。

1.2 剖析Android系统架构

Android系统是一个移动设备的开发平台，其软件层次结构包括操作系统（OS）、中间件（MiddleWare）和应用程序（Application）。根据 Android 的软件框图，其软件层次结构自下而上分为以下4层。

- (1) 操作系统层（OS）。
- (2) 各种库（Libraries）和Android运行环境（RunTime）。
- (3) 应用程序框架（Application Framework）。
- (4) 应用程序（Application）。

上述各个层的具体结构如图1-1所示。

在本节的内容中，将详细介绍Android操作系统的组件结构方面的知识。

。

▲图1-1 Android操作系统的组件结构图

1.2.1 底层操作系统层（OS）

因为Android源于Linux，使用了Linux内核，所以，Android使用Linux 2.6作为操作系统的基础。Android对操作系统的使用包括核心和驱动程序两部分，Android内核对应于Linux内核，Android更多的是需要一些与移动设备相关的驱动程序。主要的驱动如下所示。

显示驱动（Display Driver）：是常用的基于Linux的帧缓冲（Frame Buffer）驱动。

Flash内存驱动（Flash Memory Driver）：是基于MTD的Flash驱动程序。

照相机驱动（Camera Driver）：常用基于Linux的V4L（Video for Linux）驱动。

音频驱动（Audio Driver）：常用基于ALSA（Advanced Linux Sound Architecture，高级Linux声音体系）驱动。

WiFi驱动（Camera Driver）：基于IEEE 802.11标准的驱动程序。

键盘驱动（KeyBoard Driver）：作为输入设备的键盘驱动。

蓝牙驱动（Bluetooth Driver）：基于IEEE 802.15.1标准的无线传输技术。

Binder IPC驱动：Android一个特殊的驱动程序，具有单独的设备节点，提供进程间通信的功能。

Power Management（能源管理）：用于管理电池电量等信息。

1.2.2 各种库（Libraries）和Android运行环境（RunTime）

本层次对应一般嵌入式系统，相当于中间件层次。Android的本层次分成两个部分，一个是各种库，另一个是Android运行环境。本层的内容大多是使用C实现的，其中包含了如下所示的各种库。

C库：C语言的标准库，也是系统中一个最为底层的库，C库是通过Linux的系统调用来实现。

多媒体框架（MediaFramework）：这部分内容是Android多媒体的核心部分，基于PacketVideo（即PV）的OpenCORE，从功能上看本库一共分为两大部分，一部分是音频、视频的回放（PlayBack），另一部分是音、视频的记录（Recorder）。

SGL：2D图像引擎。

SSL：即Secure Socket Layer，位于TCP/IP协议与各种应用层协议之间，为数据通信提供安全支持。

OpenGL ES：提供了对3D的支持。

界面管理工具（Surface Management）：提供了管理显示子系统等功能。

SQLite：一个通用的嵌入式数据库。

WebKit：网络浏览器的核心。

FreeType：位图和矢量字体的功能。

在一般情况下，Android的各种库是以系统中间件的形式提供的，它们的显著特点是与移动设备平台的应用密切相关。另外，Android的运行环境主要是指Dalvik（虚拟机）技术。Dalvik和一般的Java虚拟机（Java VM）是有区别的。

Java虚拟机：执行的是Java标准的字节码（Bytecode）。

ART+Dalvik：执行的是Dalvik可执行格式（.dex）中的执行文件。在执行的过程中，每一个应用程序即一个进程（Linux的一个Process）。二者最大的区别在于Java VM是以基于栈的虚拟机（Stack-based），而Dalvik是基于寄存器的虚拟机（Register-based）。显然，后者最大的好处在于可以根据硬件实现更大的优化，这更适合移动设备的特点。从Android 5.0版本开始，Android的默认运行环境为ART。ART的机制与Dalvik不同。在Dalvik下，应用每次运行的时候，字节码都需要通过即时编译器转换为机器码，这会拖慢应用的运行效率，而在ART环境中，应用在第一次安装的时候，字节码就会预先编译成机器码，使其成为真正的本地应用。这个过程叫做预编译（AOT，Ahead-Of-Time）。这样改进后，应用的启动（首次）和执行都会变得更加快速。

1.2.3 ApplicationFramework（应用程序框架）

在整个Android系统中，和应用开发最相关的是Application Framework，在这一层，Android为应用程序层的开发者提供了各种功能强大的APIs，这实际上是一个应用程序的框架。由于上层的应用程序是以Java构建的。在本层提供了程序中所需要的各种控件，例如，视图组件（Views）、列表（List）、栅格（Grid）、文本框（Text Box）、按钮（Button），甚至还有一个嵌入式的Web浏览器。

一个基本的Android应用程序可以利用应用程序框架中的以下5个部分。

Activity：活动。

Broadcast Intent Receiver：广播意图接收者。

Service：服务。

欢迎访问：电子书学习和下载网站 (<https://www.shgis.cn>)

文档名称：深入理解Android 5 源代码 - 李骏.epub

请登录 <https://shgis.cn/post/1866.html> 下载完整文档。

手机端请扫码查看：

