

Android 进阶解密

刘望舒◎著



电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

内 容 简 介

本书是一本 Android 进阶书籍，主要针对 Android 8.0 系统源码并结合应用开发相关知识进行介绍。本书共分为 17 章，从 3 个方面来组织内容。

第一方面介绍 Android 应用开发所需要掌握的系统源码知识，第二方面介绍 JNI、ClassLoader、Java 虚拟机、DVM&ART 虚拟机和 Hook 等技术，第三方面介绍热修复原理、插件化原理、绘制优化和内存优化等与应用开发相关的知识点。3 个方面有所关联并形成知识体系，从而使 Android 开发者能通过阅读本书达到融会贯通的目的。

本书适合有一定基础的 Android 应用开发工程师、Android 系统开发工程师和对 Android 系统源码感兴趣的读者阅读。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目 (CIP) 数据

Android 进阶解密 / 刘望舒著. —北京: 电子工业出版社, 2018.10

ISBN 978-7-121-34838-9

I. ①A… II. ①刘… 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 (2018) 第 179236 号

策划编辑: 付 睿

责任编辑: 牛 勇 特约编辑: 赵树刚

印 刷: 三河市良远印务有限公司

装 订: 三河市良远印务有限公司

出版发行: 电子工业出版社

北京市海淀区万寿路 173 信箱 邮编: 100036

开 本: 787×980 1/16 印张: 29.25 字数: 702 千字

版 次: 2018 年 10 月第 1 版

印 次: 2018 年 10 月第 1 次印刷

定 价: 99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 zltz@phei.com.cn，盗版侵权举报请发邮件到 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819, faq@phei.com.cn。

前言

为什么写这本书

Android 进阶三部曲包括《Android 进阶之光》和本书，因此写这本书的原因和《Android 进阶之光》有些关联，主要有以下几点：

（1）《Android 进阶之光》适合初、中级工程师阅读，因此我需要写一本适合中、高级工程师阅读的进阶书。

（2）目前市场上的系统源码分析的书大部分不是专门为应用开发编写的，因此我要专门为 Android 应用开发编写一本系统源码分析的书，不仅如此，我还要将系统源码和应用开发相结合。

（3）目前 Android 应用综合类进阶书籍很少，在 2017 年出版的只有《Android 进阶之光》，在 2018 年我仍要贡献出自己的力量。

（4）目前市面上的源码分析类书籍大部分是基于 Android 6.0 以前版本的，需要有一本书籍来对更新版本的系统源码进行分析。

（5）《Android 进阶之光》覆盖的知识点还远远不够，我希望能覆盖更多的知识点，让更多的人受益。

本书内容

本书共分为 17 章，各章内容如下：

- 第 1 章介绍 Android 系统架构、系统源码目录和如何阅读源码，带领大家走进 Android 系统源码的世界。

- 第 2 章介绍 Android 系统启动过程，为下面的章节做好铺垫。
- 第 3 章介绍应用程序进程启动过程。
- 第 4 章介绍四大组件的工作过程，包括根 Activity 的启动过程，Service 的启动和绑定过程，广播的注册、发送和接收过程，Content Provider 的启动过程。
- 第 5 章从源码角度分析上下文 Context。
- 第 6 章介绍 ActivityManagerService，包括 AMS 家族、AMS 的启动过程、AMS 重要的数据结构和 Activity 栈管理等内容。
- 第 7 章介绍 WindowManager，包括 WindowManager 的关联类、Window 的属性和 Window 的操作等内容。
- 第 8 章介绍 WindowManagerService，包括 WMS 的创建过程、WMS 的重要成员和 Window 的添加过程等内容。
- 第 9 章结合 MediaRecorder 框架来介绍 JNI 的原理。
- 第 10 章介绍 Android 开发所需要了解的 Java 虚拟机知识。
- 第 11 章介绍 Dalvik 和 ART 虚拟机。
- 第 12 章介绍 ClassLoader，它是理解热修复原理和插件化原理必备的知识点。
- 第 13 章介绍热修复原理，包括热修复框架的对比、资源修复、代码修复和动态链接库的修复。
- 第 14 章介绍 Hook 技术，为讲解插件化原理做铺垫。
- 第 15 章介绍插件化原理，包括插件化的产生、四大组件的插件化、资源的插件化和 so 的插件化。
- 第 16 章介绍绘制优化，包括绘制性能分析和布局优化。
- 第 17 章介绍内存优化，从避免内存泄漏开始讲起，然后介绍常用的内存分析工具 Memory Monitor、Allocation Tracker 和 Heap Dump，最后介绍分析内存泄漏的利器 MAT 和 LeakCanary。

本书特色

本书主要有以下特点：

- 本书的知识点自成体系并且环环相扣，每一个章节都或多或少地与其他章节有所关联。
- 本书是目前市面上少有的专门为 Android 应用开发者所编写的源码分析类书籍，并且将系统源码和应用开发相结合。

- 本书是目前市面上少有的讲解插件化和热修复原理的书。
- 本书为了更好地讲解知识点，会先介绍一些知识点做铺垫，比如要学习插件化原理，就需要先学习四大组件工作过程、AMS、ClassLoader 和 Hook 技术等相关知识点。

读者对象

本书适合以下读者阅读：

- 有一定基础的 Android 应用开发工程师。
- Android 系统开发工程师。
- 对 Android 系统源码感兴趣的读者。

致谢

感谢本书的策划编辑付睿，她在 CSDN 博客中发现了 我，并积极推动本书的出版进度，才使得本书能够及时出版。感谢我的父母以及所有关注我的朋友，你们的鼓励和认可为我写书以及写博客带来了源源不断的动力。

勘误与互动

本人虽已竭尽全力，但书中难免会有错误，欢迎大家向我反馈，我也会在独立博客和 CSDN 博客中定期发布本书的勘误信息。

本书互动地址

独立博客：<http://liuwangshu.cn>

CSDN 博客：<http://blog.csdn.net/itachi85>

Github：<https://github.com/henrymorgen>

微信公众号：刘望舒

QQ 交流群：499174415

刘望舒

2018 年 6 月于北京

目录

第 1 章	Android 系统架构	1
1.1	Android 系统架构.....	1
1.2	Android 系统源码目录.....	4
1.2.1	整体结构	4
1.2.2	应用层部分	5
1.2.3	应用框架层部分	6
1.2.4	C/C++程序库部分	6
1.3	源码阅读	7
1.3.1	在线阅读	7
1.3.2	使用 Source Insight	9
1.4	本章小结	12
第 2 章	Android 系统启动	13
2.1	init 进程启动过程.....	13
2.1.1	引入 init 进程	13
2.1.2	init 进程的入口函数	14
2.1.3	解析 init.rc.....	17
2.1.4	解析 Service 类型语句	19
2.1.5	init 启动 Zygote.....	20
2.1.6	属性服务	23
2.1.7	init 进程启动总结	27

2.2	Zygote 进程启动过程.....	27
2.2.1	Zygote 概述.....	28
2.2.2	Zygote 启动脚本.....	28
2.2.3	Zygote 进程启动过程介绍.....	30
2.2.4	Zygote 进程启动总结.....	38
2.3	SystemService 处理过程.....	39
2.3.1	Zygote 处理 SystemServer 进程.....	39
2.3.2	解析 SystemServer 进程.....	44
2.3.3	SystemService 进程总结.....	48
2.4	Launcher 启动过程.....	48
2.4.1	Launcher 概述.....	48
2.4.2	Launcher 启动过程介绍.....	49
2.4.3	Launcher 中应用图标显示过程.....	54
2.5	Android 系统启动流程.....	59
2.6	本章小结.....	60
第 3 章	应用程序进程启动过程.....	61
3.1	应用程序进程简介.....	61
3.2	应用程序进程启动过程介绍.....	62
3.2.1	AMS 发送启动应用程序进程请求.....	62
3.2.2	Zygote 接收请求并创建应用程序进程.....	68
3.3	Binder 线程池启动过程.....	75
3.4	消息循环创建过程.....	78
3.5	本章小结.....	80
第 4 章	四大组件的工作过程.....	81
4.1	根 Activity 的启动过程.....	82
4.1.1	Launcher 请求 AMS 过程.....	82
4.1.2	AMS 到 ApplicationThread 的调用过程.....	85
4.1.3	ActivityThread 启动 Activity 的过程.....	94
4.1.4	根 Activity 启动过程中涉及的进程.....	99
4.2	Service 的启动过程.....	101

4.2.1	ContextImpl 到 AMS 的调用过程.....	101
4.2.2	ActivityThread 启动 Service.....	103
4.3	Service 的绑定过程.....	110
4.3.1	ContextImpl 到 AMS 的调用过程.....	111
4.3.2	Service 的绑定过程.....	112
4.4	广播的注册、发送和接收过程.....	122
4.4.1	广播的注册过程.....	122
4.4.2	广播的发送和接收过程.....	127
4.5	Content Provider 的启动过程.....	137
4.5.1	query 方法到 AMS 的调用过程.....	137
4.5.2	AMS 启动 Content Provider 的过程.....	143
4.6	本章小结.....	148
第 5 章	理解上下文 Context.....	149
5.1	Context 的关联类.....	149
5.2	Application Context 的创建过程.....	151
5.3	Application Context 的获取过程.....	156
5.4	Activity 的 Context 创建过程.....	156
5.5	Service 的 Context 创建过程.....	161
5.6	本章小结.....	163
第 6 章	理解 ActivityManagerService.....	164
6.1	AMS 家族.....	164
6.1.1	Android 7.0 的 AMS 家族.....	164
6.1.2	Android 8.0 的 AMS 家族.....	170
6.2	AMS 的启动过程.....	171
6.3	AMS 与应用程序进程.....	174
6.4	AMS 重要的数据结构.....	176
6.4.1	解析 ActivityRecord.....	177
6.4.2	解析 TaskRecord.....	177
6.4.3	解析 ActivityStack.....	178
6.5	Activity 栈管理.....	181

6.5.1	Activity 任务栈模型	181
6.5.2	Launch Mode	182
6.5.3	Intent 的 FLAG	182
6.5.4	taskAffinity	185
6.6	本章小结	186
第 7 章	理解 WindowManager	187
7.1	Window、WindowManager 和 WMS	187
7.2	WindowManager 的关联类	188
7.3	Window 的属性	193
7.3.1	Window 的类型和显示次序	193
7.3.2	Window 的标志	195
7.3.3	软键盘相关模式	196
7.4	Window 的操作	196
7.4.1	系统窗口的添加过程	197
7.4.2	Activity 的添加过程	202
7.4.3	Window 的更新过程	203
7.5	本章小结	206
第 8 章	理解 WindowManagerService	207
8.1	WMS 的职责	207
8.2	WMS 的创建过程	209
8.3	WMS 的重要成员	217
8.4	Window 的添加过程 (WMS 处理部分)	219
8.5	Window 的删除过程	225
8.6	本章小结	230
第 9 章	JNI 原理	231
9.1	系统源码中的 JNI	232
9.2	MediaRecorder 框架中的 JNI	233
9.2.1	Java Framework 层的 MediaRecorder	233
9.2.2	JNI 层的 MediaRecorder	234

9.2.3	Native 方法注册	235
9.3	数据类型的转换	239
9.3.1	基本数据类型的转换	240
9.3.2	引用数据类型的转换	240
9.4	方法签名	242
9.5	解析 JNIEnv	244
9.5.1	jfieldID 和 jmethodID	245
9.5.2	使用 jfieldID 和 jmethodID	247
9.6	引用类型	249
9.6.1	本地引用	249
9.6.2	全局引用	249
9.6.3	弱全局引用	250
9.7	本章小结	251
第 10 章	Java 虚拟机	252
10.1	概述	252
10.1.1	Java 虚拟机家族	253
10.1.2	Java 虚拟机执行流程	253
10.2	Java 虚拟机结构	254
10.2.1	Class 文件格式	255
10.2.2	类的生命周期	256
10.2.3	类加载子系统	257
10.2.4	运行时数据区域	258
10.3	对象的创建	260
10.4	对象的堆内存布局	262
10.5	oop-klass 模型	263
10.6	垃圾标记算法	266
10.6.1	Java 中的引用	266
10.6.2	引用计数算法	267
10.6.3	根搜索算法	269
10.7	Java 对象在虚拟机中的生命周期	270
10.8	垃圾收集算法	271

10.8.1	标记—清除算法	271
10.8.2	复制算法	272
10.8.3	标记—压缩算法	273
10.8.4	分代收集算法	274
10.9	本章小结	275
第 11 章	Dalvik 和 ART	276
11.1	Dalvik 虚拟机	276
11.1.1	DVM 与 JVM 的区别	276
11.1.2	DVM 架构	278
11.1.3	DVM 的运行时堆	280
11.1.4	DVM 的 GC 日志	280
11.2	ART 虚拟机	281
11.2.1	ART 与 DVM 的区别	281
11.2.2	ART 的运行时堆	282
11.2.3	ART 的 GC 日志	283
11.3	DVM 和 ART 的诞生	285
11.4	本章小结	288
第 12 章	理解 ClassLoader	289
12.1	Java 中的 ClassLoader	289
12.1.1	ClassLoader 的类型	289
12.1.2	ClassLoader 的继承关系	291
12.1.3	双亲委托模式	292
12.1.4	自定义 ClassLoader	295
12.2	Android 中的 ClassLoader	298
12.2.1	ClassLoader 的类型	298
12.2.2	ClassLoader 的继承关系	300
12.2.3	ClassLoader 的加载过程	302
12.2.4	BootClassLoader 的创建	306
12.2.5	PathClassLoader 的创建	309
12.3	本章小结	311

第 13 章 热修复原理	312
13.1 热修复的产生	312
13.2 热修复框架的种类和对比	313
13.3 资源修复	314
13.3.1 Instant Run 概述	314
13.3.2 Instant Run 的资源修复	315
13.4 代码修复	318
13.4.1 类加载方案	319
13.4.2 底层替换方案	321
13.4.3 Instant Run 方案	322
13.5 动态链接库的修复	323
13.5.1 System 的 load 和 loadLibrary 方法	323
13.5.2 nativeLoad 方法分析	327
13.6 本章小结	333
第 14 章 Hook 技术	334
14.1 Hook 技术概述	334
14.2 Hook 技术分类	336
14.3 代理模式	336
14.3.1 代理模式简单实现	337
14.3.2 动态代理的简单实现	338
14.4 Hook startActivity 方法	339
14.4.1 Hook Activity 的 startActivity 方法	340
14.4.2 Hook Context 的 startActivity 方法	343
14.4.3 Hook startActivity 总结	344
14.5 本章小结	345
第 15 章 插件化原理	346
15.1 动态加载技术	346
15.2 插件化的产生	347
15.2.1 应用开发的痛点和瓶颈	347
15.2.2 插件化思想	348

15.2.3	插件化定义	350
15.3	插件化框架对比	351
15.4	Activity 插件化	352
15.4.1	Activity 的启动过程回顾	352
15.4.2	Hook IActivityManager 方案实现	354
15.4.3	Hook Instrumentation 方案实现	364
15.4.4	总结	367
15.5	Service 插件化	368
15.5.1	插件化方面 Service 与 Activity 的不同	368
15.5.2	代理分发实现	370
15.6	ContentProvider 插件化	376
15.6.1	ContentProvider 的启动过程回顾	376
15.6.2	VirtualApk 的实现	377
15.7	BroadcastReceiver 的插件化	385
15.7.1	广播插件化思路	386
15.7.2	VirtualApk 的实现	386
15.8	资源的插件化	387
15.8.1	系统资源加载	387
15.8.2	VirtualApk 实现	389
15.9	so 的插件化	390
15.10	本章小结	393
第 16 章	绘制优化	394
16.1	绘制性能分析	394
16.1.1	绘制原理	395
16.1.2	Profile GPU Rendering	396
16.1.3	Systrace	398
16.1.4	Traceview	404
16.2	布局优化	407
16.2.1	布局优化工具	407
16.2.2	布局优化方法	411
16.2.3	避免 GPU 过度绘制	419

16.3	本章小结	420
第 17 章	内存优化	421
17.1	避免可控的内存泄漏	421
17.1.1	什么是内存泄漏	421
17.1.2	内存泄漏的场景	422
17.2	Memory Monitor	428
17.2.1	使用 Memory Monitor	429
17.2.2	大内存申请与 GC	430
17.2.3	内存抖动	430
17.3	Allocation Tracker	430
17.3.1	使用 Allocation Tracker	431
17.3.2	alloc 文件分析	431
17.4	Heap Dump	434
17.4.1	使用 Heap Dump	434
17.4.2	检测内存泄漏	436
17.5	内存分析工具 MAT	438
17.5.1	生成 hprof 文件	438
17.5.2	MAT 分析 hprof 文件	440
17.6	LeakCanary	448
17.6.1	使用 LeakCanary	449
17.6.2	LeakCanary 应用举例	449
17.7	本章小结	453

第 1 章

Android 系统架构

本书是系统源码和应用开发相结合的书籍，想要更好地学习系统源码，就要先了解 Android 系统架构。本章将简单地介绍 Android 系统架构、系统源码目录和如何阅读源码，带领大家走进 Android 系统源码的世界，并为本书后续的章节做好知识铺垫。另外，本书作为《Android 进阶之光》的续作，本章也起到了承上启下的作用。

1.1 Android 系统架构

Android 系统架构分为五层，从上到下依次是应用层、应用框架层、系统运行库层、硬件抽象层和 Linux 内核层，如图 1-1 所示。

1. 应用层（System Apps）

系统内置的应用程序以及非系统级的应用程序都属于应用层，负责与用户进行直接交互，通常都是用 Java 进行开发的。

2. 应用框架层（Java API Framework）

应用框架层为开发人员提供了开发应用程序所需要的 API，我们平常开发应用程序都是调用这一层所提供的 API，当然也包括系统应用。这一层是由 Java 代码编写的，可以称为 Java Framework。下面来看这一层所提供的主要组件，如表 1-1 所示。

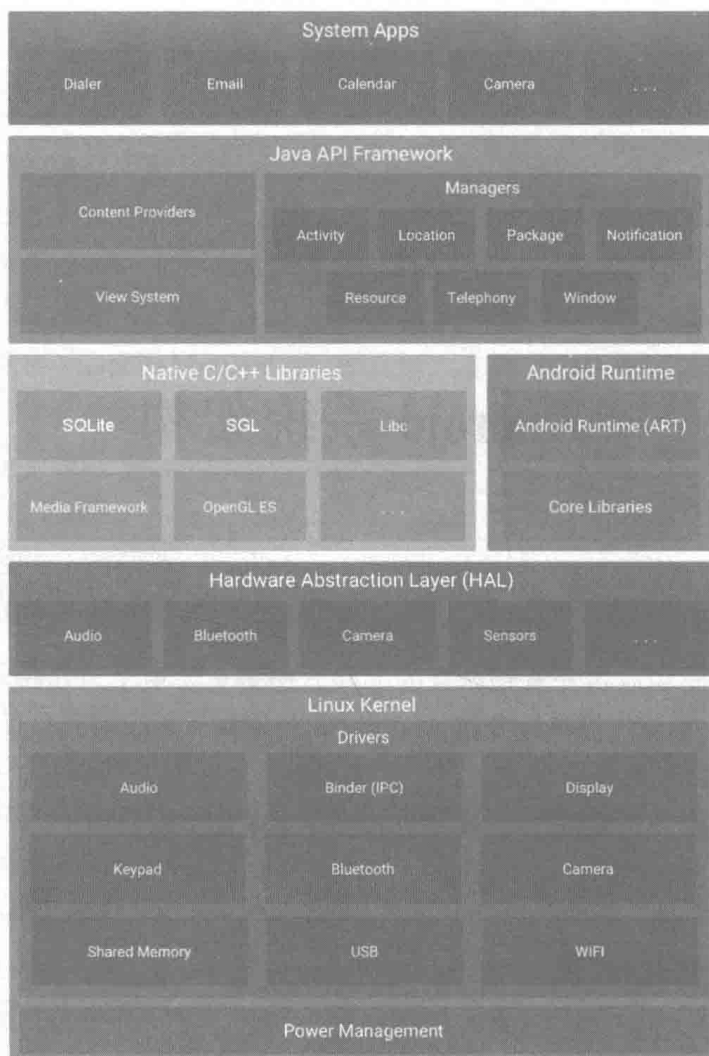


图 1-1 Android 系统架构

表 1-1 应用框架层提供的组件

名 称	功 能 描 述
Activity Manager (活动管理器)	管理各个应用程序生命周期, 以及常用的导航回退功能
Location Manager (位置管理器)	提供地理位置及定位功能服务
Package Manager (包管理器)	管理所有安装在 Android 系统中的应用程序
Notification Manager (通知管理器)	使得应用程序可以在状态栏中显示自定义的提示信息
Resource Manager (资源管理器)	提供应用程序使用的各种非代码资源, 如本地化字符串、图片、布局文件、颜色文件等
Telephony Manager (电话管理器)	管理所有的移动设备功能