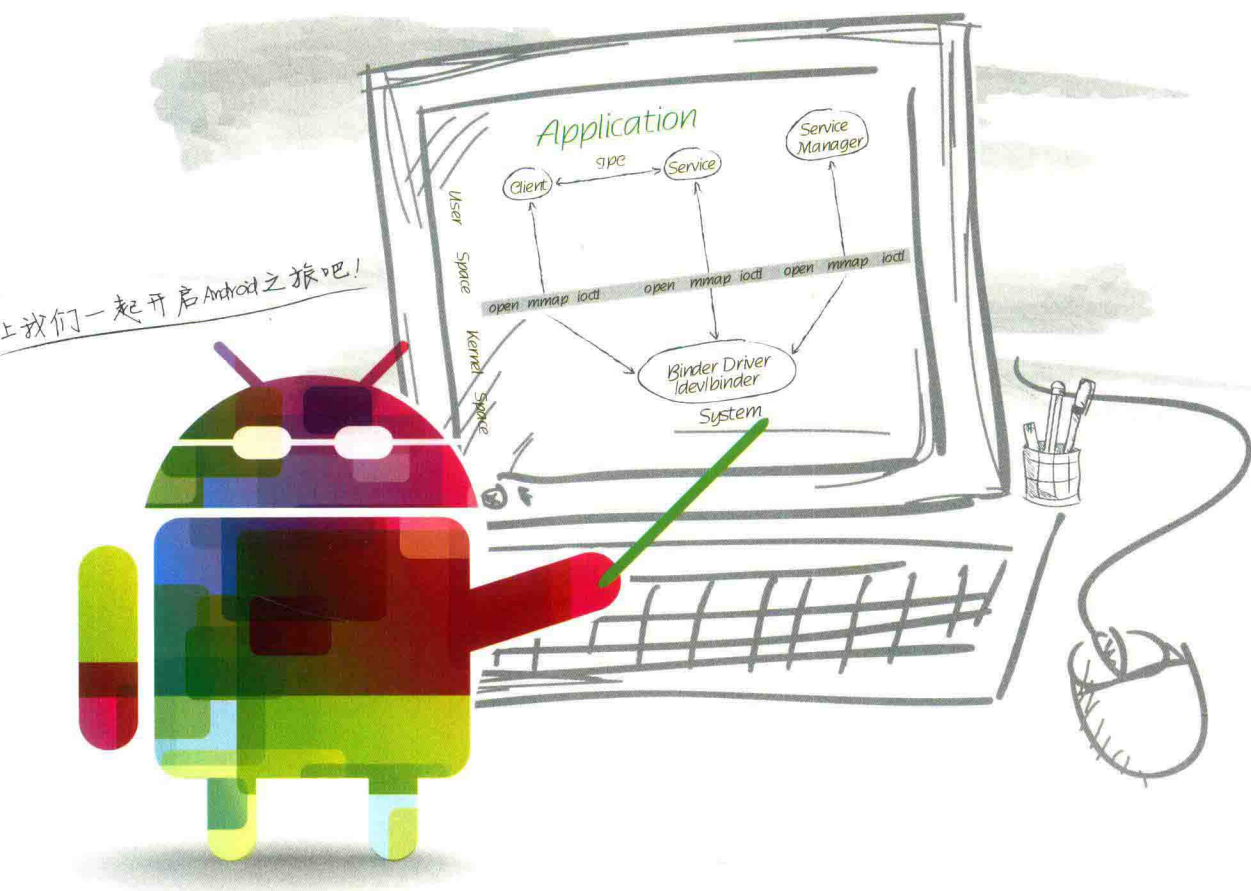


全面、深入、细致地掌握Android系统，引领移动互联网新时代！

# Android

## 系统源代码情景分析 (第三版)

◎ 罗升阳 著 ◎



# Android

系统源代码情景分析 (第三版)

---

◎ 罗升阳 著 ◎

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

在内容上,本书结合使用情景,全面、深入、细致地分析了Android系统的源代码,涉及Linux内核层、硬件抽象层(HAL)、运行时库层(Runtime)、应用程序框架层(Application Framework)及应用程序层(Application)。

在组织上,本书将上述内容划分为初识Android系统、Android专用驱动系统及Android应用程序框架三大篇。初识Android系统篇介绍了参考书籍、基础知识及实验环境搭建;Android专用驱动系统篇介绍了Logger日志驱动程序、Binder进程间通信驱动程序及Ashmem匿名共享内存驱动程序;Android应用程序框架篇从组件、进程、消息和安装四个维度对Android应用程序的框架进行了深入的剖析。

通过上述内容及其组织,本书使读者既能从整体上把握Android系统的层次结构,又能从细节上掌握每一个层次的重点。

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系统源代码情景分析/罗升阳著.—3版.—北京:电子工业出版社,2017.10  
ISBN 978-7-121-32521-2

I. ①A… II. ①罗… 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(2017)第199928号

策划编辑:符隆美

责任编辑:葛娜

印刷:北京京科印刷有限公司

装订:三河市华成印务有限公司

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:850×1168 1/16

印张:53

字数:1585千字

版次:2012年10月第1版

2017年10月第3版

印次:2017年10月第1次印刷

定 价:129.00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zlt@phei.com.cn,盗版侵权举报请发邮件至dbqq@phei.com.cn。

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

# 前 言

Android系统自2008年9月发布第一个版本1.0以来，截至2016年8月发布最新版本7.0，一共存在十多个版本。本书在2012年10月出版第一版时，选择了Android 2.3的源代码来分析Android系统的实现，主要是因为就当时而言，它的基础架构是最稳定的，而且也是使用最广泛的。

本书有两个特点。一是通过使用情景来分析Android源代码。这种方式不仅能使读者带着目标去阅读源代码，还能把系统各个模块有机地联系在一起理解。二是分析的内容都是Android系统的核心基础架构。这些内容在后来的版本中，都是灵魂般地稳定存在的。例如，智能指针、硬件抽象层（HAL）、匿名共享内存（Ashmem）、Binder IPC和四大组件管理等。

因此，读者阅读了本书之后，能够自行去阅读和分析其他的Android源代码。这种“授人以鱼，不如授人以渔”的效果，正是本书所追求的目标。因为技术发展日新月异，只有掌握了本质和宗旨，才能以不变应万变。

笔者记得2010年学习Linux内核时，一开始就从当时的最新版本（2.6.32）入手，结果犹如跌入茫茫大海之中。高版本的源代码，为了解决和优化一些问题，往往会引进一些复杂的概念，以及增加一些难以理解的代码。对于初学者来说，这些复杂的概念和难以理解的代码，会阻碍他们理解代码的核心思想。后来笔者改为从0.11版本（Linux的第一个基本可以正常运行的内核版本，于1991年发布）入手，才逐渐进入状态。《Linux内核完全注释》这本书，就是以0.11版本为蓝本，对Linux内核进行分析的。

本书自第一版发布以来，经历了6次重印，得到了读者的热烈反馈和建议。基于这些反馈和建议，本书在2015年12月进行了相应的修订。在不到两年的时间内，修订版又经历了4次重印，这充分说明了读者对本书的厚爱。本次修订仍以Android 2.3版本的源代码为蓝本，并且根据读者持续的反馈和建议（<http://blog.csdn.net/luoshengyang/article/details/8116866>）进行。由于读者提供的反馈和建议众多，这里就不一一列出，具体可以参考上述链接。

同时，本次修订还根据最新的官方文档，优化了Android源代码编译环境搭建的内容，使读者可以简单、快速、准确地初始化好Android源代码编译环境。另外，本次修订还增加了一节内容，介绍如何为真机编译ROM。让自己编译的Android源代码运行在真机上，是一件令人兴奋的事情，一方面，可以加深对Android系统的理解；另一方面，可以随心所欲地定制系统的行为，这不仅可以满足特殊的手机使用需求，还能辅助日常的开发工作。

为了更好地帮助读者理解本书的内容，笔者创建了一个网站（<http://0xcc0xcd.com>）。该网站会陆续提供一些免费视频，讲解Android系统相关的知识。最后，希望本书能够继续为有志于深入研究Android的读者服务。同时也非常感谢读者的反馈和建议，你们的支持是笔者持续研究和分享Android系统技术的动力所在。

## 本书内容

全书分为初识Android系统篇、Android专用驱动系统篇和Android应用程序框架篇三个部分。

**初识Android系统篇**包含三个章节的内容，主要介绍Android系统的基础知识。第1章介绍与Android系统有关的参考书籍、Android源代码工程环境的搭建方法，以及为真机编译ROM的过程；第2章介绍Android系统的硬件抽象层；第3章介绍Android系统的智能指针。读者可能会觉得奇怪，为什么一开始就介绍Android系统的硬件抽象层呢？因为涉及硬件，它似乎是一个深奥的知识点。其实不然，Android系统的硬件抽象层无论是从实现上，还是从使用上，它的层次都是非常清晰的；而且从下到上涵盖了整个Android系统，包括Android系统在用户空间和内核空间的实现。内核空间主要涉及硬件驱动程序的编写方法，而用户空间涉及运行时库层、应用程序框架层及应用程序层。因此，尽早学习Android系统的硬件抽象层，有助于我们从整体上去认识Android系统，以便后面可以更好地分析它的源代码。在分析Android系统源代码的过程中，经常会碰到智能指针，第3章我们就重点分析Android系统智能指针的实现原理，也是为了后面可以更好地分析Android系统源代码。

**Android专用驱动系统篇**包含三个章节的内容。我们知道，Android系统是基于Linux内核来开发的，但是由于移动设备的CPU和内存配置都要比PC低，因此，Android系统并不是完全在Linux内核上开发的，而是在Linux内核里面添加了一些专用的驱动模块来使它更适合于移动设备。这些专用的驱动模块同时也形成了Android系统的坚实基础，尤其是Logger日志驱动程序、Binder进程间通信驱动程序，以及Ashmem匿名共享内存驱动程序，它们在Android系统中被广泛地使用。在此篇中，我们分别在第4章、第5章和第6章分析Logger日志系统、Binder进程间通信系统和Ashmem共享内存系统的实现原理，为后面深入分析Android应用程序的框架打下良好的基础。

**Android应用程序框架篇**包含十个章节的内容。我们知道，在移动平台中，Android系统和iOS系统比的是谁的应用程序更丰富、质量更高、用户体验更好，谁就能取得最终的胜利。因此，每个平台都在尽最大努力吸引第三方开发者来为其开发应用程序。这就要求平台必须提供良好的应用程序架构，以便第三方开发者可以将更多的精力集中在应用程序的业务逻辑上，从而开发出数量更多、质量更高和用户体验更好的应用程序。在此篇中，我们将从组件、进程、消息和安装四个维度来分析Android应用程序的实现框架。第7章到第10章分析Android应用程序四大组件Activity、Service、Broadcast Receiver和Content Provider的实现原理；第11章和第12章分析Android应用程序进程的启动过程；第13章到第15章分析Android应用程序的消息处理机制；第16章分析Android应用程序的安装和显示过程。学习了这些知识之后，我们就可以掌握Android系统的精髓了。

## 本书特点

本书从初学者的角度出发，结合具体的使用情景，在纵向和横向上对Android系统的源代码进行了全面、深入、细致的分析。在纵向上，采用从下到上的方式，分析的源代码涉及Android系统的内核层（Linux Kernel）、硬件抽象层（HAL）、运行时库层（Runtime）、应用程序框架层（Application Framework）以及应用程序层（Application），这有利于读者从整体上掌握Android系统的架构。在横向上，从Android应用程序的组件、进程、消息和安装四个维度出发，全面地剖析Android系统的应用程序框架层，这有利于读者深入理解Android应用程序的架构及运行原理。

# 目 录

## 第1篇 初识Android系统

<b>第1章 准备知识</b> .....	<b>2</b>
1.1 Linux内核参考书籍 .....	2
1.2 Android应用程序参考书籍 .....	3
1.3 下载、编译和运行Android源代码 .....	3
1.3.1 下载Android源代码 .....	5
1.3.2 编译Android源代码 .....	6
1.3.3 运行Android模拟器 .....	7
1.4 下载、编译和运行Android内核源代码 .....	8
1.4.1 下载Android内核源代码 .....	8
1.4.2 编译Android内核源代码 .....	8
1.4.3 运行Android模拟器 .....	9
1.5 开发第一个Android应用程序 .....	10
1.6 单独编译和打包Android应用程序模块 .....	12
1.6.1 导入单独编译模块的mmm命令 .....	12
1.6.2 单独编译Android应用程序模块 .....	13
1.6.3 重新打包Android系统镜像文件 .....	13
1.7 为真机编译ROM .....	14
1.7.1 下载LineageOS .....	16
1.7.2 下载设备开源代码 .....	16
1.7.3 下载设备私有文件 .....	17
1.7.4 编译ROM .....	17
1.7.5 刷入TWRP .....	17
1.7.6 刷入ROM .....	19
<b>第2章 硬件抽象层</b> .....	<b>21</b>
2.1 开发Android硬件驱动程序 .....	22
2.1.1 实现内核驱动程序模块 .....	22
2.1.2 修改内核Kconfig文件 .....	29

2.1.3	修改内核Makefile文件 .....	30
2.1.4	编译内核驱动程序模块 .....	30
2.1.5	验证内核驱动程序模块 .....	31
2.2	开发C可执行程序验证Android硬件驱动程序 .....	32
2.3	开发Android硬件抽象层模块 .....	34
2.3.1	硬件抽象层模块编写规范 .....	34
2.3.2	编写硬件抽象层模块接口 .....	37
2.3.3	硬件抽象层模块的加载过程 .....	41
2.3.4	处理硬件设备访问权限问题 .....	44
2.4	开发Android硬件访问服务 .....	46
2.4.1	定义硬件访问服务接口 .....	46
2.4.2	实现硬件访问服务 .....	47
2.4.3	实现硬件访问服务的JNI方法 .....	48
2.4.4	启动硬件访问服务 .....	51
2.5	开发Android应用程序来使用硬件访问服务 .....	52
<b>第3章</b>	<b>智能指针 .....</b>	<b>57</b>
3.1	轻量级指针 .....	58
3.1.1	实现原理分析 .....	58
3.1.2	应用实例分析 .....	61
3.2	强指针和弱指针 .....	62
3.2.1	强指针的实现原理分析 .....	63
3.2.2	弱指针的实现原理分析 .....	69
3.2.3	应用实例分析 .....	75

## 第2篇 Android专用驱动系统

<b>第4章</b>	<b>Logger日志系统 .....</b>	<b>82</b>
4.1	Logger日志格式 .....	83
4.2	Logger日志驱动程序 .....	84
4.2.1	基础数据结构 .....	85
4.2.2	日志设备的初始化过程 .....	86
4.2.3	日志设备文件的打开过程 .....	91
4.2.4	日志记录的读取过程 .....	92
4.2.5	日志记录的写入过程 .....	96
4.3	运行时库层日志库 .....	101
4.4	C/C++日志写入接口 .....	108
4.5	Java日志写入接口 .....	112
4.6	Logcat工具分析 .....	118
4.6.1	基础数据结构 .....	119

4.6.2	初始化过程	123
4.6.3	日志记录的读取过程	135
4.6.4	日志记录的输出过程	140
<b>第5章</b>	<b>Binder进程间通信系统</b>	<b>152</b>
5.1	Binder驱动程序	153
5.1.1	基础数据结构	154
5.1.2	Binder设备的初始化过程	172
5.1.3	Binder设备文件的打开过程	173
5.1.4	Binder设备文件的内存映射过程	174
5.1.5	内核缓冲区管理	181
5.2	Binder进程间通信库	191
5.3	Binder进程间通信应用实例	196
5.4	Binder对象引用计数技术	204
5.4.1	Binder本地对象的生命周期	205
5.4.2	Binder实体对象的生命周期	209
5.4.3	Binder引用对象的生命周期	212
5.4.4	Binder代理对象的生命周期	217
5.5	Binder对象死亡通知机制	220
5.5.1	注册死亡接收通知	221
5.5.2	发送死亡接收通知	224
5.5.3	注销死亡接收通知	229
5.6	Service Manager的启动过程	232
5.6.1	打开和映射Binder设备文件	234
5.6.2	注册为Binder上下文管理者	235
5.6.3	循环等待Client进程请求	239
5.7	Service Manager代理对象的获取过程	246
5.8	Service组件的启动过程	252
5.8.1	注册Service组件	253
5.8.2	启动Binder线程池	297
5.9	Service代理对象的获取过程	299
5.10	Binder进程间通信机制的Java接口	308
5.10.1	Service Manager的Java代理对象的获取过程	308
5.10.2	Java服务接口的定义和解析	318
5.10.3	Java服务的启动过程	321
5.10.4	Java服务代理对象的获取过程	328
5.10.5	Java服务的调用过程	331
<b>第6章</b>	<b>Ashmem匿名共享内存系统</b>	<b>335</b>
6.1	Ashmem驱动程序	336
6.1.1	基础数据结构	336
6.1.2	匿名共享内存设备的初始化过程	338



6.1.3	匿名共享内存设备文件的打开过程 .....	340
6.1.4	匿名共享内存设备文件的内存映射过程 .....	342
6.1.5	匿名共享内存块的锁定和解锁过程 .....	344
6.1.6	匿名共享内存块的回收过程 .....	352
6.2	运行时库cutils的匿名共享内存访问接口 .....	353
6.3	匿名共享内存的C++访问接口 .....	357
6.3.1	MemoryHeapBase .....	357
6.3.2	MemoryBase .....	367
6.3.3	应用实例 .....	372
6.4	匿名共享内存的Java访问接口 .....	378
6.4.1	MemoryFile .....	378
6.4.2	应用实例 .....	383
6.5	匿名共享内存的共享原理 .....	394

### 第3篇 Android应用程序框架

<b>第7章</b>	<b>Activity组件的启动过程 .....</b>	<b>400</b>
7.1	Activity组件应用实例 .....	400
7.2	根Activity组件的启动过程 .....	406
7.3	子Activity组件在进程内的启动过程 .....	440
7.4	子Activity组件在新进程中的启动过程 .....	448
<b>第8章</b>	<b>Service组件的启动过程 .....</b>	<b>451</b>
8.1	Service组件应用实例 .....	451
8.2	Service组件在新进程中的启动过程 .....	459
8.3	Service组件在进程内的绑定过程 .....	471
<b>第9章</b>	<b>Android系统广播机制 .....</b>	<b>494</b>
9.1	广播机制应用实例 .....	495
9.2	广播接收者的注册过程 .....	501
9.3	广播的发送过程 .....	509
<b>第10章</b>	<b>Content Provider组件的实现原理 .....</b>	<b>532</b>
10.1	Content Provider组件应用实例 .....	533
10.1.1	ArticlesProvider .....	533
10.1.2	Article .....	543
10.2	Content Provider组件的启动过程 .....	558
10.3	Content Provider组件的数据共享原理 .....	581

10.3.1	数据共享模型 .....	581
10.3.2	数据传输过程 .....	584
10.4	Content Provider组件的数据更新通知机制 .....	604
10.4.1	注册内容观察者 .....	605
10.4.2	发送数据更新通知 .....	611
<b>第11章</b>	<b>Zygote和System进程的启动过程 .....</b>	<b>619</b>
11.1	Zygote进程的启动脚本 .....	619
11.2	Zygote进程的启动过程 .....	622
11.3	System进程的启动过程 .....	630
<b>第12章</b>	<b>Android应用程序进程的启动过程 .....</b>	<b>638</b>
12.1	应用程序进程的创建过程 .....	638
12.2	Binder线程池的启动过程 .....	647
12.3	消息循环的创建过程 .....	649
<b>第13章</b>	<b>Android应用程序的消息处理机制 .....</b>	<b>653</b>
13.1	创建线程消息队列 .....	653
13.2	线程消息循环过程 .....	658
13.3	线程消息发送过程 .....	663
13.4	线程消息处理过程 .....	668
<b>第14章</b>	<b>Android应用程序的键盘消息处理机制 .....</b>	<b>675</b>
14.1	键盘消息处理模型 .....	675
14.2	InputManager的启动过程 .....	678
14.2.1	创建InputManager .....	678
14.2.2	启动InputManager .....	681
14.2.3	启动InputDispatcher .....	683
14.2.4	启动InputReader .....	685
14.3	InputChannel的注册过程 .....	696
14.3.1	创建InputChannel .....	697
14.3.2	注册Server端InputChannel .....	705
14.3.3	注册系统当前激活的应用程序窗口 .....	709
14.3.4	注册Client端InputChannel .....	714
14.4	键盘消息的分发过程 .....	717
14.4.1	InputReader获得键盘事件 .....	718
14.4.2	InputDispatcher分发键盘事件 .....	725
14.4.3	系统当前激活的应用程序窗口获得键盘消息 .....	735
14.4.4	InputDispatcher获得键盘事件处理完成通知 .....	751

14.5	InputChannel的注销过程.....	754
14.5.1	销毁应用程序窗口.....	755
14.5.2	注销Client端InputChannel.....	764
14.5.3	注销Server端InputChannel.....	766
<b>第15章</b>	<b>Android应用程序线程的消息循环模型 .....</b>	<b>772</b>
15.1	应用程序主线程消息循环模型.....	773
15.2	与界面无关的应用程序子线程消息循环模型.....	774
15.3	与界面相关的应用程序子线程消息循环模型.....	777
<b>第16章</b>	<b>Android应用程序的安装和显示过程 .....</b>	<b>786</b>
16.1	应用程序的安装过程 .....	786
16.2	应用程序的显示过程 .....	822

---

## 读者服务

轻松注册成为博文视点社区用户（[www.broadview.com.cn](http://www.broadview.com.cn)），扫码直达本书页面。

- **下载资源**：本书所提供的示例代码及资源文件，均可在[下载资源处](#)下载。
- **提交勘误**：您对书中内容的修改意见可在[提交勘误处](#)提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- **交流互动**：在页面下方[读者评论处](#)留下您的疑问或观点，与我们和其他读者一同学习交流。

页面入口：<http://www.broadview.com.cn/32521>



# 第1篇 初识Android系统

Linux内核的鼻祖Linus在回答一个Minix系统<sup>1</sup>的问题时，第一句话便是“Read The Fucking Source Code”。这句话虽然颇有调侃的味道，但是它道出了阅读源代码的重要性。由于Android系统的源代码是开放的，因此，认识Android系统的最好方法莫过于阅读它的源代码。

我们知道，Android系统是基于Linux内核来开发的，因此，在阅读它的源代码之前，需要掌握Linux内核的基础知识。有了Linux内核的基础知识之后，就可以下载Android源代码来阅读。阅读Android源代码时，宜采用动静结合的方法。所谓静，就是研究和思考源代码的实现；而所谓动，就是通过运行系统来证实自己对源代码的研究和思考。运行系统并不是单纯地将系统运行起来就可以了，还需要亲自动手编写一些应用程序来验证系统的行为。因此，在第1章中，我们首先介绍几本Linux内核的经典参考书籍，然后介绍如何搭建Android源代码工程环境，最后介绍如何在Android源代码工程环境中开发应用程序。

在阅读Android源代码之前，首先需要从整体上了解Android系统的架构。Android系统大致可以划分为五个层次，它们分别是Linux内核层、硬件抽象层、运行时库层、应用程序框架层和应用程序层。其中，硬件抽象层从实现到使用上涉及这五个层次，因此，在第2章中，我们通过具体的实例来介绍Android系统的硬件抽象层，以便对Android系统的实现层次有一个感性的认识。

虽然Android应用程序是使用Java语言来开发的，但是在Android应用程序框架层中，有相当多的一部分代码使用C++语言来编写，在阅读这些C++代码时，经常会碰到智能指针。Android系统的智能指针对于C++开发者来说，是比较容易理解的；但是对于Java开发者来说，就比较苦涩了。因此，在第3章中，我们继续通过具体的实例来分析Android系统的智能指针实现原理。

---

<sup>1</sup> Minix是一个类UNIX的操作系统，见<http://en.wikipedia.org/wiki/MINIX>。

# 第 1 章

## 准备知识

Android系统的源代码非常庞大和复杂，我们不能贸然进入，否则很容易在里面迷失方向，进而失去研究它的信心。为了有条不紊地对Android系统的源代码进行全面、深入、细致的分析，我们需要准备一些参考书籍，搭建好Android系统源代码工程环境，以及对Android系统有一个感性认识。

### 1.1 Linux内核参考书籍

在阅读分析Android系统的源代码时，经常会碰到诸如管道（pipe）、套接字（socket）和虚拟文件系统（VFS）等知识。此外，Android系统通过模块的形式在Linux内核中增加了一些专用的驱动程序，如Logger日志驱动程序、Binder进程间通信驱动程序，以及Ashmem匿名共享内存驱动程序等，这些都是Linux内核的基础知识，涉及进程、内存管理等内容。由于本书的重点是分析Android系统的源代码，因此，下面推荐四本介绍Linux内核基础知识的经典书籍。

#### （1）*Linux Kernel Development*

这本书的作者是Robert Love，目前最新的版本是第3版。它对Linux内核的设计原理和实现思路提供了一个总览视图，并且对Linux内核的各个子系统的设计目标进行了清晰的描述，非常适合初学者阅读。从软件工程的角度来看，这本书相当于Linux内核的概要设计文档。

#### （2）*Understanding the Linux Kernel*

这本书的作者是Daniel P. Bovet和Marco Cesati，目前最新的版本是第3版。它对Linux内核的实现提供了更多的细节，详细地描述了内核开发中用到的各种重要数据结构、算法以及编程技巧等，非常适合中、高级读者阅读。从软件工程的角度来看，这本书相当于Linux内核的详细设计文档。

#### （3）*Linux Device Drivers*

这本书的作者是Jonathan Corbet, Alessandro Rubini和Greg Kroah-Hartman，目前最新的版本是第3版。它更加注重于实际操作，详细地讲解了Linux内核驱动程序的实现原理，对分析Android系统的专用驱动模块有非常大的帮助。

#### （4）《Linux内核源代码情景分析》

这本书的作者是毛德操和胡希明，是中国人自己编写的一本经典的Linux内核书籍。它最大的特点是从使用情景出发，对Linux内核作了详细的分析，为读者在Linux内核源代码的汪洋大海中指明方向。

## 1.2 Android应用程序参考书籍

分析Android系统的源代码时，应该带着问题或者目标。要把问题或者目标挖掘出来，最好的方法就是在Android系统中编写应用程序。通过编写Android应用程序，我们可以知道系统提供了哪些功能，并且如何去使用这些功能，进而激发我们去了解这些功能是如何实现的。这样我们就可以获得分析Android系统源代码所需要的问题或者目标。下面推荐两本介绍Android应用程序开发的书籍。

(1) *Professional Android 2 Application Development*

(2) 《Google Android SDK开发范例大全》

这两本书的特点是都使用了大量的例子来描述如何使用Android SDK来开发Android应用程序。我们可以根据实际情况来熟悉Android应用程序的开发方法，主要掌握Android应用程序四大组件（Activity、Service、Broadcast Receiver和Content Provider）的用法。在学习的过程中，如果遇到其他问题，还可以参考官方API文档，其网址为：

<http://developer.android.com/index.html>

## 1.3 下载、编译和运行Android源代码

目前，官方支持Android源代码在Ubuntu和Mac系统上编译。本书推荐使用Ubuntu系统。Ubuntu系统是一个广受称道的Linux发行版本，它具有强大的软件包管理系统，并且简单易用，官方下载地址为：<http://www.ubuntu.com/>。

不同版本的Android源代码对Ubuntu系统版本的要求不一样，具体如下。

- Android 6.0 (Marshmallow) ~ AOSP master: Ubuntu 14.04 (Trusty)
- Android 2.3.x (Gingerbread) ~ Android 5.x (Lollipop): Ubuntu 12.04 (Precise)
- Android 1.5 (Cupcake) ~ Android 2.2.x (Froyo): Ubuntu 10.04 (Lucid)

本书分析的Android源代码版本为2.3.x，因此推荐安装Ubuntu 12.04。另外，如果要安装Ubuntu 14.04，则需要安装64位版本。

如果读者习惯使用Windows系统，就可以考虑先在Windows上安装虚拟机，然后在虚拟机上安装Ubuntu系统。虚拟机推荐使用VMware，官方网站为：<http://www.vmware.com/>。

安装VMware时，最好选择6.0以上的版本，因为较旧版本的VMware在网络连接支持上比较差，而我们在下载Android源代码时，是必须要联网的。

安装好Ubuntu系统之后，还需要安装JDK及其他的依赖包，然后才可以正常下载、编译和运行Android源代码。接下来我们就介绍它们的安装方法<sup>1</sup>。

### 1. JDK

不同版本的Android源代码对JDK版本的要求也不一样，具体如下。

- AOSP master: OpenJDK 8
- Android 5.x (Lollipop) ~ Android 6.0 (Marshmallow): OpenJDK 7
- Android 2.3.x (Gingerbread) ~ Android 4.4.x (KitKat): Java JDK 6
- Android 1.5 (Cupcake) ~ Android 2.2.x (Froyo): Java JDK 5

<sup>1</sup> 随着Android源代码版本的迭代，对于Ubuntu版本、JDK及其他依赖包都会发生变化，最新的安装指令可以参考官方网站：<https://source.android.com/source/initializing>。

Java JDK 5和6的官方网站为：<http://www.oracle.com>。

该网站为Linux提供的JDK有bin和rpm.bin两种包，推荐下载bin包。以JDK 6为例，假设下载文件为jdk-6u45-linux-x64.bin，保存在~/Downloads目录下，执行以下命令进行安装。

```
USER@MACHINE:~$ sudo cp ~/Downloads/jdk-6u45-linux-x64.bin /usr/java/
USER@MACHINE:~$ cd /usr/java
USER@MACHINE:/usr/java$ chmod a+x ./jdk-6u45-linux-x64.bin
USER@MACHINE:/usr/java$ ./jdk-6u45-linux-x64.bin
```

安装完成后，可以通过update-alternatives工具管理JDK，如下所示。

```
USER@MACHINE:~$ sudo update-alternatives --install /usr/bin/java java /usr/java/jdk1.6.0_45/bin/
java 1061
USER@MACHINE:~$ sudo update-alternatives --install /usr/bin/javac javac /usr/java/jdk1.6.0_45/
bin/java 1061
```

如果电脑上也安装了其他版本的JDK，则可以通过以下命令选择所需要的版本。

```
USER@MACHINE:~$ sudo update-alternatives --config java
There are 5 choices for the alternative java (providing /usr/bin/java).
  Selection    Path                                                    Priority    Status
-----
  0            /usr/java/jdk1.8.0_131/jre/bin/java                    1082      auto mode
  1            /usr/java/jdk1.6.0_45/bin/java                          1061      manual mode
  * 2          /usr/java/jdk1.8.0_131/bin/java                        1080      manual mode
  3            /usr/java/jdk1.8.0_131/jre/bin/java                    1082      manual mode
  4            /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java         1071      manual mode
  5            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java         1081      manual mode
Press <enter> to keep the current choice[*], or type selection number: 1
```

输入数字“1”，就可以将系统的java命令指向JDK 6安装目录下的Java文件。

```
USER@MACHINE:~$ sudo update-alternatives --config javac
There are 4 choices for the alternative javac (providing /usr/bin/javac).
  Selection    Path                                                    Priority    Status
-----
  0            /usr/lib/jvm/java-8-openjdk-amd64/bin/javac             1081      auto mode
  1            /usr/java/jdk1.6.0_45/bin/javac                         1061      manual mode
  * 2          /usr/java/jdk1.8.0_131/bin/javac                       1080      manual mode
  3            /usr/lib/jvm/java-7-openjdk-amd64/bin/javac             1071      manual mode
  4            /usr/lib/jvm/java-8-openjdk-amd64/bin/javac             1081      manual mode
Press <enter> to keep the current choice[*], or type selection number: 1
```

输入数字“1”，就可以将系统的javac命令指向JDK 6安装目录下的javac文件。

OpenJDK 7和8的下载地址分别如下：

<http://archive.ubuntu.com/ubuntu/pool/universe/o/openjdk-7/>

<http://archive.ubuntu.com/ubuntu/pool/universe/o/openjdk-8/>

下载得到一个deb文件，可以使用dpkg命令进行安装。例如，假设下载OpenJDK 8，文件名为openjdk-8-dbg\_8u45-b14-1\_amd64.deb，保存在~/Downloads目录下，执行以下命令进行安装。

```
USER@MACHINE:~$ sudo apt-get update
USER@MACHINE:~$ sudo dpkg --install ~/Downloads/openjdk-8-dbg_8u45-b14-1_amd64.deb
```

在安装过程中如果提示依赖缺失，则可以执行以下命令进行修复安装。

```
USER@MACHINE:~$ sudo apt-get -f install
```

## 2. 其他依赖包

Ubuntu 14.04的依赖包可以执行以下命令进行安装。

```
USER@MACHINE:~$ sudo apt-get install git-core gnupg flex bison gperf build-essential \  
zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \  
lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache \  
libgll-mesa-dev libxml2-utils xsltproc unzip
```

Ubuntu 12.04的依赖包可以执行以下命令进行安装。

```
USER@MACHINE:~$ sudo apt-get install git gnupg flex bison gperf build-essential \  
zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev \  
libx11-dev:i386 libreadline6-dev:i386 libgll-mesa-glx:i386 \  
libgll-mesa-dev g++-multilib mingw32 tofrodos \  
python-markdown libxml2-utils xsltproc zlib1g-dev:i386  
USER@MACHINE:~$ sudo ln -s /usr/lib/i386-linux-gnu/mesa/libGL.so.1 /usr/lib/i386-linux-gnu/  
libGL.so
```

Ubuntu 10.04的依赖包可以执行以下命令进行安装。

```
USER@MACHINE:~$ sudo apt-get install git gnupg flex bison gperf build-essential \  
zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs \  
x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev \  
libgll-mesa-dev g++-multilib mingw32 tofrodos python-markdown \  
libxml2-utils xsltproc
```

在上述依赖包中，包含了Git工具。Git是一种分布式的源代码管理工具，它可以有效、高速地对项目源代码进行版本管理<sup>1</sup>。Android源代码就是采用它来管理的。

### 1.3.1 下载Android源代码

为了方便开发者下载Android源代码，Google提供了一个repo工具。这个工具实际上是一个脚本文件，里面封装了用来下载Android源代码所需要的git命令。它的下载和安装方法如下：

```
USER@MACHINE:~$ wget https://dl-ssl.google.com/dl/googlesource/git-repo/repo  
USER@MACHINE:~$ chmod a+x repo  
USER@MACHINE:~$ sudo mv repo /bin/
```

安装好repo工具之后，我们就可以创建一个空目录，然后进入到这个目录中执行repo命令来下载Android源代码了。

```
USER@MACHINE:~$ mkdir Android  
USER@MACHINE:~$ cd Android  
USER@MACHINE:~/Android$ repo init -u https://android.googlesource.com/platform/manifest  
USER@MACHINE:~/Android$ repo sync
```

下载的过程可能会比较漫长，这取决于网络连接速度，期间还可能会碰到网络中断的现象，这时候只需要重复执行repo sync命令就可以继续下载了。

上述命令下载的是主线上的Android源代码，即最新版本的Android源代码。一般来说，主线上的源代码是正在开发的版本，它是不稳定的，编译和运行时都可能会遇到问题。如果想下载稳定的版本，就需要选择某一个支线上的代码。例如，如果我们想下载Android 2.3.1版本的代码，就可以在运行repo init命令时指定-b选项。

```
USER@MACHINE:~/Android$ repo init -u https://android.googlesource.com/platform/manifest -b  
android-2.3.1_r1
```

在本书接下来的内容中，如果没有特别声明，我们所分析的Android源代码都是基于Android 2.3版本的，并且位于~/Android目录中。

1 Git工具的使用方法可以参考官方网站：<http://git-scm.com/>。



### 1.3.2 编译Android源代码

要编译Android源代码，只需在Android源代码目录下执行make命令就可以了。

```
USER@MACHINE:~/Android$ make
```

第一次编译Android源代码时，花费的时间会比较长，同时也可能会遇到各种各样的问题，这时候一般都可以通过搜索引擎来找到解决方案。例如，如果我们是32位机器上编译主线上的Android源代码，则会碰到下面这个错误提示。

```
build/core/main.mk:76: *****
build/core/main.mk:77: You are attempting to build on a 32-bit system.
build/core/main.mk:78: Only 64-bit build environments are supported beyond froyo/2.2.
build/core/main.mk:79: *****
```

这时候可以使用关键词“You are attempting to build on a 32-bit system”在搜索引擎上找到解决方案。原来，主线上的Android源代码默认只能在64位的机器上编译，如果在32位的机器上编译，就会出现上述错误提示。如果我们仍然想在32位的机器上编译Android源代码，就可以按照下面方法来修改编译脚本。

(1) 打开build/core/main.mk文件，并且找到下面内容。

```
ifeq ($(BUILD_OS),linux)
    build_arch := $(shell uname -m)
    ifneq (64,$(findstring 64,$(build_arch)))
        $(warning *****)
        $(warning You are attempting to build on a 32-bit system.)
        $(warning Only 64-bit build environments are supported beyond froyo/2.2.)
```

将第3行修改为

```
ifneq (i686,$(findstring i686,$(build_arch)))
```

(2) 打开external/clearsilver/cgi/Android.mk、external/clearsilver/cs/Android.mk、external/clearsilver/java-jni/Android.mk和external/clearsilver/util/Android.mk这四个文件，并且找到下面内容：

```
# This forces a 64-bit build for Java6
LOCAL_CFLAGS += -m64
LOCAL_LDFLAGS += -m64
```

将后面两行修改为

```
LOCAL_CFLAGS += -m32
LOCAL_LDFLAGS += -m32
```

经过这样的修改之后，在32位的机器上编译Android源代码产生的问题就可以解决了。

编译成功后，可以看到下面的输出。

```
Target system fs image: out/target/product/generic/obj/PACKAGING/systemimage_intermediates/
system.img
Install system fs image: out/target/product/generic/system.img
Target ram disk: out/target/product/generic/ramdisk.img
Target userdata fs image: out/target/product/generic/userdata.img
Installed file list: out/target/product/generic/installed-files.txt
```

编译结果输出目录为out/target/product/\$(TARGET\_PRODUCT)，其中，TARGET\_PRODUCT是一个