

基于 2.3 *Gingerbread* 版本，
深入剖析Android内核



Android 内核剖析

◎柯元旦 著◎

Android 内核剖析

◎柯元旦 著◎

电子工业出版社
Publishing House of Electronics Industry
北京·BEIJING

内 容 简 介

本书详细分析了 Android 内核的内部机制，包括窗口管理系统、Activity 管理系统、输入法框架、编译系统等，为 Android 内核定制及高级应用程序开发提供技术参考。

书中提及“附图”请到 <http://www.broadview.com.cn/14398> 下载。

本书适合于所有 Android 相关的工程师及产品经理，还可作为相关培训机构的教材。

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 内核剖析/柯元旦著. —北京: 电子工业出版社, 2011.9
ISBN 978-7-121-14398-4

I. ①A… II. ①柯… 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 (2011) 第 168438 号

策划编辑: 符隆美

责任编辑: 高洪霞

特约编辑: 赵树刚

印 刷: 北京天宇星印刷厂

装 订: 三河市皇庄路通装订厂

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: 860×1092 1/16 印张: 38.5 字数: 1109 千字

印 次: 2011 年 12 月第 2 次印刷

印 数: 5001~9000 册 定价: 79.90 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。

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

服务热线：(010) 88258888。



“内核剖析”乍一听起来挺吓唬人的，但这个词语存在两个问题，第一个是什么才能称为内核？另一个是“谁”才有能力或者有机会写一本“内核剖析”的书？

本书之所以在前言中提出这个问题，就是为了不吓唬大家，并给大家一种信心，相信自己有能力理解本书的内容。

首先来回答第一个问题，什么才能称为内核？大家都知道，Linux 内核的本质包含了线程调度、内存管理及输入/输出管理，那么请问 Windows 操作系统的内核是什么呢？我们常说，苹果的操作系统 Mac OS X 的内核是基于 UNIX 的，那么可以说 Mac OS 的内核是 UNIX 吗？

如果仅从线程调度、内存管理，以及输入/输出的角度来区分 Windows 和 Mac OS 系统的话，能很明显地感觉到缺少点什么，那就是图形用户接口（GUI），Android、Windows、Mac OS 三者的操作方式完全不同，因此，对于图形操作系统而言，本人倾向于将 GUI 也划归到内核的范畴，这也就是为什么本书使用“内核”作为标题的原因。本书所谓的“内核剖析”的核心也正在于 Android 所设计的 GUI 框架的内部原理。Android 操作系统是基于 Linux 实现的，本书并不是去剖析 Linux。

下面再来回答第二个问题，即“谁”才有能力或者有机会写一本“内核剖析”的书？如果有人告诉你，一个非微软公司的技术人员写了一本 Windows 操作系统内核剖析的书，你信吗？反正我不信，原因是，没有阅读过 Windows 内核源码的人是不可能写出这样的书的，幸运的是 Android 的源码是开放的。可是源码开放就一定能写这样一本书吗？

在本书截稿时，我未曾见过一本真正分析 Android 内核的书，大多数书籍都是关于 Android SDK 应用开发的。在过去的工作经历中，常常遇到一些同事，由于对 Android 内核不了解，导致在应用程序开发时遇到一些无法解决的问题。遗憾的是，IT 类优秀书籍本来就很少，中文原创的更少，Android 领域的几乎没有，本人之前也写过一本《Android 程序设计》，坦白地讲，当我对 Android 内核彻底剖析后，觉得那本书有“误人子弟”的成分。因此，我才决定要将自己对 Android 的理解分享给更多的读者。

那么，我有可能写出一本真正的“内核剖析”的书吗？

我 2003 年毕业于西安电子科技大学通信工程学院，毕业后与两名同学一起创业，当时我们的目标是做一个“心情播放器”，其本质是一个彩屏多媒体掌上设备，当初想把它做成一个能够根据人的心情自动播放音乐的设备，不谈产品，仅从技术的角度来讲，我们基于美国德州仪器（TI）公司的一款 DSP 处理器完成了“心情播放器”的设计，包括软件和硬件，该软件系统包括支持最多 16 个线程的多线程管理、内存管理、FAT16 文件系统、GUI 子系统，以及一套标准应用程序开发框

架、桌面程序等，在这里要再次感谢同宿舍的陈静军同学，他是我到目前为止见过的写代码最优秀的人，在这个项目中，静军设计了这个操作系统的内核及 GUI 子系统，而我设计了硬件主板、驱动、系统开发框架等，说到这里，如果静军来写一本内核剖析的书，肯定会比我写得更好，在当初设计操作系统前，由于静军还没有加入到我们团队，我才花时间研究了嵌入式操作系统，并设计了一些简单的接口，而当静军加入后，这些工作就由他完成了，因此，从严格意义上讲，我并没有实际编写过操作系统内核代码，只不过从硬件、驱动、系统等不同层面设计了一个系统框架而已。

在这个项目中，一切只是从一颗处理器入手，没有基于任何代码，所有底层代码都是我们编写的，包括汇编和 C 语言程序设计，因此，在这个过程中，我彻底了解了 C 语言如何被编译成汇编代码，以及特定处理器如何影响上层的 C 程序。

当然，这个故事是以失败而告终的，后来我继续从事嵌入式产品设计，包括使用 TI 高性能 DSP 处理器、x86 处理器，ARM 处理器等，不过，仅过了两年时间，又去从事互联网产品的设计，并开始使用 Java、C++、PHP、JavaScript、Erlang 等不同语言进行软件开发，在使用各种语言时，我常常思考这些语言与底层系统的关系，并从编译原理的角度来理解每一种语言，从而能够理解不同语言的运行环境和操作系统的关系。

直到 Android 的诞生，我当时对 Android 的描述是，这是一个把嵌入式系统和互联网应用集合在一起的一个技术。幸运的是这些我都还算熟悉，因此就开始了 Android 的开发，最开始的时候仅仅是应用程序的开发，虽然也常常考虑 Android 底层的问题，但由于没有源码，所以也就没有仔细研究，后来发现，这也是一件好事，因为如果不熟悉上层的开发接口，则很难理解内核的一些概念。

后来，应用层积累得差不多了，源码也开放了，于是我就迫不及待地开始了内核之旅，所有的分析都是基于源码的阅读和测试，中间的过程的确是辛苦的，包括在 Ubuntu 及 Mac OS 上建立编译环境、思考 Android 中的异步调度架构、平衡工作和学习的时间等，早上坐地铁也常常看 Google groups 中关于 Android 的各种问答。不过，每当你明白一个大的架构的关键之处时，也是一件很开心的事情。

谈及以上履历的目的在于启发正在读大学的朋友，一名电子工程师一定要理论、硬件、软件及梦想同时具备，不要把自己区分为“硬件工程师”、“软件工程师”，我们可以称自己为电子工程师或者“梦想家”。另外，学习一定要循序渐进，如果你还不了解微机原理，那么就不要再学习 C 语言，如果你还不了解数字电路，那么就不要再学习微机原理，上层的软件开发需要对底层基础知识的理解，只有这样才能成为一名创造者，并设计出卓越的产品。

多么希望我们中国的大学生在不远的将来也能创造出像 Google、Microsoft、Facebook 这样著名的企业。

内容介绍

本书内容分为五大部分，分别如下：

第 1 部分，基础篇。因为 Android 内核研究必须基于 Unix-Like 的主机系统上，常见的有 Ubuntu

和 Mac OS X，因此，该部分介绍 Linux 的一些基础知识，以及在 Linux 上管理源码的工具 git。

第 2 部分，内核篇。Android 内核的核心就是一套 GUI 系统。该部分主要包含视图的内部工作机制及视图管理器（Window Manager Service）和 Activity 管理器（Activity Manager Service）的内部工作机制。

第 3 部分，系统篇。内核不等于操作系统，Android 是一个操作系统，因此，除了内核之外，还必须定义一套系统架构，比如应用程序的格式定义，以及应用程序如何被安装和卸载、输入法框架等，有时候这部分内容也叫做外壳（Shell）。

第 4 部分，编译篇。Android 相关的源码据说超过 1000 万行，这套源码由众多的子项目组成，因此，联合编译这些子项目就是一个复杂的问题。Android 源码中定义了一套编译框架，该框架可以方便地编译不同类型的子项目，比如一个动态链接库项目、Jar 包项目等。了解该套编译架构后，就可以自由地在源码中添加需要的子项目，并控制系统中已有子项目的编译过程。

第 5 部分，硬件驱动篇。Android 目前最成功的产品当然就是智能手机，但同时由于 Android 开源的特点，也就可以应用于其他一些特定的产品，比如玩具、学习机、税控机、门禁系统等，因此，该部分介绍了一款硬件开源的 Android 开发板卡。本来，该部分内容还包括 OpenGL 框架、多媒体框架及 Android 硬件抽象层（HAL）三方面内容，但由于出版时间原因，暂未包含，本书下一版将包含这些内容。

读者对象

本书适合于五类读者。

第一类，开发过 Android 应用程序的工程师。如果你刚开始接触 Android，那么这本书可能会很难理解，建议去 Android 官方网站用两周的时间学习基本的 Android 应用程序开发，或者去看本人早期创作的《Android 程序设计》一书，但要带着怀疑的态度去读。

第二类，Android 技术相关的产品经理。对于产品经理而言，了解项目的技术难度及技术可行性，将有助于制定产品开发时间表。虽然产品经理不需要详细了解技术如何实现，但起码应该知道产品技术的复杂度。

第三类，有扎实的开发经验，却未曾接触过 Android 的开发人员。系统框架的表面尽管各有千秋，但其内涵却不会差别太大，对于有扎实开发经验的朋友而言，只需要重新了解一下 Android 中的新概念，就能快速地将这些新概念与已有的知识融合起来，这样，便可以节省大量的时间。

第四类，正在基于 iOS 开发的工程师。本人最近正在研究 iOS 的开发，令人惊讶的是 iOS 和 Android 开发框架是如此相似。Object-C 语言和 Java 语言的语法虽然差别较大，但其思想却很相似，包括单继承、动态性、内存回收机制等。iOS 和 Android 的 Framework 也惊人相似，比如都使用 sqlite 进行数据存储，也使用 Preference 进行参数存储；视图系统的 API 接口也类似的地方，都可以使用 OpenGL 进行界面绘制。当然，iOS 和 Android 视图系统还是有一定的差别，比如 iOS 中每一个 View 对象都有两个 Layer，从而可以方便地使用 OpenGL 绘制任何一个 View 对象，而 Android 却只有一

个，所以 Android 的动画效果没有 iOS 那样灵活。遗憾的是 iOS 不是开源的，因此，我们没有机会去了解 iOS 内部的详细机制，不过既然 iOS 和 Android 有这么多相似的地方，那么就可以通过了解 Android 的内核机制去思考 iOS 的一些特性。

第五类，想要编写一个 GUI 子系统的学生。Android 虽然更多地用于手机产品，但其内部的 GUI 子系统的实现却是一种通用的思路，因此，可以完全抛开 Android 的系统特性，而仅仅去研究其 GUI 子系统的实现思路，有了这种思路就可以使用各种语言设计自己想要的 GUI 子系统。

欢迎朋友们与我进一步交流，邮件请发至 keyuandan@googlegroups.com。

作 者



第 1 部分 基础篇

第 1 章 Linux 基础	2
1.1 Linux 文件系统概述	2
1.2 Linux 启动过程	4
1.3 常用 Linux 命令	6
1.4 Shell 脚本备忘	9
1.4.1 获取输入	10
1.4.2 变量定义	10
1.4.3 条件判断	11
1.4.4 while [...]do... done 语句	12
1.4.5 for 循环	13
1.4.6 函数	14
1.4.7 常用内置符号常量	15
1.5 Make 脚本备忘	15
1.5.1 一个简单的 Makefile 文件	16
1.5.2 变量的定义与赋值	18
1.5.3 条件控制语句	18
1.5.4 宏（函数）定义	19
1.5.5 内置符号和变量	22
1.5.6 模板目标（Pattern target）	23
1.5.7 目标特定的变量赋值（Target-specific variable）	24
1.5.8 常用选项	25
第 2 章 Java 基础	26
2.1 类装载器 DexClassLoader	26
2.1.1 DexClassLoader 的调用方法	27

2.1.2	基于类装载器设计一种“插件”架构	29
2.2	JNI 调用机制	32
2.2.1	Java 访问 C	33
2.2.2	C 访问 Java	35
2.2.3	在 C 中使用持久对象	37
2.3	异步消息处理线程	37
2.3.1	实现异步线程的一般思路	38
2.3.2	Android 中异步线程的实现方法	38
第 3 章	Android 源码下载及开发环境配置	44
3.1	Mac 系统的配置	44
3.1.1	硬盘格式的配置	44
3.1.2	port 的用法	46
3.2	在 Linux 中配置 USB 连接	46
3.3	在 Eclipse 中调试 Framework	46
3.3.1	一段防止下载异常的脚本	47
3.3.2	调试 Framework 中的代码	47
第 4 章	使用 git	51
4.1	安装 git	52
4.2	git 仓库管理	52
4.2.1	仓库的组成	52
4.2.2	创建仓库	54
4.2.3	分支管理	55
4.3	git merge 用法	57
4.4	git rebase 用法	58
4.5	git cherry-pick 用法	61
4.6	git reset 用法	62
4.7	恢复到无引用提交	63
4.8	git remote 用法	65
4.9	git 配置	67
4.9.1	基本信息配置	68
4.9.2	merge、diff 工具配置	68
4.9.3	.gitignore 配置	70
4.10	同时使用 git 和 svn	71
4.11	其他 git 常用命令示例	72

4.11.1	git branch	72
4.11.2	git checkout	72
4.11.3	git log	73
4.11.4	git commit --amend	73
4.11.5	git cherry-pick sha-1	73
4.11.6	git merge-base	74
4.11.7	git diff master...dev	74
4.11.8	git revert	75
4.11.9	git diff	75
4.11.10	git rm	75
4.11.11	git tag	76

第 2 部分 内核篇

第 5 章	Binder	78
5.1	Binder 框架	78
5.2	设计 Servier 端	80
5.3	Binder 客户端设计	81
5.4	使用 Service 类	82
5.4.1	获取 Binder 对象	82
5.4.2	保证包裹内参数顺序 aidl 工具的使用	83
5.5	系统服务中的 Binder 对象	88
5.5.1	ServiceManager 管理的服务	88
5.5.2	理解 Manager	90
第 6 章	Framework 概述	92
6.1	Framework 框架	92
6.1.1	服务端	92
6.1.2	客户端	93
6.1.3	Linux 驱动	94
6.2	APK 程序的运行过程	94
6.3	客户端中的线程	94
6.4	几个常见问题	95
6.4.1	Acitivity 之间如何传递消息 (数据)	95
6.4.2	窗口相关的概念	96

第 7 章 理解 Context	98
7.1 Context 是什么	98
7.2 一个应用程序中包含多少个 Context 对象	99
7.3 Context 相关类的继承关系	99
7.4 创建 Context	100
7.4.1 Application 对应的 Context	101
7.4.2 Activity 对应的 Context	102
7.4.3 Service 对应的 Context	103
7.4.4 Context 之间的关系	104
第 8 章 创建窗口的过程	106
8.1 窗口的类型	106
8.2 token 变量的含义	108
8.2.1 Activity 中的 mToken	108
8.2.2 Window 中的 mAppToken	109
8.2.3 WindowManager.LayoutParams 中的 token	109
8.2.4 View 中的 token	110
8.3 创建应用窗口	111
8.4 创建子窗口	120
8.4.1 Dialog 的创建	123
8.4.2 PopupWindow 的创建	126
8.4.3 ContextMenu 的创建	128
8.4.4 OptionMenu 的创建	132
8.5 系统窗口 Toast 的创建	136
8.5.1 Toast 调用流程	137
8.5.2 Toast 添加窗口	139
8.6 创建窗口示例	140
第 9 章 Framework 的启动过程	142
9.1 Framework 运行环境综述	142
9.2 Dalvik 虚拟机相关的可执行程序	143
9.2.1 dalvikvm	144
9.2.2 dvz	144
9.2.3 app_process	145
9.3 zygote 的启动	147
9.3.1 在 init.rc 中配置 zygote 启动参数	147

9.3.2	启动 Socket 服务端口	148
9.3.3	加载 preload-classes	151
9.3.4	加载 preload-resources	152
9.3.5	使用 fork 启动新的进程	152
9.4	SystemService 进程的启动	155
9.4.1	启动各种系统服务线程	156
9.4.2	启动第一个 Activity	158
第 10 章	AmS 内部原理	160
10.1	Activity 调度机制	160
10.1.1	几个重要概念	161
10.1.2	AmS 中的一些重要调度相关变量	163
10.1.3	startActivity()的流程	165
10.1.4	stopActivityLocked()停止 Activity	183
10.1.5	按“Home”键回到桌面的过程	186
10.1.6	按“Back”键回到上一个 Activity	187
10.1.7	长按“Home”键	189
10.1.8	Activity 生命期的代码含义	190
10.2	内存管理	192
10.2.1	关闭而不退出	192
10.2.2	Android 与 Linux 的配合	194
10.2.3	各种关闭程序的过程	196
10.2.4	释放内存详解	197
10.3	对 AmS 中数据对象的理解	211
10.3.1	常见的对象操作	212
10.3.2	理解 Activity	213
10.3.3	Android 多进程吗，是同时在运行多个应用程序吗	213
10.4	ActivityGroup 的内部机制	214
10.4.1	TabActivity 使用时的类关系结构	215
10.4.2	LocalActivityManager 的内部机制	217
10.4.3	ActivityGroup 内部的 Activity 生命期控制	220
第 11 章	从输入设备中获取消息	221
11.1	Android 消息获取过程概述	221
11.2	与消息处理相关的源码文件分布	223
11.3	创建 InputDispatcher 线程	226

11.4	把窗口信息传递给 InputDispatcher 线程	227
11.5	创建 InputChannel	229
11.6	在 WmS 中注册 InputChannel	232
11.7	在客户进程中注册 InputChannel	233
11.8	WmS 中处理消息的时机	234
11.9	客户窗口获取消息的时机	235
第 12 章	屏幕绘图基础	237
12.1	绘制屏幕的软件架构	237
12.2	Java 客户端绘制调用过程	239
12.3	C 客户端绘制过程	241
12.4	Java 客户端绘制相关类的关系	244
第 13 章	View 工作原理	247
13.1	导论	247
13.2	用户消息类型	249
13.2.1	按键消息	249
13.2.2	触摸消息	250
13.3	按键消息派发过程	252
13.3.1	KeyEvent.DispatcherState 中的长按监测	252
13.3.2	按键消息总体派发过程	254
13.3.3	根视图内部派发过程	256
13.3.4	Activity 内部派发过程	257
13.3.5	View 类内部的 onKeyDown()和 onKeyUp()	260
13.3.6	Activity 中的 onKeyDown()和 onKeyUp()	261
13.3.7	PhoneWindow 内部消息派发过程	262
13.4	按键消息在 WmS 中的派发过程	263
13.5	触摸消息派发过程	266
13.5.1	触摸消息总体派发过程	266
13.5.2	根视图内部消息派发过程	267
13.5.3	ViewGroup 内部消息派发过程	268
13.5.4	各种消息监测的基本实现方法	271
13.5.5	View 内默认消息派发过程	272
13.6	导致 View 树重新遍历的时机	274
13.6.1	状态的分类	274
13.6.2	导致 View 树重新遍历的总体诱因图	275

13.6.3	refreshDrawableList()	276
13.6.4	onFocusedChanged()	278
13.6.5	ensureTouchMode()	279
13.6.6	setVisibility()	282
13.6.7	setEnabled()	284
13.6.8	setSelected()	285
13.6.9	invalidate()	286
13.6.10	requestFocus()	290
13.6.11	requestLayout()	292
13.7	遍历 View 树 performTraversals()的执行过程	293
13.8	计算视图大小 (measure) 的过程	296
13.8.1	measure 内部设计思路	297
13.8.2	ViewGroup 中的 measureChildWithMargins()	301
13.8.3	LinearLayout 中的 onMeasure()过程举例	304
13.9	布局 (layout) 过程	308
13.9.1	layout 过程的设计思路	308
13.9.2	LinearLayout 中 onLayout()内部过程	309
13.9.3	TextView 中 gravity 与 layout 的关系	311
13.10	绘制 (draw) 过程	313
13.10.1	视图中可绘制的元素	313
13.10.2	绘制过程的设计思路	314
13.10.3	ViewRoot 中 draw()的内部流程	315
13.10.4	View 类中 draw()函数内部流程	318
13.10.5	ViewGroup 类中绘制子视图 dispatchDraw()内部流程	322
13.10.6	ViewGroup 类中 drawChild()过程	325
13.10.7	绘制滚动条	328
13.11	动画的绘制	331
13.11.1	动画的设计思路	332
13.11.2	ViewGroup 类中 drawChild()函数中视图动画绘制过程	334
13.11.3	ViewGroup 中 dipsatchDraw()中布局动画绘制流程	337
第 14 章	WmS 工作原理	340
14.1	概述	340
14.1.1	窗口的定义	340
14.1.2	窗口管理要解决的核心问题	341
14.1.3	解决核心问题所使用的相关的变量列表	343

14.1.4	几个操作的概念	346
14.1.5	什么是 Policy, 以及其与 WmS 的关系	346
14.1.6	WmS 接口结构	347
14.2	WmS 主要内部类	348
14.2.1	表示窗口的数据类	348
14.2.2	DimAnimator	348
14.2.3	FadeInOutAnimation	349
14.2.4	InputMonitor 类	350
14.2.5	PolicyThread	351
14.2.6	Session	352
14.2.7	Watermark	353
14.2.8	WMThread	354
14.3	窗口的创建和删除	355
14.3.1	创建窗口的时机和过程	355
14.3.2	assignLayersLocked()的执行过程	360
14.3.3	addWindowToListInOrderLocked()的执行过程	362
14.3.4	删除窗口的时机	364
14.3.5	删除窗口的过程	366
14.3.6	removeWindowInnerLocked()	367
14.4	计算窗口的大小	371
14.4.1	描述窗口尺寸的变量	371
14.4.2	窗口大小的变化过程	372
14.4.3	Policy 中 layoutWindowLw()的执行过程	375
14.4.4	输入法窗口如何影响应用窗口的大小	378
14.5	切换窗口	379
14.5.1	切换要解决的问题	379
14.5.2	InputManager 和 WmS 的接口	381
14.5.3	AmS 与 WmS 的接口	383
14.5.4	从 A 到 B 的切换	387
14.5.5	从 B 回到 A 的过程	390
14.5.6	A 中长按 “Home” 键切换到 B	391
14.5.7	setAppVisiblity()与销毁 Surface	393
14.5.8	computeFocusedWindowLocked()	396
14.6	performLayoutAndPlaceSurfacesLockedInner()的执行过程	398
14.6.1	总体过程	399

14.6.2	第一大步骤：计算窗口的大小	401
14.6.3	第二大步骤：计算窗口的可视状态	401
14.6.4	第三大步骤：通知 SurfaceFlinger 进行窗口重绘	404
14.7	窗口动画	406
14.8	屏幕旋转及 Configuration 的变化过程	409

第 3 部分 系统篇

第 15 章	资源访问机制	414
15.1	定义资源	414
15.2	存储资源	415
15.3	styleable、style、attr、theme 的意义	417
15.4	AttributeSet 与 TypedArray 类	420
15.5	获取 Resources 的过程	425
15.5.1	通过 Context 获取	425
15.5.2	通过 PackageManager 获取	429
15.6	Framework 资源	431
15.6.1	加载和读取	432
15.6.2	添加	434
15.6.3	实现真正主题切换的两种思路	436
第 16 章	程序包管理 (Package Manager Service)	439
16.1	包管理概述	439
16.2	packages.xml 文件格式	442
16.2.1	last-platform-version 标签	443
16.2.2	permissions 标签	443
16.2.3	cert 标签	444
16.2.4	sigs 标签	444
16.2.5	perms 标签	444
16.2.6	package 标签	444
16.2.7	shared-user 标签	445
16.3	包管理服务的启动过程	446
16.3.1	各主要功能类的关系	446
16.3.2	PmS 主体启动过程	448
16.3.3	readPermission()内部过程	450

16.3.4	mSettings.readLP()	452
16.3.5	scanPackageLI()内部过程	454
16.3.6	mSettings.writeLP()	455
16.4	应用程序的安装和卸载	455
16.4.1	各主要功能类关系	456
16.4.2	应用程序安装过程	457
16.4.3	应用程序的卸载过程	461
16.5	intent 匹配框架	463
16.5.1	主要功能类的关系	463
16.5.2	主体调用过程	465
第 17 章	输入法框架	467
17.1	输入法框架组成概述	468
17.2	输入法中各 Binder 对象的创建过程	469
17.2.1	InputConnection	469
17.2.2	IInputMethodClient	471
17.2.3	InputMethodSession	472
17.2.4	InputMethod	475
17.3	输入法主要操作过程	477
17.3.1	输入法相关模块的启动过程	477
17.3.2	切换输入法	478
17.3.3	启动输入法	480
17.3.4	显示输入法	485
17.3.5	输入法操作过程中的重要变量总结	489
17.4	输入法窗口内部的显示过程	490
17.4.1	IMS 中的 showWindow()的内部执行过程	491
17.4.2	标准布局的 IMS	496
17.4.3	自定义布局的 IMS	502
17.5	向编辑框传递字符	503
17.6	输入法相关源码清单	504

第 4 部分 编译篇

第 18 章	Android 编译系统	508
18.1	Android 源码文件结构	509