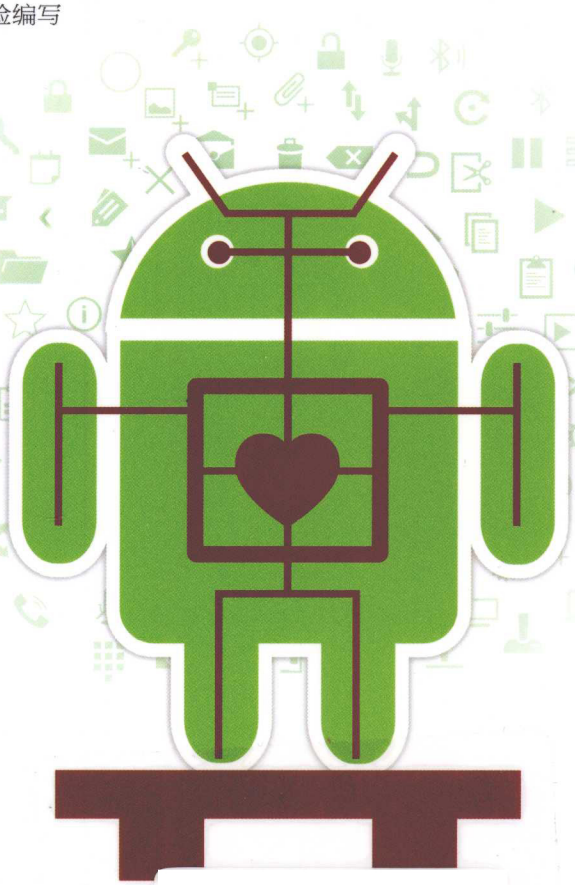
Broadview
www.broadview.com.cn

◎ 用框图描述每个子系统，区分BSP和公用部分
◎ 比较几个有代表性的硬件平台的实现
◎ 比较Android的不同版本
◎ 展示Android系统设计的核心思路
◎ 列出相关代码路径
◎ 根据实际开发经验编写



Android

板级支持与硬件相关子系统

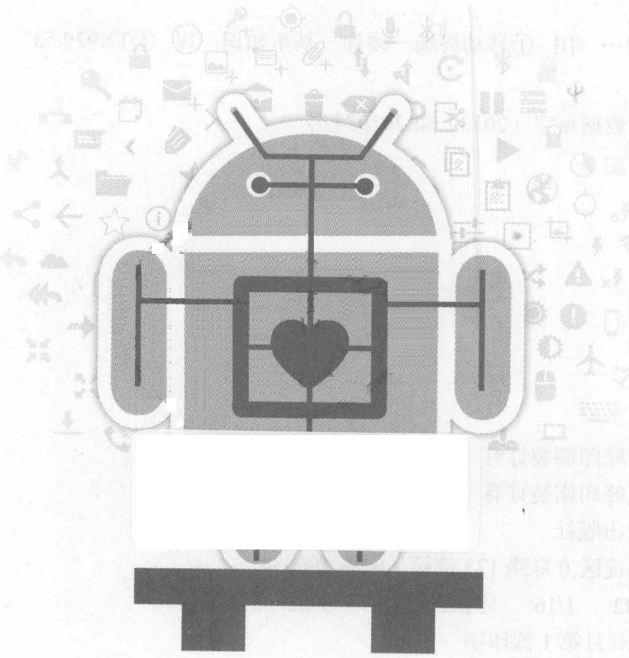
韩超 等著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Android

板级支持与硬件相关子系统

韩超 等著



電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书以硬件相关的子系统为核心，提供具有完整知识体系 Android 系统级的开发知识。本书选定了几个流行的硬件作为参考平台，读者可以很容易地得到硬件和开源代码。本书突出了硬件相关的子系统的特点，展示了几个不同的硬件平台的内核结构，介绍了每个子系统的总体结构和 BSP 结构、每个子系统的 BSP 的实现要点，以及具体硬件在 Linux 内核与 Android 硬件抽象层相关的实现。

本书适用于各类 Android 技术群体，也适用于嵌入式 Linux 的技术人员了解实际系统。作者根据丰富的开发经验和对 Android 系统发展 5 年的总结完成本书，希望为 Android 系统的开发者和学习者提供切实有效的帮助。

图书在版编目（CIP）数据

Android 板级支持与硬件相关子系统 / 韩超等著. —北京：电子工业出版社，2013.10
ISBN 978-7-121-21348-9

Ⅰ. ①A… Ⅱ. ①韩… Ⅲ. ①移动终端－硬件－基本知识 Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2013）第 202974 号

策划编辑：李　冰
责任编辑：高洪霞
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：787×1092　1/16　印张：26.25　字数：672 千字
印　　次：2013 年 10 月第 1 次印刷
印　　数：3000 册　　定价：59.00 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。

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

服务热线：（010）88258888。

前　言

开发者的需要

Android 系统已经推出了将近 5 个年头了，从 1.0 版本一直到本书写作时的 4.2 版本。作为其载体的硬件也经过了多次升级。到今天，Android 设备已经成为硬件的集大成者。硬件方面的开发一直是开发的难点，凡是一个完整 Android 设备的开发者，无论处于产业链的哪一个阶段，都不可避免地要处理与硬件相关的问题。

Android 的开发者通常面对几个方面的难点：

- Android 系统的代码庞大，难以把握硬件相关的调试思路。
- 不清楚软件和硬件之间的直接关系。
- 对某个硬件平台的知识和经验不适用于其他硬件平台。
- Android 系统的版本升级过程中，与硬件相关的部分常常发生重大变动。

另外一个客观的情况是，目前一般处理器或者基本硬件平台的 BSP（Board Support Package，板级支持包）部分都是由芯片的厂商统一完成的，并且已经趋近于成熟。因此，开发者的主要工作不再是构建完整的 BSP，而是调试和修改现有的 BSP。

本书特色

本书的目的是要为开发者提供切实有效的帮助。针对开发者的现实情况，本书主要具有以下几个特点：

- 用框图描述每一个硬件相关子系统的结构，并区分 BSP 部分和公用部分。
- 选用多个流行的硬件平台，对比其中不同的实现和相同的理念。
- 对比 Android 2.3 和 Android 4.x 的实现，展示硬件相关部分的升级。
- 对庞大的系统去耦合，展示 Android 一些原始的核心设计思路。
- 列出每一个部分相关的代码路径。
- 简要列出代码的关键部分。
- 根据实际经验编写，工程性强。

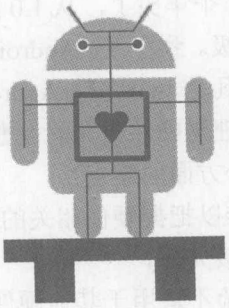
通过对每一个子系统的学习，读者可以了解现有 Android 系统的结构，经过深入理解后，可以明白 Android 系统的设计思路。学习后，如果要在系统中增加一个新的非标准硬件，读者也能比较容易地完成。

本书特别选定了 Nexus One、Nexus S、Galaxy Nexus 等几款手机作为参考平台。其中一个很大的优点就是以上几个平台都是 Google 认定的，具有典型性，并且它们从内核到 Android 系统的代码都是开源的，读者可以很容易获得。虽然以上几个硬件平台不是最新的，

但是根据嵌入式 SOC 的特点，高通的处理器都与 Nexus One 的 QSD 8x 处理器类似，三星的处理器都与 Nexus S 的 Exynos 处理器类似，德州仪器的处理器都与 Galaxy Nexus 的 OMAP 处理器类似。因此，这样的选择既方便又具有广泛的适应性，当读者熟悉了这几个硬件平台后，置于 Marvel、Freescale 和 NVidia 等公司的平台可以实现融会贯通。

Android 系统可以被视为一个功能完备的机器人。其中与硬件相关的 BSP 部分，则是这个机器人的根基和经脉，虽然占的比重不大，却是系统的关键所在。

本书的理念可以用下面的图来表示。



本书内容

本书提供了系统化的 Android 系统的开发知识，以硬件相关的子系统为核心，主要包括以下几个方面的内容：

- 硬件相关的子系统的特点。
- 几个不同的硬件平台的 Linux 内核结构。
- 每个子系统的总体结构和 BSP 结构。
- 每个子系统的 BSP 的实现要点。
- 具体硬件在 Linux 内核与 Android 硬件抽象层相关的实现。

人的知识和经验本身是网状结构，各部分相互关联，错综复杂。但是作为出版的书籍，则必须将其串行成章节的形式，本书总体上是以横向结构来进行组织的，大部分章节是针对每一个硬件相关子系统的描述，每一章的组织结构也比较类似。

本书读者

本书适用于各类 Android 技术群体，也适用于嵌入式 Linux 的技术人员了解实际系统。作者对读者有以下几个方面的建议：

- 根据书中提供的知识和经验，对照 Android 的源代码，有相应的 Android 设备，这三者的结合是最理想的学习环境。
- 要首先理解宏观结构，再研究细枝末节，硬件相关子系统的很多代码并非在任何情况下都会适用，读者需要了解其适用的场景。

- BSP 部分的开发偏重下层，读者不要过于依赖界面，而要习惯查看系统日志，从 Linux 系统标准的设备和特殊文件系统中获取信息，并使用各种命令行工具调试。
- 硬件抽象层的目的是为了适配各种硬件，很多程序的结构看似冗余，却正是 BSP 设计的精华所在，这也是读者需要关注的内容。
- 夯实 Linux 的基础对 Android 开发也非常重要，对 Android 的 BSP 开发尤为重要。
- 在开发的过程中，可能用到很多不同的硬件平台，要根据本书的思路掌握查看硬件信息和硬件相关代码的方法。

本书作者

本书的规划和统筹由中国大陆的韩超完成，韩超在 Linux 和 Android 领域具有丰富的一线开发经验。本书内容来源于工作在不同领域的开发者多年的经验。韩超完成了本书内容的主要部分，众多不同规模的企业开发成果也为本书的编写提供了重要的素材。参与本书编写的还有崔海斌、于仕林、张宇、张超、赵家维、黄亮、沈桢、徐威特、杨钰、马若劼、曹道刚、梁泉等。

目　录

第 1 章　Android 的 BSP 和子系统开发……1

1.1　Android 板级支持工作概述……1

1.1.1　Android 的开放源代码工程和 BSP……1

1.1.2　Android 的系统结构……1

1.2　Android 的开发环境和源代码……2

1.2.1　Android 的开发环境……2

1.2.2　源代码仓库……3

1.3　BSP 模块和相关子系统……5

1.3.1　Android 的 BSP……5

1.3.2　BSP 和硬件相关子系统……6

1.3.3　不同类型的 Android 设备……7

第 2 章　Android 系统 BSP 部分工作……8

2.1　Android 的 BSP 部分工作概述……8

2.2　BSP 的全局部分……8

2.2.1　源代码工程板级别支持部分……9

2.2.2　硬件相关的代码改动……11

2.3　Android 的 Linux 操作系统……14

2.3.1　Android 中的 Linux 操作系统的特定内容……14

2.3.2　Android 的 Linux 的基本支持……15

2.3.3　Android 各个硬件设备的驱动程序……16

2.4　Android 的硬件抽象层……17

2.4.1　硬件抽象层的地位和功能……17

2.4.2　硬件抽象层接口方式……18

2.5　各个子系统的移植方式……22

2.5.1　Android 2.3 中的实现方式……22

2.5.2　Android 2.2 及之前的实现方式……23

2.5.3　Android 4.x 中的实现方式……24

2.6　与硬件抽象层相关的框架层目录……24

2.6.1　一直保持不变的代码……24

2.6.2　框架层的本地代码……24

2.6.3　音频视频相关的代码……25

第 3 章　Android 的 Linux 内核和驱动……26

3.1　Android 的 Linux 内核概述……26

3.1.1　几个内核工程……26

3.1.2　内核工程的编译工具链……26

3.1.3　用户空间关注的内容……27

3.2　Android 专用驱动和组件……27

3.2.1　电源管理部分……27

3.2.2　staging 中的组件和驱动程序……28

3.2.3　几个主要核心模块……32

3.2.4　辅助的模块和改动……35

3.3　goldfish 平台的内核和驱动……37

3.3.1　goldfish 平台和内核概述……37

3.3.2　goldfish 体系结构移植……38

3.3.3　goldfish 的相关设备驱动……40

3.4　高通 MSM 平台的内核和驱动……42

3.4.1　平台概述……42

3.4.2　体系结构移植……43

3.4.3　设备驱动程序……43

3.5　三星平台的内核和驱动……44

3.5.1　平台概述……44

3.5.2 体系结构移植 ……45
3.5.3 驱动程序部分 ……45
3.6 德州仪器 OMAP 平台的内核和驱动 ……46
3.6.1 平台概述 ……46
3.6.2 体系结构移植 ……47
3.6.3 驱动程序部分 ……47
第 4 章 显示系统 ……49
4.1 显示系统概述 ……49
4.2 显示子系统结构 ……50
4.2.1 总体结构 ……50
4.2.2 核心结构和 UI 库 ……51
4.2.3 Surface 本地部分 ……54
4.2.4 Java 层的 Surface 的处理 ……56
4.3 显示 BSP 的结构 ……57
4.3.1 Framebuffer 驱动程序 ……57
4.3.2 gralloc 硬件抽象层 ……59
4.4 显示 BSP 的实现 ……61
4.4.1 模拟器显示系统的实现 ……61
4.4.2 Nexus One 系统的实现 ……68
4.4.3 Nexus S 系统的实现 ……72
4.4.4 Galaxy Nexus 系统的实现 ……73
第 5 章 用户输入系统 ……75
5.1 用户输入系统概述 ……75
5.2 Android 2.3 用户输入子系统 ……76
5.2.1 总体结构 ……76
5.2.2 本地框架的几个部分 ……77
5.2.3 JNI ……80
5.2.4 Java 层的部分 ……81
5.3 Android 4.2 的用户输入子系统结构 ……81
5.3.1 总体结构 ……81
5.3.2 InputManagerService 的实现 ……82
5.4 用户输入 BSP 的结构 ……84
5.4.1 Input 驱动程序 ……84
5.4.2 输入配置文件 ……87
5.5 用户输入 BSP 的实现 ……89
5.5.1 模拟器中的实现 ……89
5.5.2 Nexus One 系统中的实现 ……90
5.5.3 Nexus S 系统中的实现 ……93
5.5.4 Galaxy Nexus 系统中的实现 ……94
第 6 章 传感器系统 ……96
6.1 传感器系统概述 ……96
6.2 传感器子系统的结构 ……97
6.2.1 总体结构 ……97
6.2.2 本地框架层 ……98
6.2.3 传感器系统的 JNI ……99
6.2.4 传感器系统的 Java 层 ……100
6.3 传感器 BSP 的结构 ……101
6.3.1 驱动程序 ……101
6.3.2 硬件抽象层的内容 ……102
6.4 传感器 BSP 的实现 ……104
6.4.1 仿真器的实现 ……104
6.4.2 Nexus One 系统实现 ……106
6.4.3 Nexus S 系统实现 ……107
6.4.4 Galaxy Nexus 系统实现 ……109
第 7 章 音频系统 ……111
7.1 音频系统概述 ……111
7.2 音频子系统结构 ……112
7.2.1 总体结构 ……112
7.2.2 Audio 的本地框架层 ……113
7.2.3 Audio 系统的 JNI 和 Java 层 ……114
7.3 音频 BSP 的结构 ……116
7.3.1 Audio 驱动程序 ……116
7.3.2 硬件抽象层的内容 ……120
7.4 音频 BSP 的实现 ……124
7.4.1 通用的 Audio 系统实现 ……124
7.4.2 基于 OSS 的实现方式 ……129
7.4.3 基于 ALSA 的实现方式 ……130

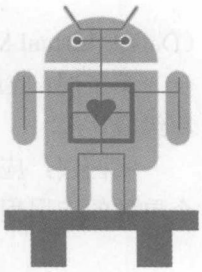
7.4.4 MSM 平台和 Nexus One 系统的实现……132
7.4.5 Nexus S 系统的实现……137
第 8 章 视频叠加输出系统……140
8.1 视频叠加输出系统概述……140
8.2 视频输出子系统的结构……141
8.2.1 Overlay 系统的结构……141
8.2.2 本地框架层……142
8.3 视频叠加输出 BSP 结构……144
8.3.1 移植的内容……144
8.3.2 驱动程序……144
8.3.3 硬件抽象层的内容……144
8.3.4 视频输出的调用者……146
8.3.5 使用 Overlay 的数据流情况……148
8.4 视频输出 BSP 的实现……149
8.4.1 骨架实现……149
8.4.2 OMAP 系统的实现……151
8.4.3 Nexus S 系统的实现……156
第 9 章 照相机系统……159
9.1 照相机系统概述……159
9.2 照相机子系统的结构……160
9.2.1 照相机系统的结构……160
9.2.2 Camera 的本地层……161
9.2.3 Camera 的 JNI 和 Java 层……165
9.3 照相机 BSP 的结构……166
9.3.1 移植的内容……166
9.3.2 Video for 4 Linux 驱动程序……166
9.3.3 硬件抽象层的内容……168
9.3.4 照相机系统上下层的关系……173
9.4 照相机 BSP 的实现……175
9.4.1 桩实现……175
9.4.2 Nexus One 系统的 Camera 实现……178
9.4.3 Nexus S 系统的 Camera 实现……180
第 10 章 OpenGL 3D 引擎……184
10.1 OpenGL 系统概述……184
10.2 OpenGL 系统的结构……184
10.2.1 OpenGL 和 OpenGL ES 的标准结构……184
10.2.2 总体结构……186
10.2.3 OpenGL 库的调用者……188
10.3 OpenGL BSP 的结构……190
10.3.1 移植的内容……190
10.3.2 OpenGL 移植层的接口……190
10.3.3 OpenGL 的调用和测试……192
10.4 OpenGL BSP 的实现……193
10.4.1 Android 软件 OpenGL 的实现……193
10.4.2 Nexus One 系统的实现……195
10.4.3 Nexus S 系统的实现……195
10.4.4 Galaxy Nexus 系统的实现……196
第 11 章 OpenMax 引擎……197
11.1 OpenMax 系统概述……197
11.2 OpenMax 子系统结构……197
11.2.1 OpenMax 系统的结构……197
11.2.2 Android 中 OpenMax 的适配层……201
11.3 OpenMax BSP 的结构……203
11.3.1 OpenMax IL 层的接口……203
11.3.2 Android 的 OpenMax……207
11.4 OpenMax BSP 的实现……207
11.4.1 OpenMax IL 实现的内容……207
11.4.2 OMAP3 的 OpenMax IL 实现的结构和机制……208
11.4.3 OMAP4 的 OpenMax IL 实现……213

第 12 章　位块复制……216
12.1　位块复制概述……216
12.2　位块复制子系统结构……216
12.2.1　总体结构……216
12.2.2　copybit 的调用者……217
12.3　位块复制 BSP 的结构……218
12.3.1　驱动程序……218
12.3.2　硬件抽象层的接口……218
12.3.3　实现硬件抽象层……220
12.4　位块复制的实现……220
第 13 章　无线局域网系统……223
13.1　无线局域网系统概述……223
13.2　无线局域网子系统的结构……223
13.2.1　总体的结构……223
13.2.2　wpa_supplicant 工程……225
13.2.3　WiFi 本地适配库……227
13.2.4　WiFi 的 JNI 部分……228
13.2.5　WiFi 的 Java 层……228
13.3　无线局域网 BSP 的结构……229
13.3.1　协议和驱动程序……229
13.3.2　用户空间的内容……230
13.4　无线局域网 BSP 的实现……231
13.4.1　基于 BCM4329 的方案（Nexus One 和 Nexus S）……231
13.4.2　OMAP 平台的一个典型实现……232
13.4.3　Galaxy Nexus 的实现……234
第 14 章　蓝牙系统……237
14.1　蓝牙系统概述……237
14.2　蓝牙子系统的结构……237
14.2.1　蓝牙系统的结构……237
14.2.2　BlueZ……239
14.2.3　bluedroid 库……241
14.2.4　蓝牙的 JNI 部分……241
14.2.5　蓝牙的 Java 部分……242
14.3　BSP 的结构……243
14.3.1　协议和驱动程序……244
14.3.2　本地代码的配置部分……245
14.4　Android 4.2 的蓝牙系统……246
14.4.1　系统结构……246
14.4.2　蓝牙硬件模块……246
14.4.3　蓝牙系统的本地层部分……252
14.4.4　BlueTooth 包……252
14.5　BSP 的实现……253
14.5.1　Nexus One 系统的蓝牙实现……253
14.5.2　Nexus S 系统的蓝牙实现……254
14.5.3　Galaxy Nexus 系统的蓝牙实现……255
第 15 章　定位系统……257
15.1　定位系统的概述……257
15.2　定位子系统的结构……257
15.2.1　总体结构……257
15.2.2　JNI 部分……259
15.2.3　Java 部分……260
15.3　定位 BSP 的结构……263
15.3.1　驱动程序……263
15.3.2　硬件抽象层的接口……264
15.3.3　实现硬件抽象层……266
15.4　定位 BSP 的实现……266
15.4.1　仿真器的 GPS 实现……266
15.4.2　Nexus One 系统的实现……268
15.4.3　Nexus S 系统的实现……269
15.4.4　Galaxy Nexus 系统的实现……270
第 16 章　电话系统……271
16.1　电话系统概述……271
16.2　电话子系统的结构……271
16.2.1　总体结构……271
16.2.2　rild 层……273

16.2.3 Java 层中的电话部分 ……275
16.3 电话 BSP 的结构……278
16.3.1 驱动程序 ……278
16.3.2 RIL 实现库接口（作为硬件抽象层） ……280
16.4 电话 BSP 部分的实现 ……281
16.4.1 RIL 的参考实现 ……281
16.4.2 数据连接部分 ……287
16.4.3 Mock RIL ……288
第 17 章 警报器—实时时钟系统……290
17.1 警报器—实时时钟系统 ……290
17.2 警报器—实时时钟子系统的结构……290
17.2.1 总体结构 ……290
17.2.2 JNI 部分 ……291
17.2.3 Java 部分 ……292
17.2.4 Android 系统时间方面的调用 ……292
17.3 警报器—实时时钟 BSP 部分的结构 ……293
17.3.1 RTC 驱动程序……293
17.3.2 Alarm 驱动程序 ……294
17.4 警报器—实时时钟 BSP 部分的实现 ……295
17.4.1 模拟器环境中的实现……295
17.4.2 MSM 平台和 Nexus One 系统的实现 ……295
17.4.3 Nexus S 系统的实现 ……297
17.4.4 Galaxy Nexus 系统的实现 ……298
第 18 章 光系统 ……300
18.1 光系统概述 ……300
18.2 背光和指示灯子系统的结构 ……300
18.2.1 总体结构 ……300
18.2.2 JNI 部分 ……301
18.2.3 Java 服务部分和调用部分 ……302
18.3 背光和指示灯 BSP 部分的结构 ……303
18.3.1 驱动程序 ……303
18.3.2 硬件抽象层的内容……304
18.4 背光和指示灯 BSP 部分的实现……305
18.4.1 Nexus One 系统的实现 ……305
18.4.2 Nexus S 系统的实现 ……307
18.4.3 Galaxy Nexus 系统的实现 ……308
第 19 章 振动器系统……311
19.1 振动器系统概述 ……311
19.2 振动器子系统的结构……311
19.2.1 振动器部分的结构……311
19.2.2 JNI 部分……312
19.2.3 Java 框架部分 ……313
19.3 振动器 BSP 部分的结构……313
19.3.1 驱动程序 ……313
19.3.2 硬件抽象层的内容……314
19.4 振动器 BSP 部分的实现……314
19.4.1 Nexus One 系统的实现……315
19.4.2 Nexus S 系统的实现 ……316
19.4.3 Galaxy Nexus……316
第 20 章 电池信息部分 ……318
20.1 电池信息部分……318
20.2 电池信息子系统的结构 ……318
20.2.1 电池系统部分的结构……318
20.2.2 JNI 部分……319
20.2.3 Java 部分 ……321
20.3 电池信息 BSP 部分的结构 ……321
20.4 电池信息 BSP 部分的实现 ……322
20.4.1 模拟器中的实现……322
20.4.2 Nexus One……323
20.4.3 Nexus S……324

20.4.4 Galaxy Nexus……325
第 21 章 Android 4.x 的音频、视频系统……326
21.1 Android 4.x 的音频系统……326
21.1.1 音频系统的结构……326
21.1.2 音频框架层……327
21.1.3 音频 BSP 部分结构……327
21.2 Android 4.x 音频的 BSP 实现……330
21.2.1 主实现和策略实现……330
21.2.2 仿真器实现……330
21.2.3 A2DP 实现……331
21.2.4 Galaxy Nexus 的实现……332
21.3 Android 4.x 照相机系统……336
21.3.1 照相机系统的结构……336
21.3.2 Camera 的框架层……336
21.3.3 照相机 BSP 部分结构……339
21.4 Android 4.x 照相机的 BSP 实现……342
21.4.1 仿真器实现……342
21.4.2 Galaxy Nexus 的实现……346
21.5 Android 4.x 视频组合系统……349
21.5.1 视频组合系统结构……349
21.5.2 SurfaceFlinger 对视频组合的使用……350
21.5.3 视频组合 BSP 部分结构……351
21.6 Android 4.x 视频组合的 BSP 实现……352
21.6.1 默认实现……352
21.6.2 Galaxy Nexus 的视频组合……352
第 22 章 Android 4.x 近场通信系统……357
22.1 近场通信系统概述……357
22.2 近场通信子系统的结构……358
22.2.1 总体结构……358
22.2.2 NFC 本地库……359
22.2.3 Android 框架层的 NFC 相关内容……360
22.2.4 NFC 包……361
22.3 近场通信 BSP 的结构……365
22.3.1 NFC-NCI 接口……365
22.3.2 NFC 接口……366
22.4 近场通信 BSP 的实现……366
22.4.1 NCI-NFC 的桩实现……366
22.4.2 NFC 的桩实现……366
22.4.3 Galaxy Nexus 的 NFC 实现……367
第 23 章 Android 4.2 的电源控制……368
23.1 电源控制……368
23.2 电源控制的结构……368
23.2.1 总体结构……368
23.2.2 电源控制的使用……368
23.3 电源控制 BSP 的结构……369
23.4 电源控制 BSP 的实现……369
23.4.1 通用的电源控制实现……369
23.4.2 Galaxy Nexus 的电源控制实现……370
第 24 章 本地时间……372
24.1 本地时间子系统结构……372
24.1.1 本地时间的结构……372
24.1.2 本地时间的使用……372
24.2 本地时间 BSP 的结构……373
24.3 本地时间 BSP 的实现……373
第 25 章 Android 4.2 密钥……375
25.1 密钥概述……375
25.2 安全和密钥子系统结构……376
25.2.1 安全和密钥的总体结构……376
25.2.2 keystore 守护进程……376
25.2.3 android.security 的内容……377
25.3 密钥的 BSP 部分的结构……379

25.4 密钥的 BSP 实现……380
25.4.1 通用的软件密钥实现……380
25.4.2 Galaxy Nexus 的密钥实现……381
第 26 章 电源管理……384
26.1 Android 电源管理……384
26.2 Android 内核空间的电源管理……385
26.2.1 总体结构……385
26.2.2 wakelock……386
26.2.3 wakelock 的用户空间……388
26.2.4 earlysuspend 部分……389
26.2.5 其他……391
26.3 Android 用户空间的电源管理……392
26.3.1 电源管理的本地库……392
26.3.2 电源管理的 JNI 库……393
26.3.3 电源管理的 Java 部分……393
26.4 电源管理的策略……394
26.4.1 驱动程序的变化……394
26.4.2 用户空间的控制……396
第 27 章 恢复和升级……397
27.1 恢复和升级概述……397
27.1.1 Android 的 Recovery 系统的组成……397
27.1.2 Android 的 Recovery 系统的功能和运行流程……398
27.2 recovery 系统……399
27.2.1 编译系统……399
27.2.2 init.rc 脚本……400
27.2.3 Recovery 可执行程序和相关的库……401
27.3 Android 系统交互的过程……405
27.3.1 Java 部分……405
27.3.2 交互的场景……406



第1章 Android的BSP和子系统开发

1.1 Android板级支持工作概述

1.1.1 Android的开放源代码工程和BSP

Android是一个开放源代码的系统，基于Android的开源代码，可以构建各种系统，并可以适配到任何支持的硬件平台上。

BSP（Board Support Package）的含义为板级支持包，通常表示某个系统的特定的硬件支持部分。从系统结构上，某一个硬件相关的软件系统可以分成通用部分和BSP部分。前者与硬件无关，后者与硬件相关。

AOSP（Android Open-Source Project）的含义为Android开放源代码工程。Google官方的Android代码和相关内容都是开放的，其网站为：http://source.android.com/。

AOSP的大部分工程采用了Apache协议（Apache License）。Apache协议鼓励代码共享和尊重原作者的著作权，同样允许代码修改。Apache协议也是对商业应用的友好许可，使用者也可以在需要时修改代码满足需要并作为开源或商业产品发布或销售。

各个基于Android的设备基本都是直接或间接地从Android的开源工程获取源代码，然后加入针对特定硬件的BSP部分，由此构建了自己的软件部分。

1.1.2 Android的系统结构

Android系统虽然庞大，但是具有清晰的软件层次结构。按照自下而上的结构，Android的软件系统分成4个层次，如图1-1所示。

第一层：操作系统（Linux kernel）。Android系统基于Linux操作系统，第一层次为Linux内核和相关驱动。不同的硬件平台使用不同的内核。

第二层：库（Libraries）和运行环境（Android Runtime）。包括本地的各个C语言和C++库，可能来自开源工程或者Android原生的。运行环境包括C++语言实现的虚拟机

（Dalvik Virtual Machine）和 Java 核心类（Core Libraries）。

第三层：应用程序框架（Application Framework）。Android 原创 Java 框架层由几个 Java 库组成。

第四层：应用程序（Applications）。由 Java 代码、资源文件、工程描述文件生成的各个单独的应用程序包。

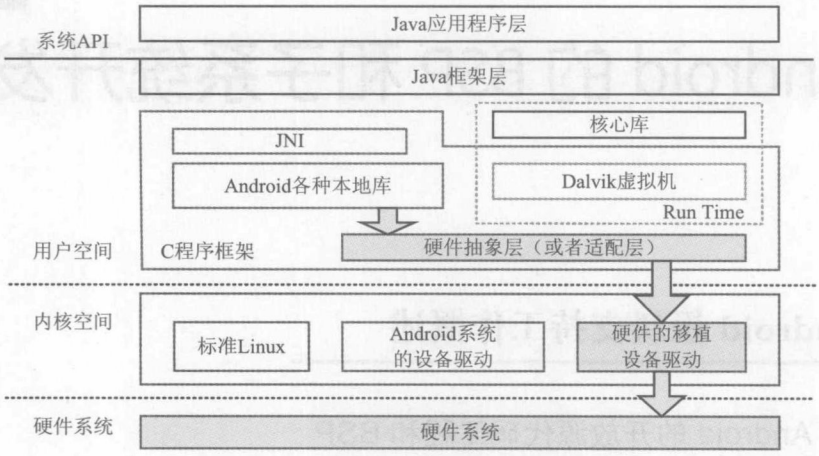


图 1-1　Android 系统的层次结构

从 BSP 开发的角度来说，重点在 Android 系统较下层，Android 系统软件的运行基础是需要适配系统的硬件，在软件方面主要的工作是 Linux 内核中的硬件驱动程序（第一层），以及 Android 本地层当中的硬件抽象层或者适配层（第二层）。

1.2　Android 的开发环境和源代码

1.2.1　Android 的开发环境

Android 的 Linux 开发环境一般使用 Ubuntu 主机。在基于 Ubuntu 的主机环境中，开发 Android 主机环境包括以下需求：git 工具、repo 工具、Java 的 JDK、主机编译工具等。编译过程中主要的目标机编译工具则使用 Android 开源工程中自带的。

Android 系统在编译过程中需要编译主机的工具，因此需要使用主机的 GCC 工具链。而对于编译目标机文件，Android 在 prebuilt 目录中集成了 GCC 交叉编译工具链。

新版本 Android 源代码编译应当使用 64 位的主机环境，配置方式如下所示：

```
$ sudo apt-get install git-core 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 \
  libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown \
  libxml2-utils
```

而 32 位的主机开发环境，配置方式如下所示：

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential \
  zip curl zlib1g-dev libc6-dev libncurses5-dev x11proto-core-dev \
  libx11-dev libreadline6-dev libgl1-mesa-dev tofrodos python-markdown \
  libxml2-utils
```

repo 是调用 git 封装的工具，安装 repo 的准备工作如下所示：

```
$ mkdir ~/bin
$ PATH=~/bin:$PATH
```

repo 工具同样可以在 Android 的网站上获得，方法如下所示：

```
$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
```

repo 是一个由 Google 提供的工具，本质上是一个脚本，它可能随时更新，因此应当从 Google 的相应地址获得最新的 repo，复制到本地之后更改权限来使用。

1.2.2　源代码仓库

AOSP 工程的源代码放置在 Android 的源代码仓库中，其网站为：

https://android.googlesource.com/

这个网址既可以作为使用 git 工具获取和提交源代码的地方，也可以通过浏览器使用 https 协议访问，由此获得源代码仓库的一些信息。

1. repo 获取全部工程

直接使用 repo 获取 Android 完全的源代码方法包括初始化代码仓库和获取代码两个步骤，每个步骤可以增加不同的参数。

使用 repo 初始化 Android 的代码仓库的一般方法如下所示：

```
$ repo init -u https://android.googlesource.com/platform/manifest
```

由于没有使用额外的参数，此时得到的是代码仓库中 master 分支（主分支）的最新版本。在初始化过程中，也可以看到列出的各个分支（branch）和标签（tag）的名称，这些就是可以在初始化的过程中使用-b 指定的参数。

在 repo init 时，使用-b 选项下载稳定的 Android 2.3.3 版本：

```
$ repo init -u https://android.googlesource.com/platform/manifest -b android-2.3.3_r1
```

在 repo init 时，使用-b 选项下载 Android 4.x 的版本：

```
$ repo init -u https://android.googlesource.com/platform/manifest -b android-4.0.4_r2
```

repo init 之后，将生成隐藏目录.repo，其中文件.repo/manifest.xml 为 repo 工程的描述文件，表示 repo 时包含的各个工程，其中的几个条目如下所示：

```
<project path="frameworks/base"  name="platform/frameworks/base" />
<project path="hardware/libhardware" name="platform/hardware/libhardware" />
<project path="hardware/libhardware_legacy"
                    name="platform/hardware/libhardware_legacy" />
```

repo/manifest.xml 中的 path 表示工程获取后的路径（基于当前目录），name 表示工程

的原始名称。

在经过 repo init 之后，可以使用 repo 获取 Android 的全部代码，方法如下所示：

```
$ repo sync
```

使用 repo sync 时，也可以同步一个单个工程的内容，需要使用工程的名称作为 repo sync 的参数，工程的名称可以从 manifest.xml 获得。获得单个工程的方法如下所示：

```
$ repo sync {project_name}
```

获取工程后，每个工程的目录中含有一个.git 目录，这就是工程版本管理目录，这些.git 目录中的内容大部分是到.repo/projects/<project_path >的连接。

从 BSP 开发的角度，Android 源代码工程中，几个重点的目录如下。

- frameworks/base：核心框架部分，包括 C++和 Java 的框架。
- hardware/libhardware：硬件相关内容，包括硬件模块。
- hardware/libhardware_legacy：陈旧的硬件相关内容，有些内容依然在使用。
- hardware/<vendor>/：包括 qcom 和 ti 等厂商针对自己硬件的内容。
- device/*/<TARGET_PRODUCT>：各个目标板的配置目录。

2. 其他工程

Android 的开源代码包括了很多工程，使用 repo 并非得到所有的工程，而是使用一个特定的工程列表。可以直接使用 git 工具获取其他工程：

```
$ git clone https://android.googlesource.com/name
```

工程的名字可以从 https://android.googlesource.com/获取。

例如，获得 repo 工具的方法如下所示：

```
$ git clone https://android.googlesource.com/tools/repo
```

命令执行完成后，将获得名称为 repo 的目录，其中包含.git 目录用于存放工程的版本控制信息。可以进入并查看其中的内容：

```
$ cd repo
$ git branch -r
  origin/HEAD -> origin/master
  origin/maint
  origin/master
  origin/stable
```

git branch –r 列出了远端的分支名称，进一步使用 git checkout 可以取出某个稳定分支的代码。

一些与 BSP 开发相关的工程如下所示。

- kernel/*：各个不同系统的 Linux 内核的源代码。
- platform/prebuilts/gcc/*：各种交叉编译工具链（包括不同版本和体系结构）。
- toolchain/*：主机自身的一些工具。