

剖析各类驱动的实现过程，分享移动开发的移植技巧



高金昌 张明星◎编著

Android

底层驱动分析和移植

- ✚ 贯通底层驱动、中间层JNI制作、上层UI 接口设计。
- ✚ 面向实战，深入剖析各驱动系统的完整实现流程。
- ✚ 源码分析+全真示例+图片解析=更易于理解的思维路径。
- ✚ 教授精髓，精讲精炼。赠送源码，拿来就用。



清华大学出版社

Android 底层驱动分析和移植

高金昌 张明星 编著

清华大学出版社

北 京

内 容 简 介

Android 系统从诞生到现在,在短短的几年时间里,便凭借其操作易用性和开发的简洁性,赢得了广大用户和开发者的支持。本书内容分为 3 篇,共 22 章,循序渐进地讲解了 Android 底层系统中的典型驱动方面的知识。本书从获取源码和源码结构分析讲起,依次讲解了基础知识篇、Android 专有驱动篇和典型驱动移植篇 3 部分的基本知识。在讲解每一个驱动时,从 Android 系统的架构开始讲起,从内核分析到具体的驱动实现,再从 JNI 层架构分析到 Java 应用层的接口运用,最后到典型驱动系统移植和开发,彻底剖析了每一个典型驱动系统的完整实现流程。本书几乎涵盖了所有 Android 底层驱动的内容,讲解方法通俗易懂,内容翔实,不但适合应用高手的学习,也特别有利于初学者学习和消化。

本书适合作为 Android 驱动开发者、Linux 开发人员、Android 底层学习人员、Android 爱好者、Android 源码分析人员、Android 应用开发人员的学习用书,也可以作为相关培训学校和大专院校相关专业的教学用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

Android 底层驱动分析和移植/高金昌,张明星编著. —北京:清华大学出版社,2015
ISBN 978-7-302-39745-8

I. ①A… II. ①高… ②张… 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 071318 号



责任编辑:朱英彪

封面设计:刘超

版式设计:魏远

责任校对:王云

责任印制:杨艳

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社总机:010-62770175 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者:清华大学印刷厂

装 订 者:三河市少明印务有限公司

经 销:全国新华书店

开 本:203mm×260mm 印 张:50.75 字 数:1530 千字

版 次:2015 年 7 月第 1 版 印 次:2015 年 7 月第 1 次印刷

印 数:1~3000

定 价:96.00 元

产品编号:061538-01

前 言

2007 年 11 月 5 日, Google 公司宣布的基于 Linux 平台的开源手机操作系统 Android 诞生, 该平台号称是首个为移动终端打造的真正开放和完整的移动软件。本书将引领广大读者一起探讨这款系统的神奇之处。

市场占有率高居第一

截至 2014 年 1 月, Android 在手机市场上的占有率从 2013 年的 68.8% 上升到 78.9%。而 iOS 则从 2013 年的 19.4% 下降到 15.5%, WP 系统从原来的 2.7% 小幅上升到 3.6%。

从数据上来看, Android 平台占据了市场的主导地位, 继续称当老大的角色。Android 市场的占有率增加幅度较大, WP 市场小幅增长, 但 iOS 却有所下降。就目前来看, 智能手机的市场已经饱和, 大多数人都 在各个平台中转换。而就在这样的市场背景下, Android 还增长了 10% 左右的占有率, 确实不易。

为开发人员提供了滋长的“沃土”

(1) 保证开发人员可以迅速转型为 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, 就能很快地上手并掌握。作为单独的 Android 应用开发, 对 Java 编程门槛的要求并不高, 即使是没有编程经验的门外汉, 也可以在突击学习 Java 之后学习 Android。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, 例如常见的工具、管理、互联网和游戏等。

(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 公司已经成功举办了奖金为数千万美元的开发竞赛, 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, 不但能练习自己的开发水平, 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作品赚钱

为了能让 Android 平台吸引更多的关注, Google 公司提供了一个专门下载 Android 应用的门店: Android Market, 地址是 <https://play.google.com/store>。这个门店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, 因此吸引了更多的开发人员加入 Android 大军中来。

本书的内容

本书内容分为 3 篇, 共 22 章, 循序渐进地讲解了 Android 底层系统中的典型驱动方面的知识。本书从

获取源码和源码结构分析讲起，依次讲解了基础知识篇、Android 专有驱动篇和典型驱动移植篇 3 部分的基本知识。在讲解每一个驱动时，从 Android 系统的架构开始讲起，从内核分析到具体的驱动实现，再从 JNI 层架构分析到 Java 应用层的接口运用，最后到典型驱动系统移植和开发，彻底剖析了每一个典型驱动系统的完整实现流程。本书几乎涵盖了所有 Android 底层驱动的内容，讲解方法通俗易懂，内容翔实，不但适合应用高手们的学习，也特别有利于初学者学习和消化。

本书的版本

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。由此可见，Android 系统升级频率较快，一年之中最少有两个新版本诞生。如果过于追求新版本，会造成力不从心的结果。所以在此建议广大读者不必追求最新的版本，只需关注最流行的版本即可。据官方统计，截至 2014 年 10 月 25 日，占据前 3 位的版本分别是 Android 4.4、Android 4.3 和 Android 4.2。其中从 Android 4.4 开始，Android L 和 Android 5.0 的底层架构知识基本类似。其实这 3 个版本的区别并不是很大，只是在顶层 API 应用层的细节上进行了更新。为了及时学习 Android 系统的最新功能，本书中使用的版本是目前（本书成稿时）最新的 Android 5.0。

本书特色

本书内容十分丰富，分析细致、精准、全面。我们的目标是通过一本图书，提供多本图书的价值，读者可以根据自己的需要有选择地阅读。在内容的编写上，本书具有以下特色。

1. 内容全面，讲解细致

本书几乎涵盖了 Android 系统中所有的独有驱动和设备驱动，详细讲解了每一个典型驱动的实现过程和具体移植方法。每一个知识点都力求用通俗易懂的语言详尽展现在读者面前。

2. 遵循合理的主线进行讲解

为了使广大读者彻底了解 Android 平台中的各个驱动系统，在讲解每一个驱动系统时，从 Linux 内核开始讲起，依次剖析了驱动层实现、JNI 层分析、Java 应用和系统移植改造等内容，遵循了从底层到顶层，实现了驱动系统大揭秘的目标。

3. 章节独立，自由阅读

本书中的每一章内容都可以独自成书，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针对性的学习。与传统的计算机书籍相比，阅读本书会更轻松。

4. 驱动典型，实用性强

本书讲解了现实中最典型驱动系统的实现和移植知识，这些驱动都是在商业项目中最需要的部分，读者可以直接将本书中的知识应用到自己的项目中，实现无缝对接。

读者对象

本书适合作为 Android 驱动开发者、Linux 开发人员、Android 底层学习人员、Android 爱好者、Android 源码分析人员、Android 应用开发人员的学习用书，也可以作为相关培训学校和大专院校相关专业的教学用书。

参与本书编写的人员还有周秀、付松柏、邓才兵、钟世礼、谭贞军、张加春、王教明、万春潮、郭慧玲、侯恩静、程娟、王文忠、陈强、何子夜、李天祥、周锐、朱桂英、张元亮、张韶青、秦丹枫。本团队在编写过程中，得到了清华大学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。此外也十分感谢我们的家人，在写作时给予了巨大的支持。另外告知各位读者，因编者水平有限，如有纰漏和不尽如人意之处，恳请读者提出意见或建议，以便修订并使之更臻完善。另外我们提供了售后支持网站：<http://www.chubanbook.com/>和 QQ 群 192153124，读者朋友如有疑问可以在此提出，一定会得到满意的答复。

编 者

目 录

第 1 篇 基础知识篇

第 1 章 Android 底层开发基础	2	2.2.4 Android 和 Linux 内核的区别	32
1.1 Android 系统介绍	2	2.2.5 Android 独有的驱动	34
1.2 Android 系统架构介绍	2	2.2.6 为 Android 构建 Linux 的操作系统	35
1.2.1 底层操作系统层 (OS)	3	2.3 Linux 内核结构	35
1.2.2 各种库 (Libraries) 和 Android 运行环境 (RunTime)	3	2.3.1 Linux 内核的体系结构	35
1.2.3 应用程序框架 (Application Framework)	4	2.3.2 和 Android 驱动开发相关的内核知识	37
1.2.4 顶层应用程序 (Application)	4	2.4 分析 Linux 内核源码	40
1.3 获取 Android 源码	5	2.4.1 源码目录结构	40
1.3.1 在 Linux 系统中获取 Android 源码	5	2.4.2 浏览源码的工具	42
1.3.2 在 Windows 平台上获取 Android 源码	7	2.4.3 GCC 特性	43
1.4 分析 Android 源码结构	9	2.4.4 链表的重要性	46
1.4.1 总体结构	10	2.4.5 Kconfig 和 Makefile	48
1.4.2 应用程序部分	11	2.5 学习 Linux 内核的方法	50
1.4.3 应用程序框架部分	13	2.5.1 分析 USB 子系统的代码	50
1.4.4 系统服务部分	13	2.5.2 分析 USB 系统的初始化代码	50
1.4.5 系统程序库部分	15	2.6 Linux 中的 3 类驱动程序	54
1.4.6 系统运行库部分	18	2.6.1 字符设备驱动	54
1.4.7 硬件抽象层部分	19	2.6.2 块设备驱动	61
1.5 编译源码	20	2.6.3 网络设备驱动	65
1.5.1 搭建编译环境	20	2.7 Android 系统移植基础	65
1.5.2 在模拟器中运行	22	2.7.1 移植的任务	65
1.5.3 编译源码生成 SDK	23	2.7.2 需要移植的内容	66
第 2 章 Android 驱动开发基础	28	2.7.3 驱动开发需要做的工作	67
2.1 驱动程序基础	28	2.8 内核空间和用户空间之间的接口	67
2.1.1 什么是驱动程序	28	2.8.1 内核空间和用户空间的相互作用	67
2.1.2 驱动开发需要做的工作	29	2.8.2 实现系统和硬件之间的交互	67
2.2 Linux 开发基础	30	2.8.3 从内核到用户空间传输数据	69
2.2.1 Linux 简介	30	2.9 编写 JNI 方法	72
2.2.2 Linux 的发展趋势	31	第 3 章 主流内核系统解析	76
2.2.3 Android 基于 Linux 系统	31	3.1 Goldfish 内核和驱动解析	76

3.1.1 Goldfish 基础.....	77
3.1.2 Logger 驱动.....	78
3.1.3 Low Memory Killer 组件.....	79
3.1.4 Timed Output 驱动.....	79
3.1.5 Timed Gpio 驱动.....	80
3.1.6 Ram Console 驱动.....	80
3.1.7 Ashmem 驱动.....	81
3.1.8 Pmem 驱动.....	81
3.1.9 Alarm 驱动.....	81
3.1.10 USB Gadget 驱动.....	82

3.1.11 Paranoid 驱动介绍.....	82
3.1.12 Goldfish 的设备驱动.....	83
3.2 MSM 内核和驱动架构.....	85
3.2.1 高通公司介绍.....	85
3.2.2 常见的 MSM 处理器产品.....	86
3.2.3 MSM 内核移植.....	87
3.2.4 Makefile 文件.....	88
3.2.5 驱动和组件.....	90
3.2.6 设备驱动.....	92
3.2.7 高通特有的组件.....	94

第 2 篇 Android 专有驱动篇

第 4 章 分析硬件抽象层..... 98

4.1 HAL 基础.....	98
4.1.1 推出 HAL 的背景.....	98
4.1.2 HAL 的基本结构.....	99
4.2 分析 HAL module 架构.....	101
4.2.1 结构体 hw_module_t.....	101
4.2.2 结构体 hw_module_methods_t.....	102
4.2.3 结构体 hw_device_t.....	103
4.3 分析文件 hardware.c.....	103
4.3.1 寻找动态链接库的地址.....	103
4.3.2 数组 variant_keys.....	104
4.3.3 载入相应的库.....	104
4.3.4 获得 hw_module_t 结构体.....	105
4.4 分析硬件抽象层的加载过程.....	106
4.5 分析硬件访问服务.....	109
4.5.1 定义硬件访问服务接口.....	109
4.5.2 具体实现.....	110
4.6 分析 Mokoid 实例.....	111
4.6.1 获取实例工程源码.....	112
4.6.2 直接调用 service 方法的实现代码.....	113
4.6.3 通过 Manager 调用 service 的实现代码.....	117
4.7 HAL 和系统移植.....	120
4.7.1 移植各个 Android 部件的方式.....	120
4.7.2 设置设备权限.....	121
4.7.3 init.rc 初始化.....	125
4.7.4 文件系统的属性.....	125

4.8 开发自己的 HAL 驱动程序..... 126

4.8.1 封装 HAL 接口.....	126
4.8.2 开始编译.....	129

第 5 章 Binder 通信驱动详解..... 130

5.1 分析 Binder 驱动程序.....	130
5.1.1 数据结构 binder_work.....	130
5.1.2 结构体 binder_node.....	131
5.1.3 结构体 binder_ref.....	132
5.1.4 通知结构体 binder_ref_death.....	133
5.1.5 结构体 binder_buffer.....	133
5.1.6 结构体 binder_proc.....	134
5.1.7 结构体 binder_thread.....	135
5.1.8 结构体 binder_transaction.....	136
5.1.9 结构体 binder_write_read.....	136
5.1.10 Binder 驱动协议.....	137
5.1.11 枚举 BinderDriverReturnProtocol.....	137
5.1.12 结构体 binder_ptr_cookie 和 binder_transaction_data.....	138
5.1.13 结构体 flat_binder_object.....	139
5.1.14 设备初始化.....	139
5.1.15 打开 Binder 设备文件.....	141
5.1.16 实现内存映射.....	142
5.1.17 释放物理页面.....	147
5.1.18 分配内核缓冲区.....	148
5.1.19 释放内核缓冲区.....	150

第 3 篇 典型驱动移植篇

第 10 章 电源管理驱动	286	11.2.1 设备实现	341
10.1 Power Management 架构基础	286	11.2.2 PMEM 驱动的具体实现	343
10.2 分析 Framework 层	287	11.2.3 调用 PMEM 驱动的流程	367
10.2.1 文件 PowerManager.java	287	11.3 用户空间接口	367
10.2.2 提供 PowerManager 功能	288	11.3.1 释放位图内存	368
10.3 JNI 层架构分析	309	11.3.2 释放位图内存空间	369
10.3.1 定义两层之间的接口函数	309	11.3.3 获取位图占用内存	370
10.3.2 与 Linux Kernel 层进行交互	311	11.4 实战演练——将 PMEM 加入到 内核中	370
10.4 Kernel (内核) 层架构分析	311	11.5 实战演练——将 PMEM 加入到 内核中	372
10.4.1 文件 power.c	312	11.6 实战演练——PMEM 在 Camera 中的 应用	373
10.4.2 文件 earlysuspend.c	314	11.7 实战演练——PMEM 的移植与测试 ...	375
10.4.3 文件 wakelock.c	315	第 12 章 调试机制驱动 Ram Console	378
10.4.4 文件 resume.c	317	12.1 Ram Console 介绍	378
10.4.5 文件 suspend.c	317	12.2 实现 Ram Console	378
10.4.6 文件 main.c	318	12.2.1 定义结构体 ram_console_platform_data ...	379
10.4.7 proc 文件	319	12.2.2 实现具体功能	379
10.5 wakelock 和 early_suspend	319	第 13 章 USB Gadget 驱动	389
10.5.1 wakelock 的原理	319	13.1 分析 Linux 内核的 USB 驱动程序	389
10.5.2 early_suspend 原理	320	13.1.1 USB 设备基础	389
10.5.3 Android 休眠	321	13.1.2 USB 和 sysfs	393
10.5.4 Android 唤醒	323	13.1.3 urb 通信	396
10.6 Battery 电池系统架构和管理	323	13.2 USB Gadget 驱动架构详解	401
10.6.1 实现驱动程序	324	13.2.1 分析软件结构	401
10.6.2 实现 JNI 本地代码	325	13.2.2 层次整合	411
10.6.3 Java 层代码	325	13.2.3 USB 设备枚举	421
10.6.4 实现 Uevent 部分	327	13.3 实战演练——USB 驱动例程分析	437
10.7 JobScheduler 节能调度机制	331	13.3.1 结构体 usb_device_id	437
10.7.1 JobScheduler 机制的推出背景	331	13.3.2 结构体 usb_driver	439
10.7.2 JobScheduler 的实现	332	13.3.3 注册 USB 驱动程序	440
10.7.3 实现操作调度	332	13.3.4 加载和卸载 USB 骨架程序模块	441
10.7.4 封装调度任务	335	13.3.5 探测回调函数	441
第 11 章 PMEM 内存驱动架构	339	13.3.6 清理数据	443
11.1 PMEM 初步	339		
11.1.1 什么是 PMEM	339		
11.1.2 Platform 设备基础	339		
11.2 PMEM 驱动架构	341		

13.3.7 函数 skel_write()和 skel_write_bulk_callback()	444	16.2 硬件抽象层架构	513
13.3.8 获取 USB 的接口	446	16.3 JNI 层架构	514
13.3.9 释放不需要的资源	447	16.4 Java 层架构	515
13.3.10 字符设备函数	448	16.5 实战演练——移植振动器系统	519
13.3.11 读取的数据量	449	16.5.1 移植振动器驱动程序	519
13.4 实战演练	450	16.5.2 实现硬件抽象层	520
13.4.1 移植 USB Gadget 驱动	450	16.6 实战演练——在 MSM 平台实现 振动器驱动	520
13.4.2 移植 USB HOST 驱动	452	16.7 实战演练——在 MTK 平台实现 振动器驱动	523
第 14 章 Time Device 驱动	453	16.8 实战演练——移植振动器驱动	526
14.1 Timed Output 驱动架构	453	第 17 章 输入系统驱动	527
14.1.1 设备类	453	17.1 输入系统架构分析	527
14.1.2 分析 Timed Output 驱动的具体实现	458	17.2 移植输入系统驱动的方法	528
14.1.3 实战演练——实现设备的读写操作	460	17.3 Input (输入) 系统驱动详解	529
14.2 Timed Gpio 驱动架构	461	17.3.1 分析头文件	529
14.2.1 分析文件 timed_gpio.h	462	17.3.2 分析核心文件 input.c	533
14.2.2 分析文件 timed_gpio.c	462	17.3.3 event 机制详解	548
第 15 章 警报器系统驱动 Alarm	467	17.4 硬件抽象层详解	551
15.1 Alarm 系统基础	467	17.4.1 处理用户空间	551
15.1.1 Alarm 层次结构介绍	467	17.4.2 定义按键的字符映射关系	555
15.1.2 需要移植的内容	468	17.4.3 KL 格式的按键布局文件	556
15.2 RTC 驱动程序架构	468	17.4.4 KCM 格式的按键字符映射文件	557
15.3 Alarm 驱动架构	469	17.4.5 分析文件 EventHub.cpp	558
15.3.1 分析文件 android_alarm.h	469	17.5 实战演练	561
15.3.2 分析文件 alarm.c	471	17.5.1 在内置模拟器中实现输入驱动	562
15.3.3 分析文件 alarm-dev.c	483	17.5.2 在 MSM 高通处理器中实现输入驱动	562
15.4 JNI 层详解	491	17.5.3 在 Zoom 平台中实现输入驱动	571
15.5 Java 层详解	493	第 18 章 LCD 显示驱动	573
15.5.1 分析 AlarmManagerService 类	493	18.1 LCD 系统介绍	573
15.5.2 分析 AlarmManager 类	501	18.2 FrameBuffer 内核层详解	573
15.6 模拟器环境的具体实现	503	18.2.1 分析接口文件 fb.h	574
15.7 实战演练	504	18.2.2 内核实现文件	577
15.7.1 编写 PCF8563 芯片的 RTC 驱动程序	504	18.3 硬件抽象层详解	600
15.7.2 在 2440 移植 RTC 驱动程序	507	18.3.1 Galloc 模块的头文件	601
15.7.3 在 mini2440 开发板上的移植	508	18.3.2 硬件帧缓冲区	603
15.7.4 实现一个秒表定时器	509	18.3.3 显示缓冲区的分配	604
第 16 章 振动器驱动架构和移植	512	18.3.4 显示缓冲映射	605
16.1 振动器系统架构	512		

18.3.5 分析管理库文件 LayerBuffer.cpp.....	606	19.4.2 实现硬件抽象层.....	681
18.4 Goldfish 中的 FrameBuffer 驱动程序 详解.....	607	19.5 实战演练——在 OSS 平台实现 Audio 驱动.....	684
18.5 使用 Gralloc 模块的驱动程序.....	610	19.5.1 OSS 驱动基础.....	685
18.5.1 文件 gralloc.cpp.....	611	19.5.2 函数 mixer().....	685
18.5.2 文件 mapper.cpp.....	614	19.6 实战演练——在 ALSA 平台实现 Audio 系统.....	692
18.5.3 文件 framebuffer.cpp.....	615	19.6.1 注册音频设备和音频驱动.....	692
18.6 MSM 高通处理器中的显示驱动.....	620	19.6.2 在 Android 中使用 ALSA 声卡.....	693
18.6.1 msm fb 设备的文件操作函数接口.....	621	19.6.3 在 OMAP 平台移植 Android 的 ALSA 声卡驱动.....	701
18.6.2 高通 msm fb 的 driver 接口.....	621	19.6.4 基于 ARM 的 AC97 音频驱动.....	704
18.6.3 特殊的 ioctl.....	621	第 20 章 Overlay 系统驱动详解.....	710
18.7 MSM 中的 Gralloc 驱动程序详解.....	623	20.1 视频输出系统结构.....	710
18.7.1 文件 gralloc.cpp.....	623	20.2 移植 Overlay 系统.....	711
18.7.2 文件 framebuffer.cpp.....	624	20.3 硬件抽象层详解.....	711
18.7.3 文件 gralloc.cpp.....	627	20.3.1 Overlay 系统硬件抽象层的接口.....	711
18.8 OMAP 处理器中的显示驱动实现.....	630	20.3.2 实现 Overlay 系统的硬件抽象层.....	714
18.8.1 文件 omapfb-main.c.....	631	20.3.3 实现 Overlay 接口.....	714
18.8.2 文件 omapfb.h.....	633	20.4 实现 Overlay 硬件抽象层.....	715
18.9 实战演练.....	633	20.5 实战演练——在 OMAP 平台实现 Overlay 系统.....	717
18.9.1 S3C2440 上的 LCD 驱动.....	633	20.5.1 实现输出视频驱动程序.....	717
18.9.2 编写访问 FrameBuffer 设备文件的驱动.....	658	20.5.2 实现 Overlay 硬件抽象层.....	719
18.9.3 在 S3C6410 下移植 FrameBuffer 驱动.....	659	20.6 实战演练——在系统层调用 Overlay HAL.....	724
第 19 章 音频系统驱动.....	664	20.6.1 测试文件.....	724
19.1 音频系统架构基础.....	664	20.6.2 在 Android 系统中创建 Overlay.....	725
19.1.1 层次说明.....	665	20.6.3 管理 Overlay HAL 模块.....	726
19.1.2 Media 库中的 Audio 框架.....	665	20.6.4 S3C6410 Android Overlay 的测试代码.....	727
19.2 音频系统层次详解.....	668	第 21 章 照相机驱动.....	729
19.2.1 本地代码详解.....	668	21.1 Camera 系统的结构.....	729
19.2.2 JNI 代码详解.....	670	21.1.1 Java 程序部分.....	731
19.2.3 Java 层代码详解.....	671	21.1.2 Camera 的 Java 本地调用部分.....	731
19.3 移植 Audio 系统.....	672	21.1.3 Camera 的本地库 libui.so.....	732
19.3.1 移植需要做的工作.....	672	21.1.4 Camera 服务 libcameraservice.so.....	733
19.3.2 硬件抽象层移植分析.....	672	21.2 移植 Camera 系统.....	737
19.3.3 AudioFlinger 中的 Audio 硬件抽象层.....	674	21.2.1 实现 V4L2 驱动.....	737
19.3.4 真正实现 Audio 硬件抽象层.....	679		
19.4 实战演练——在 MSM 平台实现 Audio 驱动.....	680		
19.4.1 实现 Audio 驱动程序.....	680		

21.2.2 实现硬件抽象层.....	744	22.3 低功耗蓝牙协议栈详解.....	767
21.3 实战演练——在 MSM 平台实现 Camera 驱动.....	747	22.3.1 低功耗蓝牙协议栈基础.....	767
21.4 实战演练——在 OMAP 平台实现 Camera 驱动.....	750	22.3.2 低功耗蓝牙 API 详解.....	768
21.5 Android 实现 S5PV210 FIMC 驱动.....	751	22.4 Android 中的 BlueDroid.....	769
第 22 章 蓝牙系统驱动.....	764	22.4.1 Android 系统中 BlueDroid 的架构.....	770
22.1 Android 系统中的蓝牙模块.....	764	22.4.2 Application Framework 层分析.....	770
22.2 分析蓝牙模块的源码.....	766	22.4.3 分析 Bluetooth System Service 层.....	778
22.2.1 初始化蓝牙芯片.....	766	22.4.4 JNI 层详解.....	778
22.2.2 蓝牙服务.....	766	22.4.5 HAL 硬件抽象层详解.....	783
22.2.3 管理蓝牙电源.....	767	22.5 Android 蓝牙模块的运作流程.....	783
		22.5.1 打开蓝牙设备.....	783
		22.5.2 搜索蓝牙.....	788
		22.5.3 传输 OPP 文件.....	793

第 1 篇



基础知识篇

- 第 1 章 Android 底层开发基础
- 第 2 章 Android 驱动开发基础
- 第 3 章 主流内核系统解析

第 1 章 Android 底层开发基础

2007 年, Google 公司推出了一款无与伦比的移动智能设备系统——Android, 这是一种建立在 Linux 基础之上为手机、平板等移动设备提供的软件解决方案。截至 2014 年 9 月, 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 82% 的份额, 已经成为了当今最受欢迎的智能设备系统之一。本章将引领读者一起来了解 Android 系统基本知识, 并详细讲解 Android 底层开发所必须具备的基础知识。

1.1 Android 系统介绍

Android 一词最早出现于法国作家利里亚当在 1886 年发表的科幻小说《未来夏娃》中, 他将外表像人的机器起名为 Android, 如图 1-1 所示。

从 2008 年 HTC 和 Google 联手推出第一台 Android 手机 G1 开始, 在 2011 年第一季度, Android 在全球的市场份额首次超过塞班系统, 跃居全球第一。2011 年 11 月数据显示, Android 占据全球智能手机操作系统市场 52.5% 的份额, 中国市场占有率为 58%。如今 Android 已经成为了现在市面上主流的智能手机操作系统, 随处都可以见到这个绿色机器人的身影。



图 1-1 Android 机器人

Android 机型数量庞大, 简单易用, 相当自由的系统能让厂商和客户轻松地定制各样的 ROM、各种桌面部件和主题风格。其简单而华丽的界面得到了广大客户的认可, 对手机进行刷机也是不少 Android 用户所津津乐道的事情。

Android 版本数量较多, 市面上同时存在着 1.6、2.0、2.1、2.2、2.3、4.4.2、5.0 等各种版本的 Android 系统手机, 应用软件对各版本系统的兼容性对程序开发人员是一个不小的挑战。同时由于开发门槛低, 导致应用数量虽然很多但是应用质量参差不齐, 甚至出现不少恶意软件, 使一些用户受到损失。同时 Android 没有对各厂商在硬件上进行限制, 导致一些用户在低端机型上体验不佳。另一方面, 因为 Android 的应用主要使用 Java 语言开发, 其运行效率和硬件消耗一直是其他手机用户所非议的地方。

1.2 Android 系统架构介绍

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 1-2 所示。



图 1-2 Android 操作系统的组件结构图

本节将详细介绍 Android 操作系统的基本组件结构方面的知识。

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。Linux 2.6 是一种标准的技术，Linux 也是一个开放的操作系统。Android 对操作系统的使用包括核心和驱动程序两部分，Android 的 Linux 核心为标准的 Linux 2.6 内核，其更需要一些与移动设备相关的驱动程序，主要的驱动如下。

- ☑ 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 和 C++实现的，其中包含了如下各种库。

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

- ☑ 多媒体框架 (MediaFramework)：是 Android 多媒体的核心部分，基于 PacketVideo (即 PV) 的 OpenCORE，从功能上本库一共分为两部分，一部分是音频、视频的回放 (PlayBack)，另一部分则是音视频的记录 (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)：提供了对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。

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) 有如下区别。

- ☑ Java 虚拟机：执行的是 Java 标准的字节码 (Bytecode)。从 Android 5.0 开始，ART 将作为应用程序的默认运行环境。而 Java 虚拟机只是作为一个备选项，迟早会退出历史舞台。
- ☑ Dalvik：执行的是 Dalvik 可执行格式 (.dex) 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，更适合移动设备的特点。

1.2.3 应用程序框架 (Application Framework)

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 Views (视图组件)、List (列表)、Grid (栅格)、Text Box (文本框)、Button (按钮)，甚至还有一个嵌入式的 Web 浏览器。

一个基本的 Andoid 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。
- ☑ Content Provider：内容提供者。
- ☑ Intent and Intent Filter：意图和意图过滤器。

1.2.4 顶层应用程序 (Application)

Android 的应用程序主要是用户界面 (User Interface) 方面的，本层通常使用 Java 语言编写，其中还可以包含各种被放置在 res 目录中的资源文件。Java 程序和相关资源在经过编译后，会生成一个 APK 包。Android 本身提供了主屏幕 (Home)、联系人 (Contact)、电话 (Phone)、浏览器 (Browsers) 等众多的核心应用，同时应用程序的开发者还可以使用应用程序框架层的 API 实现自己的程序，这也是 Android 开源的巨大潜力体现。