

资深 Android 系统工程师和软件开发工程师倾力打造 Android 专题书籍  
知识框架完整、条理清晰、理论和实战相结合  
深度解析 Android 手机定制开发中的重点和难点 Telephony 通信模块

杨青平  
◎ 著



# Android Telephony

## 原理解析与开发指南

Android Telephony  
Principle Analysis and Development Guide

 中国工信出版集团

 人民邮电出版社  
POSTS & TELECOM PRESS

杨青平◎著



# Android Telephony

## 原理解析与开发指南

Android Telephony  
Principle Analysis and Development Guide

人民邮电出版社

北京

## 图书在版编目 (C I P) 数据

Android Telephony原理解析与开发指南 / 杨青平著

— 北京 : 人民邮电出版社, 2018. 9

ISBN 978-7-115-48915-9

I. ①A… II. ①杨… 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182123号

## 内 容 提 要

随着 Android 平台的应用越来越广泛, 越来越多的人加入到 Android 系统的定制研发中来。Android 的基本通信功能是 Android 系统定制的核心模块, 本书主要围绕 Android Telephony 关键业务流程展开, 从接打电话、网络服务、数据上网三方面解析 Telephony。

全书共 10 章, 主要内容包括初识 Android、搭建 Android 源代码编译调试环境、深入解析通话流程、详解 Telecom、详解 TeleService、Voice Call 语音通话模型、ServiceState 网络服务、Data Call 移动数据业务、SMS & MMS 业务、Radio Interface Layer。

本书适合计算机科学技术、信息技术、通信工程、软件工程等专业的研究生、本科学士、高职高专学生使用。

- 
- ◆ 著 杨青平
  - 责任编辑 祝智敏
  - 责任印制 马振武
  - ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号  
邮编 100164 电子邮件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京圣夫亚美印刷有限公司印刷
  - ◆ 开本: 787×1092 1/16  
印张: 19 2018 年 9 月第 1 版  
字数: 519 千字 2018 年 9 月北京第 1 次印刷

---

定价: 59.80 元

读者服务热线: (010) 81055256 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

广告经营许可证: 京东工商广登字 20170147 号

# 前 言

随着智能手机的日益普及，移动互联网正深刻改变着我们的日常生活，iOS 和 Android 两大移动操作系统功不可没。由于 Android 的开源，各大厂商纷纷通过定制 Android 系统降低研发成本和周期，从而快速推出智能产品。

Telephony 作为 Android 系统的核心业务，包括接打电话、手机上网、短信、彩信等应用场景，也是终端用户使用最多、最频繁的业务。因此，在定制 Android 系统的开发过程中，对 Telephony 业务的定制和修改也非常多，如优化通话界面、过滤骚扰短信和电话，分析短信内容和优化展示关键信息等，既能方便用户操作和使用，提升用户体验，也能增加广告收入。

Android Telephony 业务跨度大，涉及多个层之间的交互：应用层、系统框架层、HAL 硬件抽象层和 BP Modem。理解和掌握 Android Telephony 业务和技术，方能更好地完成 Android 系统定制、业务扩展和数据挖掘。

本书主要特点如下。

## 1. 解析源代码并总结 Android Telephony 业务模型、设计原理和系统架构

Android 的源代码汇集了大量 Google 工程师的设计思想和理念。本书对 Telephony 源代码中的关键设计模式、设计原理和实现方式做了详细的分析和总结，可帮助读者拓展思路和加强训练，以提升软件设计水平和编码能力。

本书由浅入深地详细讲解了搭建 Android 编译环境、分析源代码并推导出 Telephony 业务模型，可帮助读者提升 Ubuntu 系统动手能力、Java 语言编程能力、Android 开发能力、UML 阅读能力，理解和掌握常用的设计模式，以及 Android 系统平台定制的核心工作。

## 2. 合理、有效的组织

本书以 Telephony 关键业务流程为主线，分析和总结主要业务、核心实现和业务模型。首先分析和总结 Telephony 业务模型中 Voice Call 语音通话、ServiceState 网络服务、Data Call 移动数据和 SMS&MMS 的关键业务流程和核心实现机制，然后扩展 Telephony Phone、Voice Call、Data Call 等关键业务模型，同时输出时序图、类图、状态图等 UML 图和架构图，以帮助读者理解和掌握 Telephony 业务。

## 3. 内容充实、实用并结合实践

本书紧紧围绕 Android Telephony 关键业务展开，主要包括 Voice Call 语音通话、ServiceState 网络服务、Data Call 移动数据和 SMS&MMS 中需要优化和定制的关键业务，以及 Radio Interface Layer（无线通信接口抽象层）的消息处理机制和系统架构。

本书总结的 Telephony 业务模型可在华为 Nexus 6P 手机上进行运行和验证，使读者对 Telephony 业务和实现有更加直观的认识和理解，并掌握调试 Android 代码的方法和技巧。

尽管作者在写作过程中力求准确、完善，但书中不妥或错误之处仍在所难免，殷切希望广大读者批评指正！恳请读者一旦发现错误，于百忙之中及时与作者联系，以便尽快更正，作者将不胜感激。

E-mail: android\_tele@163.com。

杨青平

2018年4月

# 目 录

第 1 章 初识 Android	1
1.1 智能手机的系统结构	1
1.2 Android 系统架构	2
1.2.1 应用层	3
1.2.2 应用框架层	3
1.2.3 系统运行库层	3
1.2.4 核心层	4
1.3 Android Telephony 框架结构	5
1.3.1 系统运行库层的 HAL	6
1.3.2 简析 HAL 结构	6
1.3.3 Android 为什么引入 HAL	7
1.3.4 Android 中 HAL 的运行结构	7
本章小结	8
第 2 章 搭建 Android 源代码编译调试环境	9
2.1 Ubuntu Linux 操作系统及工具安装	10
2.1.1 PC 配置建议	10
2.1.2 Ubuntu 安装光盘制作	10
2.1.3 Ubuntu 安装过程	10
2.1.4 安装 OpenJDK	12
2.1.5 Ubuntu 系统工具包更新升级	13
2.2 Android 源代码下载及编译过程	13
2.2.1 工作目录设置	13
2.2.2 源代码下载	13
2.2.3 开始编译 Android 源代码	14
2.2.4 编译单个模块	16
2.3 Android Studio 及 SDK	17
2.3.1 下载和配置 Android Studio	17
2.3.2 Android SDK 下载及配置和使用	17
2.3.3 使用 Android SDK 启动 Android 虚拟设备	19
2.3.4 Android 调试工具 adb 的使用方法	20
2.3.5 相关技巧汇总	20
2.4 在 Google 手机上调试 Android 源码	21
2.4.1 Google 手机对应编译选项	21

2.4.2	Google 手机刷入工厂镜像	21
2.4.3	编译本地镜像并刷入 Google 手机	22
2.4.4	Google 手机上调试 Android 源码	25
2.4.5	关键问题总结	26
	本章小结	27
<b>第 3 章</b>	<b>深入解析通话流程</b>	<b>29</b>
3.1	拨号流程分析	29
3.1.1	打开 Nexus 6P 手机的拨号盘	30
3.1.2	进入拨号界面 DialtactsActivity	30
3.1.3	DialpadFragment 拨号盘	32
3.1.4	ITelecomService 接收拨号请求服务	33
3.1.5	CallsManager 拨号流程处理	35
3.1.6	IInCallService 服务的响应过程	40
3.1.7	继续分析 CallsManager.placeOutgoingCall	46
3.1.8	Telecom 应用拨号流程回顾与总结	50
3.1.9	IConnectionService 服务的响应过程	51
3.1.10	TelecomAdapter 接收消息回调	55
3.1.11	拨号流程总结	56
3.2	来电流程分析	57
3.2.1	分析 radio 来电日志	58
3.2.2	UNSOL_RESPONSE_CALL_STATE_CHANGED 消息处理	58
3.2.3	扩展 RegistrantList 消息处理机制	59
3.2.4	GsmCdmaCallTracker 消息处理	61
3.2.5	ITelecomService 处理来电消息	63
3.2.6	来电流程总结	66
3.3	通话总结	66
3.3.1	通话关键代码汇总	66
3.3.2	通话状态更新消息上报流程	68
3.3.3	控制通话消息下发流程	69
3.4	建立 Android 通话模型	70
	本章小结	71
<b>第 4 章</b>	<b>详解 Telecom</b>	<b>73</b>
4.1	Telecom 应用加载入口	73
4.1.1	TelecomManager 类核心逻辑分析	74
4.1.2	Telecom 应用代码汇总	76
4.1.3	ITelecomService 的 onBind 过程	77
4.1.4	第二个拨号入口	79
4.2	Telecom 交互模型	79



4.2.1	汇总 frameworks/base/telecomm 代码	80
4.2.2	绑定 InCallService 机制	81
4.2.3	绑定 IConnectionService 机制	82
4.2.4	演进 Telecom 交互模型	85
4.3	核心 Listener 回调消息处理	86
4.3.1	CallsManagerListener	86
4.3.2	Call.Listener	88
4.3.3	CreateConnectionResponse	90
4.3.4	总结 Listener 消息	90
4.4	扩展 CallsManager	92
4.4.1	记录通话日志	92
4.4.2	耳机 Hook 事件	93
4.4.3	通知栏信息同步	93
	本章小结	94
<b>第 5 章 详解 TeleService</b>		95
5.1	加载过程分析	95
5.1.1	应用基本信息	96
5.1.2	PhoneGlobals.onCreate	97
5.1.3	TelephonyGlobals.onCreate	98
5.2	Telephony Phone	98
5.2.1	GsmCdmaPhone	99
5.2.2	Composition (组合) 关系	101
5.2.3	Facade Pattern	102
5.2.4	Handler 消息处理机制	103
5.3	扩展 PhoneAccount	105
5.3.1	PhoneAccount 初始化过程	105
5.3.2	PhoneAccount 注册响应	108
5.3.3	PhoneAccount 在拨号流程中的作用分析	109
5.3.4	小结	112
5.4	TeleService 服务	113
5.4.1	phone 系统服务	113
5.4.2	isub 系统服务	115
5.4.3	IConnectionService 应用服务	118
	本章小结	123
<b>第 6 章 Voice Call 语音通话模型</b>		125
6.1	详解 GsmCdmaCallTracker	125
6.1.1	代码结构解析	126
6.1.2	Handler 消息处理方式	127

6.1.3	与 RILJ 对象的交互机制	130
6.2	handlePollCalls 方法	134
6.2.1	准备阶段	134
6.2.2	更新通话相关信息	135
6.2.3	发出通知	140
6.2.4	更新 mState	141
6.3	通话管理模型分析	142
6.3.1	GsmCdmaCall	143
6.3.2	GsmCdmaConnection	143
6.3.3	DriverCall、Call、Connection	146
6.4	补充通话连接断开处理机制	149
6.4.1	本地主动挂断通话	149
6.4.2	远端断开通话连接	152
6.5	区分 Connection	154
6.6	扩展 InCallUi	155
6.6.1	初始化过程	155
6.6.2	addCall	158
6.6.3	InCallUi 通话界面	160
6.6.4	updateCall	165
6.7	验证 Call 运行模型	166
6.7.1	Telephony Voice Call	167
6.7.2	Telecom Call	170
6.7.3	InCallUi Call	171
	本章小结	173
<b>第 7 章 ServiceState 网络服务</b>		<b>175</b>
7.1	ServiceState	176
7.1.1	ServiceState 类的本质	176
7.1.2	关键常量信息	177
7.1.3	关键属性	177
7.1.4	关键方法	178
7.2	ServiceStateTracker 运行机制详解	179
7.2.1	核心类图	179
7.2.2	代码结构	180
7.2.3	Handler 消息处理机制	181
7.2.4	与 RILJ 对象的交互机制	184
7.3	handlePollStateResult 方法	186
7.3.1	异常处理	186
7.3.2	handlePollStateResultMessage	187
7.3.3	继续更新 mNewSS	190



7.3.4	完成收尾工作	191
7.4	***测试工具	193
7.4.1	网络服务信息	194
7.4.2	扩展 ITelephonyRegistry	196
7.4.3	展示小区信息	197
7.4.4	小区信息更新源头	198
7.4.5	信号强度实时变化	199
7.5	飞行模式	201
7.5.1	飞行模式开启关闭入口逻辑	201
7.5.2	Radio 模块开启关闭	202
7.5.3	WiFi 模块开启关闭	202
7.5.4	蓝牙模块开启关闭	202
7.6	扩展 SIM 卡业务	203
7.6.1	SIM 卡业务分析	203
7.6.2	驻网过程分析	204
7.6.3	SoftSim 业务实现分析	205
	本章小结	206
<b>第 8 章</b>	<b>Data Call 移动数据业务</b>	<b>207</b>
8.1	DcTracker 初始化过程	207
8.1.1	Handler 消息注册	208
8.1.2	初始化 ApnContext	208
8.1.3	认识 APN	210
8.1.4	创建 DcController	212
8.1.5	注册 Observer	213
8.1.6	广播接收器	213
8.1.7	加载 ApnSetting	213
8.2	解析 StateMachine	215
8.2.1	State 设计模式	215
8.2.2	StateMachine 核心类	215
8.2.3	初始化流程	216
8.2.4	运行流程	217
8.2.5	小结	218
8.3	DataConnection	219
8.3.1	关键属性	220
8.3.2	关键方法	220
8.3.3	StateMachine 初始化流程	221
8.4	开启移动数据业务	222
8.4.1	流程分析	222
8.4.2	前置条件分析	227

8.4.3	DcActiveState 收尾工作	231
8.4.4	Suspend 挂起状态	232
8.4.5	查看手机上网基本信息	232
8.5	关闭移动数据业务	233
8.6	DataConnection 状态转换	233
8.7	获取 Android 手机上网数据包	234
8.7.1	使用 tcpdump 工具抓取 TCP/IP 数据包	234
8.7.2	使用 Wireshark 软件分析 TCP/IP 数据包	235
	本章小结	235
<b>第 9 章</b>	<b>SMS&amp;MMS 业务</b>	<b>236</b>
9.1	短信发送流程	236
9.1.1	进入短信应用	236
9.1.2	短信编辑界面	237
9.1.3	Action 处理机制	239
9.1.4	继续跟进短信发送流程	241
9.1.5	phone 进程中的短信发送流程	243
9.2	扩展短信发送业务	245
9.2.1	确认短信发送结果	245
9.2.2	重发机制	246
9.2.3	状态报告	247
9.3	短信接收流程	247
9.3.1	RIL 接收短信消息	247
9.3.2	GsmInboundSmsHandler	248
9.3.3	Messaging 应用接收新短信	250
9.3.4	PDU	251
9.3.5	短信业务小结	252
9.4	彩信关键业务逻辑	253
9.4.1	彩信发送入口	253
9.4.2	imms 系统服务	254
9.4.3	彩信发送流程	255
9.4.4	Data Call	256
9.4.5	doHttp	259
9.4.6	接收彩信	259
9.4.7	MmsService 小结	260
	本章小结	261
<b>第 10 章</b>	<b>Radio Interface Layer</b>	<b>262</b>
10.1	解析 RILJ	263
10.1.1	认识 RIL 类	263

10.1.2	RILRequest	265
10.1.3	IRadio 关联的服务	266
10.1.4	RIL 消息分类	270
10.1.5	Solicited Request	270
10.1.6	Solicited Response	271
10.1.7	UnSolicited	274
10.2	详解 rild	274
10.2.1	RIL_startEventLoop	275
10.2.2	获取 RIL_RadioFunctions	275
10.2.3	注册 RIL_RadioFunctions	277
10.3	libril 初始化流程	278
10.3.1	RIL_startEventLoop	278
10.3.2	RIL_register	280
10.4	扩展 hal 接口	281
10.4.1	增加接口定义	282
10.4.2	验证生成的代码	282
10.4.3	实现新增接口	285
10.4.4	运行结果验证	286
10.5	RILC 运行机制	287
10.5.1	Solicited 消息	287
10.5.2	UnSolicited 消息	291
	本章小结	293

# 初识 Android

## 学习目标

- 学习智能手机基本硬件体系结构。
- 掌握基于 Linux Kernel 的 Android 系统分层架构。
- 掌握 Telephony 在 Android 系统中的结构。

Android 中文意思为“机器人”，中文译名为“安卓”，是谷歌公司于 2007 年 11 月 5 日发布的基于 Linux 平台的开源手机操作系统，其由操作系统、中间件、用户界面和应用软件组成，号称首个为移动终端打造的真正开放和完整的移动软件。谷歌公司通过与电信运营商、手机设备制造商、芯片开发商及其他有关方面结成深层次的合作伙伴关系，希望借助建立标准化、开放式的移动电话软件平台，在移动产业内形成一个开放式的生态系统。

从 2007 年至今，经过长时间的考验，Android 已经成为全球最热门的手机操作系统之一。本章主要从智能手机的基本硬件结构、Android 手机操作系统整体架构和 Android 的 Telephony 模块的体系结构三个方面逐步认识 Android，特别将 Android 手机操作系统平台下的 Telephony 模块作为本章讲解的重点内容。

## 1.1 智能手机的系统结构

Android 手机的基本硬件结构符合智能手机的基本硬件结构，我们要学习 Android 移动开发，首先需要了解智能手机的硬件系统基本结构。

随着通信领域的快速发展，移动终端的发展和变化也非常巨大，已经由原来单一的通话功能、短信功能，向彩信、数据上网、图像处理、音乐和多媒体方向演变。到目前为止，市面上的移动手机基本上可以分成两大类：一类是功能手机（Feature Phone）；另一类是智能手机（Smart Phone）。

这两类手机如何区分呢？智能手机具有传统手机的基本功能，如打电话、发短信、照相等。智能手机的特点：具有开放的操作系统、硬件和软件的可扩充性和支持第三方的二次开发。相对于功能手机，智能手机就像计算机一样，可通过安装第三方软件来扩展其功能和应用，因此，智能手机

越来越受到人们的青睐，已成为手机终端市场的一种潮流。

那么先来看看智能手机较多采用的硬件基本结构，如图 1-1 所示。

智能手机的基本硬件结构大多采用双处理器架构：主处理器和从处理器。主处理器运行开放式操作系统以及操作系统之上的各种应用，负责整个系统的控制；从处理器负责无线通信基本能力，主要包括 DBB ( Digital Baseband, 数字基带 ) 和 ABB ( Analog Baseband, 模拟基带 )，完成语音信号和数字信号调制解调、信道编码解码和无线 Modem 控制。

主处理器也叫 AP ( Application Processor, 应用处理器 )，从处理器也叫 BP ( Baseband Processor, 基带处理器 )，它们之间通过串口、总线或 USB 等方式进行通信。不同手机芯片生产厂家采用的集成方式都不一样，目前市面上仍以串口通信为主。

不难发现，在智能手机的基本硬件结构中，BP 部分只要再加一定的外围电路，如音频芯片、LCD 控制、摄像机控制器、扬声器、天线等，就是一个完整的普通手机的硬件结构。

**注意** 现在我们能区分功能手机与智能手机吗？回顾手机终端的发展历程，不难发现这样一条规律：随着手机芯片处理能力的提升、上网能力的扩展和发展（蓝牙、WiFi、3G 网络），手机应用得到非常广泛的扩大和发展。在智能手机的硬件设计上，采用处理能力比较强大的处理器作为 AP，来支持开放手机操作系统及操作系统之上的扩展应用，由此可见智能手机发展的趋势和方向。

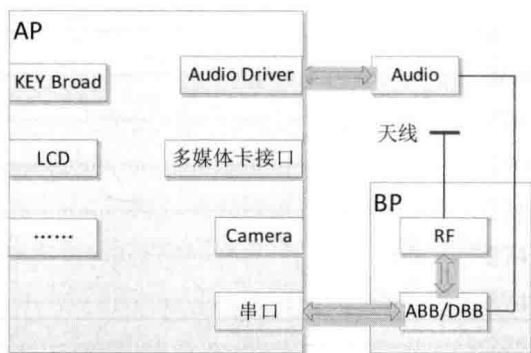


图 1-1 智能手机硬件结构图

## 1.2 Android 系统架构

前面学习了智能手机的基本硬件结构，可通过功能手机与智能手机的特点和区别从本质上去认识它们。Android 作为一款运行在 AP 上的开源智能手机操作系统，其系统架构是什么样的呢？我们先来看看图 1-2。

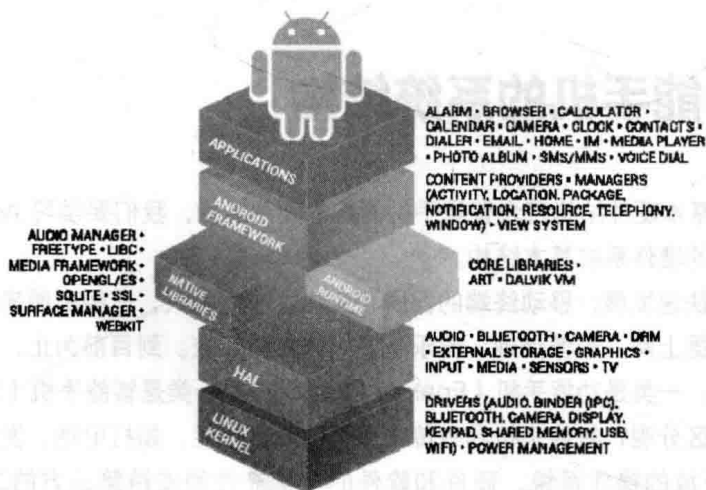


图 1-2 Android 系统架构

通过图 1-2 不难发现, Android 是一个分层的基于 Linux Kernel 的智能手机操作系统, 共分为四层, 从上到下依次是应用层 (Applications)、应用框架层 (Framework)、系统运行库层 (Libraries) 和核心层 (Linux Kernel), 下面将对这四层进行简要的分析和介绍。

## 1.2.1 应用层

Android 近几年的发展可谓是非常迅猛, 有一个很重要的原因, 那就是它的应用非常多。安卓市场已发布的软件个数和软件下载量目前仅次于苹果的应用商店, 并且保持着快速增长态势。只有这些第三方开发的应用 (如游戏、导航、播放器、桌面主题等) 日益丰富, 手机终端用户才能不断地发展和壮大, 而这些应用均在应用层运行。

应用层包括了各种 Android 应用程序, 这些应用程序是使用 Java 语言开发, 并运行在 Dalvik 虚拟机上, 处于 Android 系统架构中的第一层。在 Android 源码和 SDK 中, 谷歌公司已经捆绑和发布了一些核心应用及源代码, 如 Dialer、MMS、日历、谷歌地图、浏览器、联系人等。

## 1.2.2 应用框架层

如图 1-2 所示, Android 系统架构中的第二层是应用框架层, 是用 Java 语言实现和开发的。有了应用框架层, 开发者使用该层提供的 API 便可非常方便地完成访问设备硬件、获取位置信息、向状态栏添加通知消息、设置闹铃等操作, 而不必关心具体的底层实现机制和硬件实现方式。这样, 简化了 Android 应用开发者开发程序时的架构设计, 从而能够快速开发新的应用程序。

应用框架层是谷歌公司发布核心应用时所使用的 API 框架, 开发人员可以使用这些框架提供的 API 来快速开发自己的应用程序。下面是对 Android 中一些主要的组件的简要总结及相关说明。

- 视图 (View)

在 Android SDK (Software Development Kit, 软件开发工具包) 中介绍了其丰富的视图的使用方法及相关属性, 所有的 Android 应用程序都由这些视图构成, 主要包括列表 (List)、网格 (Grid)、文本框 (Text)、按钮 (Buttons) 等基础 Android 应用的界面控件。

- 资源管理器 (Resource Manager)

提供非代码资源转换和访问, 如本地字符串 (xml 文件配置)、图片和布局文件 (Layout File, 使用 xml 文件配置)。

- 通知管理器 (Notification Manager)

应用可以在状态栏中显示自定义的提示信息, 如新短信通知、未接来电通知、手机信号量通知等。

- Activity 管理器 (Activity Manager)

用来管理 Android 应用程序界面的生命周期 (onCreate 创建、onResume 显示、onPause 暂停、onStop 停止等), 一个手机屏幕界面可对应一个 Activity。

## 1.2.3 系统运行库层

如图 1-2 所示, Android 系统架构中的第三层为系统运行库层, 这一层主要包含了手机操作系统平台必备的 C/C++ 核心库、Dalvik 虚拟机运行环境和 HAL 子层。我们跳过 HAL, 先简单地介绍和分析 C/C++ 核心库和 Dalvik 虚拟机运行环境。



## 1. C/C++核心库

系统运行库层包含一个 C/C++库的集合，当使用 Android 应用框架的一些接口时，系统运行库层通过 C/C++核心库来支持对应的组件使用，使其能更好地为 Android 应用开发者服务。下面是一些主要的核心 C/C++库及其简要说明。

- libC (系统 C 库)

C 语言标准库，处于系统最底层的系统库，由 Linux 系统来调用。

- Media Framework (多媒体库)

Android 系统多媒体库，支持当前手机平台上主流的音频和视频格式播放和录制，以及静态图像。如 MPEG-4、MP3、AAC、JPG、PNG 等多媒体格式。

- SGL

2D 图形引擎库

- OpenGL

3D 效果的支持。

- SQLite

轻量级关系数据库引擎，可用来增、删、改、查通话记录、联系人等信息。

- WebKit

新式的 Web 浏览器引擎，支持当前非常流行的 HTML 5。

- SSL

基于 TCP/IP 网络协议，为数据安全通信提供支持。

## 2. Dalvik 虚拟机运行环境

系统运行库层包含了 Android Runtime，其核心为 Dalvik 虚拟机。每一个 Android 应用程序都运行在 Dalvik 虚拟机之上，且每一个应用程序都有自己独立运行的进程空间；Dalvik 虚拟机只执行 DEX 可执行文件。

DEX 格式是专为 Dalvik 设计的一种压缩格式，适合内存和处理器速度有限的系统。要生成 DEX 格式文件，首先通过 Java 程序编译生成 class 文件，然后通过 Android 提供的 dx 工具将 class 文件格式转换成 DEX 格式。

Dalvik 虚拟机的特性总结如下。

- 每一个 Android 应用运行在一个 Dalvik 虚拟机实例中，而每一个虚拟机实例都是一个独立的进程空间。
- 虚拟机的线程机制、内存分配和管理、Mutex (进程同步) 等的实现都依赖底层 Linux 操作系统。
- 所有 Android 应用的线程都对应一个 Linux 线程，因而虚拟机可以更多地使用 Linux 操作系统的线程调度和管理机制。



### 注意

因为 Android 的编程语言是 Java 的缘故，我们很容易将 Dalvik 虚拟机与 Java 虚拟机误认为是同一个东西。但 Dalvik 虚拟机并不是按照 Java 虚拟机的规范来实现的，两者并不兼容；它们之间最大的不同在于 Java 虚拟机运行的是 Java 字节码，而 Dalvik 虚拟机运行的是其专有的文件格式——DEX (Dalvik Executable) 文件。

## 1.2.4 核心层

Android 4.0 基于 Linux Kernel 3.0.8 提供核心系统服务，如文件管理、内存管理、进程管理、网



- Android Telephony 的业务应用跨越 AP 和 BP，AP 与 BP 相互通信，符合前面介绍的智能手机的硬件基本结构。
- Android 系统在 AP 上运行，而 Telephony 运行在 Linux Kernel 之上的用户空间。
- Android Telephony 也采用了分层结构的设计，共跨越了三层：应用层、应用框架层和系统运行库层，与 Android 操作系统整体分层结构保持一致；
- Android Telephony 从上到下共分三层：Telephony 应用、Telephony 框架、RIL (Radio Interface Layer, 无线通信接口层，主要位于系统运行库层的 HAL 中，什么是 HAL，接下来会详细介绍)。
- BP SoftWare 在 BP 上运行，主要负责实际的无线通信能力处理，不在本书讨论的范围。

### 1.3.1 系统运行库层的 HAL

HAL (Hardware Abstraction Layer, 硬件抽象层) 在 Linux 和 Windows 操作系统平台下有不同的实现方式。

Windows 下的 HAL 位于操作系统的最底层，它直接操作物理硬件设备，用来隔离与不同硬件相关的信息，为上层的操作系统和设备驱动程序提供一个统一接口，起到对硬件的抽象作用。这样更换硬件后编写硬件的驱动时，只要实现符合 HAL 定义的标准接口即可，而上层应用并不会受到影响，也不必关心具体实现的是什么硬件。

Linux 下的 HAL 与 Windows 下的 HAL 不太一样，HAL 并不是位于操作系统的最底层，直接操作硬件；相反，它位于操作系统核心层和驱动程序之上，是一个运行在用户空间中的服务程序。

### 1.3.2 简析 HAL 结构

通过前面的学习，我们知道 Android 是基于 Linux Kernel 的开源智能手机操作系统，所以这里重点介绍基于 Linux 下的 HAL 结构，就不再单独介绍 Windows 下的 HAL 结构。

要想知道 HAL 结构，先看看来源于 HAL 0.4.0 Specification 的框图，如图 1-4 所示。

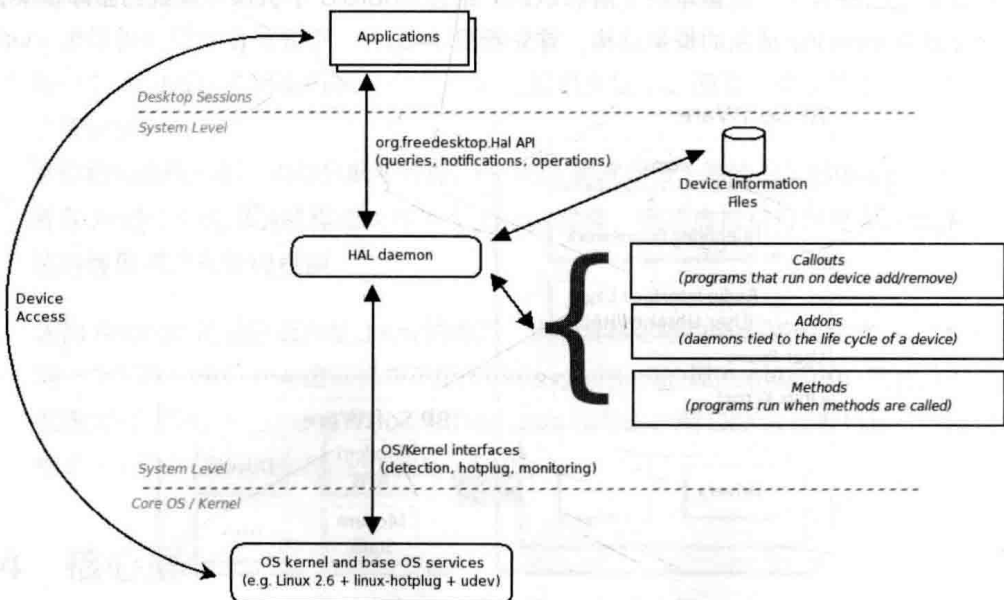


图 1-4 HAL 0.4.0 Specification