



Android

深度探索 (卷2)

系统应用源代码分析与 ROM 定制

李宁 编著

国内第一本介绍定制 ROM 的书

深度分析和讲解了 root、各类 ROM 的定制技术

分析了大量的 Android 系统应用源代码，读者可以任意定制 Android ROM

采用最流行的 CM Android 源代码进行分析，可支持上百种机型



Android

深度探索 (卷2)

**系统应用源代码分析与
ROM 定制**

李宁 编著

人民邮电出版社

北京

Android深度探索. 第2卷, 系统应用源代码分析与
ROM定制 / 李宁编著. — 北京 : 人民邮电出版社,
2015. 1

ISBN 978-7-115-36794-5

I. ①A… II. ①李… 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(2014)第206350号

内 容 提 要

本书专门介绍 Android 操作系统的编程, 全书分为两部分。第一部分主要介绍 Android 源代码和 Linux 内核源代码的下载和编译、Root 权限的提取、Android ROM 的制作和刷机、Recovery 的定制等。第二部分主要分析 Android 系统应用的实现原理和源代码。这些系统应用包括 Android 应用的安装和卸载管理、系统设置、系统设置内容提供者、电话与联系人管理、短信与彩信管理、Launcher2 和 NFC 后台服务程序。通过对这些 Android 系统应用的源代码分析, 会使读者定制出更完美的 Android ROM。

本书主要采用了 CM10.1 源代码进行讲解和分析, 使读者可以系统和完整地掌握定制 Android ROM 所需要的最新技术。无论读者是想找一份好工作, 还是想满足自己的 Geek (极客) 情结, 本书都是您的首选。

本书适合具备一定的开发经验 (最好有 Java 或 C/C++ 开发经验)、想学习 Android 和 Linux 底层开发的程序员使用, 也适合具备一定的 Android 开发经验, 想从事底层开发的编程爱好者使用。本书还适合作为相关培训学校的 Android 底层开发培训教材。

◆ 编 著 李 宁

责任编辑 张 涛

责任印制 彭志环

◆ 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>

北京鑫正大印刷有限公司印刷

◆ 开本: 787×1092 1/16

印张: 31.75

字数: 839 千字

印数: 1-3 500 册

2015 年 1 月第 1 版

2015 年 1 月北京第 1 次印刷

定价: 89.00 元 (附光盘)

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

广告经营许可证: 京崇工商广字第 0021 号

前言

为什么要写这本书

开源软件在很多年前就存在了，不过自从 Google 公司发布开源的 Android 以来，好像突然关注这个开源系统的人多了起来。而且与其他开源系统不同，很多 Android 的关注者不但使用 Android，还不断对 Android “动手动脚”。这些人通过不断修改 Android 源代码的各个部分，制作出了很多 Android 的衍生版本，也就是我们经常说的 ROM。

就目前而言，无论国外，还是国内；无论大企业，还是小公司，甚至是小团队和个人，都在疯狂地制作各种类型的 ROM。例如，有一定规模的 CM (Cyanogenmod) ROM、HTC ROM、三星 ROM 等；还有国内的小米 ROM (MIUI ROM)、点心 ROM、乐蛙 ROM，当然还有赶鸭子上架的老罗 ROM (更准确地说应该是锤子 ROM)。这些 ROM 各有特色，很难说哪个更有优势。

既然有这么多 ROM，可能很多搞 Android 开发的程序员按耐不住了。因为对于从没搞过 ROM 或对 ROM 一知半解的程序员来说，ROM 往往显得很神秘，而且很高深。所以，这些好奇的程序员中的绝大多数都曾想满足一下自己的好奇心，制作一款完全属于自己的 ROM (强烈的好奇心和求知欲望是人类的高贵品质，程序员尤为如此)。当然，部分想尝试做 ROM 的程序员还听说这类工作的薪水比较高，而且好找工作，这也许是很多程序员想学习做 ROM 的另外一个原因吧！不过在心动到行动的过程中，会发现从网上收集到的制作 ROM 的资料都是零散的，有些甚至是错误的，而且非常不系统。所以，这些想进入 ROM 世界的程序员再一次进入迷茫状态。

如果手捧本书的您正在读这段前言，说明迷茫状态快要结束了！为了满足有志于制作 ROM 的程序员们的好奇心。笔者特意编写了本书。意在系统地阐述如何从 Android 源代码制作一个完整的 ROM，以及各种刷机技术。并且可以更深层次地定制 ROM。例如，修改 Android Home 应用 Launcher2，使其更符合自己的苛刻要求。

本书的内容

通常定制 ROM 分为应用层和底层（主要指驱动层），不过，由于大多数 ROM 是基于 CM ROM 的，而 CM ROM 已经将底层的驱动适配得非常好了，支持目前大多数主流机型。而且由于国内大多数第三方的 ROM 并没有自己适配底层驱动，所以，本书将把主要精力放在定制应用层上。

关于应用层的定制通常分为 Android 系统应用和 Framework。前者主要指直接面向用户的 Android 应用，例如，Home 应用 (Launcher2)、短信管理、系统设置等。而后者主要面对程序员群体。例如，为 ROM 增加某些特有的 API，以及修改原有的 API，使其满足某些特殊的要求。由于篇幅所限。本书将主要讨论 Android 系统应用的定制。在《Android 深度探索》系列的后续著作

中会继续讨论 Framework 的定制。

本书分为如下两部分。

(1) Android ROM 的制作和刷机（第 1 章～第 5 章）。

(2) Android 系统应用源代码分析和定制（第 6 章～第 15 章）。

其中第一部分主要包括如下几个方面。

(1) 开发环境搭建以及 Android 源代码（官方 Android 源代码和 CM Android 源代码）和 Linux 内核源代码的下载和编译。

(2) Bootloader 和 Recovery ROM 的制作和刷机。

(3) 提取 Root 权限的原理和实践（包括 am 命令详解、Superuser 和 su 的源代码分析等）。

(4) ROM 包含的各种镜像（system.img、boot.img、recovery.img 等）的修改、制作和刷机技巧。这里的制作主要指从 Android 源代码制作。

(5) Recovery ROM 的核心：Edify 语言。

(6) 集成第三方的 APK 程序（包括 Google Services Framework 和 Google Play）。

(7) Recovery 的原理和定制（包括为 Recovery 增加新功能，汉化 Recovery 等）。

第二部分主要包括如下内容。

(1) 如何开发和测试 Android 系统应用。

(2) Android 应用程序的安装、卸载原理和 PackageInstaller 代码分析。

(3) 系统设置的各种功能的实现原理和源代码分析。

(4) 系统内容提供者的源代码分析。

(5) 电话与联系人的实现原理和源代码分析。

(6) 短信与彩信管理的实现原理和源代码分析。

(7) Android Home 应用（Launcher2）的实现原理和源代码分析。

(8) 近场通信（NFC）后端服务程序的实现原理和源代码分析。

要注意的是，本书将以 CM 10.1（Android 4.x）源代码作为基础进行讲解。

本书的特点

(1) 系统地阐述了 Android ROM 的制作和刷机过程。

(2) 全书使用较新的 CM10.1（Android 4.x）。

(3) 完全采用 Ubuntu Linux12.04 LTS 作为实验环境

(4) 分析了 Android 中主要的系统应用的实现原理和源代码，使读者充分掌握定制 Android 系统应用的方法。

(5) 分析了制作 ROM 过程中涉及的核心技术的实现原理，例如，Root 权限的提取、Recovery 定制等。使读者可以制作出更酷的 Android ROM。

(6) 由于本书分析了大量的源代码，所以，还详细介绍了分析源代码的工具和一些技巧。

读者对象

❑ 从事 Android 应用开发，但想进入 Android 底层开发领域的程序员。

❑ 对定制 Android ROM 感兴趣的程序员。

❑ 想进一步提高 Android 底层开发技术和实践能力的程序员。

❑ 开设 Android 底层开发课程的大专院校和培训机构。

❑ 想成为 Geek（极客），但苦于没有人指导的 Android 爱好者。

源代码和工具下载

读者可以到 <http://pan.baidu.com/s/19UeDO> 下载 CM10.1 (Android 4.x) 的源代码。本书涉及的其他源代码和工具都可以在随书光盘中找到。另外可以到 <http://blog.csdn.net/nokiaguy> 下载光盘中的内容。

相关的视频课程为:http://edu.51cto.com/lecturer/user_id-974126.html。

勘误和支持

由于作者的水平有限,编写时间仓促,书中难免会出现一些错误或者不准确的地方,恳请读者批评指正。如有问题或建议,请发送至 techcast@126.com 或在新浪微博 (<http://weibo.com/638012593>) 上留言。非常期待能够得到你们的真挚反馈。编辑联系邮箱: zhangtao@ptpress.com.cn。

致谢

感谢所有在本书写作过程中给予我指导、帮助和鼓励的朋友,尤其是人民邮电出版社的编辑张涛,不仅对本书提出了宝贵的写作建议,而且还对本书进行了仔细的审阅。

感谢一直以来信任、鼓励、支持我的家人和朋友。

感谢 eoeandroid、移动开发者社区的朋友对我技术上的帮助。

谨以此书献给我最亲爱的家人,以及众多热爱 Android 的朋友们!

编者

目 录

第 1 章 学习前的准备工作	1	第 2 章 提取 ROOT 权限	21
1.1 安装和配置 JDK	1	2.1 什么是 ROM	21
1.2 安装和配置 Android 开发环境	1	2.2 ROM 的种类	22
1.3 官方 Android 源代码	2	2.3 刷官方的 ROM	22
1.3.1 下载 Android 源代码	2	2.3.1 下载官方的 ROM	23
1.3.2 编译 Android 源代码	3	2.3.2 用无人值守方式刷 ROM	23
1.4 官方 Linux 内核源代码	4	2.3.3 分别刷 ROM 包含的各种 镜像文件	24
1.4.1 Linux 内核支持的 Android 设备	4	2.4 Android 手机获取 ROOT 权限的 必要性	26
1.4.2 下载 Linux 内核源代码	5	2.5 提取 ROOT 权限的原理	26
1.4.3 查看 Linux 内核的分支	6	2.6 用极客的方式提取 ROOT 权限	27
1.4.4 看看 Google 公司的人在做什么	6	2.6.1 提取 ROOT 权限的步骤	27
1.4.5 获取 Linux 内核的配置 文件	8	2.6.2 需要一个很酷的 recovery	28
1.4.6 安装交叉编译器	9	2.6.3 su 命令源代码分析	30
1.4.7 编译 Linux 内核源代码	10	2.6.4 制作第一个 Recovery 刷机 包 (编写 updater-script 脚本 文件)	32
1.5 Cyanogenmod(CM)源代码	10	2.6.5 首次通过 DIY 方式提取 ROOT 权限	35
1.5.1 什么是 CM	10	2.6.6 上传 Android 应用到 /system/app 目录	35
1.5.2 CM 支持哪些 Android 设备	11	2.7 小结	36
1.5.3 与 Android 设备对应的 Codename 和 CM 版本	12	第 3 章 Root 权限的安全屏障	37
1.5.4 下载 CM Android 源代码	13	3.1 通过 su 提取 Root 权限的安全 隐患	37
1.5.5 下载经过 CM 适配的 Linux 内核源代码	14	3.2 Android 应用申请 Root 权限	38
1.5.6 编译 Android 源代码生成 Recovery ROM	14	3.3 Superuser.apk 为什么不见了	39
1.5.7 单独编译 CM Linux 内核源 代码	17	3.4 申请 Root 权限为什么失灵了	41
1.5.8 刷机! 刷机!	18	3.5 CM ROM 如何为 Root 权限增加 安全屏障	42
1.5.9 下载现成的 CM ROM	19	3.6 ADB Shell 动作管理命令 (am)	44
1.6 小结	20	3.6.1 显示窗口 (Activity)	44
		3.6.2 发送广播 (Broadcast)	45

3.6.3 开始服务 (Service)	46	4.2.3 用户数据镜像 (userdata.img)	84
3.7 su 实现原理及源代码分析	46	4.2.4 内存磁盘镜像 (ramdisk.img)	87
3.7.1 su 进行 Root 授权的处理 流程	46	4.2.5 Linux 内核镜像 (boot.img)	88
3.7.2 初始化调用者数据	50	4.2.6 制作已经有 ROOT 权限的 ROM	89
3.7.3 初始化路径	53	4.2.7 Recovery 镜像 (recovery.img)	90
3.7.4 动态宏定义	54	4.2.8 缓存镜像 (cache.img)	91
3.7.5 检测数据库	56	4.2.9 制作完美的 Bootloader ROM	91
3.7.6 创建 LocalSocket 服务	57	4.3 直接从 CM 源代码制作 ROM	93
3.7.7 显示 “Root 授权” 窗口	58	4.3.1 制作 Bootloader ROM	94
3.7.8 等待 Superuser 连接 LocalSocket 服务	59	4.3.2 制作 Recovery ROM	95
3.7.9 向 Superuser 传输调用者 信息	60	4.4 Edify 语言	98
3.7.10 接收用户选择的 “Root 授权” 策略	61	4.4.1 Edify 语言概述	98
3.7.11 允许和拒绝 “Root 授权”	62	4.4.2 测试 Edify 脚本 (updater-script)	98
3.8 Superuser 的实现原理与源代码 分析	63	4.4.3 Edify 函数详解	99
3.8.1 “Root 授权” 警告窗口的 处理流程	63	4.4.4 Edify 语言的实现原理	106
3.8.2 设置 “Root 授权” 窗口的 控件	65	4.5 集成 Google Services Framework 和 Google Play	109
3.8.3 获取 “Root 授权” 持续 时间	68	4.6 在 CM ROM 中集成第三方 APK 程序	110
3.8.4 处理 “Root 授权” 策略	70	4.6.1 为什么不能直接复制 APK 和 ODEX 文件	110
3.8.5 读取 Su 协议数据	71	4.6.2 校验 odex 文件需要依赖 哪些 Library	111
3.8.6 获取和设置 “调用者” 信息	73	4.6.3 合并 APK 和 ODEX 文件	111
3.9 小结	74	4.7 小结	113
第 4 章 ROM 定制	75	第 5 章 Recovery 深度分析与定制	114
4.1 刷机的那些事	75	5.1 什么是 Recovery	114
4.1.1 Android 刷机的本质	75	5.2 Clockworkmod Recovery 源代码 解析	116
4.1.2 制作 ROM 的不同层次	77	5.2.1 如何分析 Recovery 源代码	116
4.1.3 为什么要刷机	78	5.2.2 显示主菜单	117
4.1.4 刷机带来的风险及其预防 措施	78	5.2.3 切换不同的菜单项	120
4.2 在官方 ROM 的基础上定制各种 镜像	79	5.2.4 选择菜单项	121
4.2.1 修改系统镜像 (system.img)	80	5.3 Recovery 支持的各种操作	122
4.2.2 定制开机动画	82		

5.3.1 重启和关闭 Android 设备	122	7.4.2 显示校验窗口	163
5.3.2 从 SD 卡刷 ROM	124	7.4.3 获取 Android 应用的权限列表	166
5.3.3 使用 sideload 模式刷 ROM	129	7.5 开始安装应用程序	168
5.3.4 清除指定文件系统中的数据	130	7.5.1 PackageInstaller 的安装原理	169
5.3.5 备份与恢复	133	7.5.2 静默安装 Android 应用	174
5.3.6 挂载 (mount)、卸载 (umount) 和格式化 (format) 文件系统	137	7.6 卸载 Android 应用	177
5.4 添加自己的 Recovery 菜单项	140	7.6.1 PackageInstaller 卸载 Android 应用前的确认	178
5.5 汉化 Recovery	142	7.6.2 卸载 Android 应用的原理	180
5.5.1 汉化 Recovery 的原理和步骤	142	7.6.3 静默卸载 Android 应用	182
5.5.2 初始化字体	143	7.7 小结	184
5.5.3 绘制中文字符	144	第 8 章 系统设置 (一)	185
5.6 小结	145	8.1 为系统设置添加新功能	185
第 6 章 Android 系统应用的开发与测试	146	8.1.1 系统设置的编译与权限	185
6.1 什么是 Android 系统应用	146	8.1.2 修改开机动画	186
6.2 为什么要研究 Android 系统应用	146	8.1.3 寻找 Settings 的入口点	191
6.3 如何编写 Android 系统应用	147	8.1.4 为 Settings 添加新的功能项	192
6.4 分析第一个 Android 系统应用: 计算器	149	8.1.5 迁移修改开机动画的窗口类	194
6.4.1 计算器应用 (Calculator) 的基本结构	149	8.2 Wi-Fi	195
6.4.2 编译、测试和调试 Calculator	152	8.2.1 如何为设置项添加 Switch 控件	195
6.4.3 允许其他 Android 应用通过 Calculator 计算表达式	152	8.2.2 为“修改开机动画”设置项添加 Switch 控件	199
6.5 小结	154	8.2.3 “关闭/打开” Wi-Fi	202
第 7 章 安装与卸载应用程序 (PackageInstaller)	155	8.2.4 系统信息与 SQLite 的 WAL 模式	204
7.1 分析源代码的第一步应该做什么	155	8.2.5 禁止飞行模式下单独开启 Wi-Fi	205
7.2 寻找 PackageInstaller 的突破口	157	8.2.6 系统设置内容提供者 (Settings Content Provider)	206
7.3 安装和卸载 Android 应用的全部方式	159	8.2.7 用广播方式设置 Switch 控件的状态	211
7.4 安装 Android 应用前的校验	160	8.2.8 搜索可用热点	212
7.4.1 安装 Android 应用的初始化工作	160	8.2.9 热点对象 (AccessPoint)	216
		8.2.10 获取可用热点的各种状态	223

8.2.11 连接可用热点	226	第 11 章 电话与联系人	306
8.3 蓝牙	227	11.1 拨号应用	306
8.3.1 “关闭/打开” 蓝牙	227	11.1.1 实现拨号盘 UI	306
8.3.2 蓝牙的本地状态	228	11.1.2 显示与删除电话号	310
8.3.3 蓝牙设置的架构与实现	229	11.1.3 用拨号盘输入电话号	312
8.3.4 “允许/禁止” 当前蓝牙 设备被搜索到	234	11.1.4 播放和停止按键提示音	314
8.4 流量使用情况	235	11.1.5 拨打电话	316
8.5 小结	241	11.1.6 电话号到底是如何 拨出的	320
第 9 章 系统设置 (二)	242	11.2 通话记录管理	325
9.1 声音	242	11.3 联系人管理	332
9.1.1 调整音量	242	11.3.1 初始化联系人列表	332
9.1.2 响铃模式	247	11.3.2 获取经常呼叫的联系人	334
9.1.3 手机铃声	249	11.3.3 获取所有的联系人	338
9.1.4 渐强铃声	251	11.3.4 来电黑名单	339
9.2 显示	253	11.4 小结	342
9.2.1 调整屏幕亮度	254	第 12 章 短信和彩信管理	343
9.2.2 壁纸设置 (动态扩展)	256	12.1 显示会话列表	343
9.2.3 屏幕休眠设置	258	12.1.1 主窗口类 ConversationList 和跨应用程序调用	343
9.3 存储	260	12.1.2 会话列表适配器 (ConversationList Adapter)	344
9.3.1 添加设置项	261	12.1.3 异步查询会话信息	346
9.3.2 检测存储空间	264	12.2 删除会话	349
9.4 语言与输入法	269	12.3 发送短信	354
9.4.1 改变当前语言	269	12.3.1 显示发送短信窗口	354
9.4.2 设置当前的输入法	273	12.3.2 发送短信	356
9.4.3 扩展输入法	274	12.3.3 监听发送短信状态	359
9.5 开发者选项	278	12.4 监听短信的收发	361
9.5.1 Root 授权	278	12.4.1 处理收发短信广播的 流程	362
9.5.2 启动 USB 调试	282	12.4.2 处理接收短信广播	363
9.5.3 启动网络 ADB 调试	284	12.4.3 处理发送短信广播	365
9.6 关于手机	287	12.5 浏览当前会话	368
9.6.1 手机的状态信息	287	12.6 小结	369
9.6.2 手机的其他信息	290	第 13 章 AndroidHome 应用: Launcher2 (一)	370
9.6.3 在线更新 (OTA)	291	13.1 Launcher2 的那些事	370
9.7 小结	292	13.2 初始化 Launcher Home UI	372
第 10 章 系统设置内容提供者 (SettingsProvider)	293	13.2.1 Launcher2 的主布局文件 (launcher.xml)	372
10.1 系统设置数据库创建与升级	293		
10.2 读写系统设置的数据 (settings.db)	297		
10.3 系统设置数据的备份和恢复	300		
10.4 小结	305		

13.2.2	初始化 Android 桌面	373	(DragLayer)	433
13.2.3	全局对象 LauncherApplication	376	14.2	桌面 UI
13.2.4	初始化桌面 UI 控制器	379	14.2.1	搜索框
13.2.5	装载桌面 UI 视图	381	14.2.2	拖放目标区域
13.2.6	任务装载器 (LoaderTask)	382	14.2.3	删除快捷方式和卸载 Android 应用
13.3	装载和绑定 Workspace	385	14.2.4	编辑快捷方式
13.3.1	根据不同类型装载桌面 视图	385	14.2.5	查看快捷方式和 AppWidget 的应用信息
13.3.2	装载默认的桌面 UI 数据	390	14.3	文件夹 (Folder)
13.3.3	添加默认桌面 UI 数据的 若干方法	398	14.3.1	文件夹的创建过程
13.3.4	从 favorites 表中提取和 分类桌面 UI 数据	402	14.3.2	将快捷方式拖入文件夹
13.3.5	绑定 Workspace	405	14.3.3	将快捷方式从文件夹中 拖出
13.3.6	回调方法	407	14.3.4	显示文件夹中的快捷方式 列表
13.3.7	绑定前的清理工作	408	14.4	应用程序列表
13.3.8	在 Android 桌面上添加 各种 UI 视图	408	14.5	壁纸设置
13.3.9	如何将快捷方式和文件夹 添加到 Android 桌面上	410	14.5.1	壁纸选择器
13.3.10	如何将 AppWidget 添加 到 Android 桌面上	412	14.5.2	壁纸设置与壁纸图像 来源
13.4	装载和绑定 Android 应用	413	14.6	广播接收器
13.4.1	装载和绑定 Android 应用的时机	413	14.6.1	安装快捷方式 (InstallShortcut Receiver)
13.4.2	一体化装载和绑定 Android 应用	416	14.6.2	卸载快捷方式 (UninstallShortcut Receiver)
13.4.3	隐藏和显示指定的 Android 应用	419	14.6.3	回复默认桌面 (PreloadReceiver)
13.4.4	隐藏和显示系统和 普通 Android 应用	422	14.7	内容提供者 (LauncherProvider)
13.4.5	仅绑定 Android 应用	424	14.8	小结
13.5	小结	425	第 15 章	近场通信 (NFC) 的实现
第 14 章	AndroidHome 应用: Launcher2 (二)	426	原理	482
14.1	Android 桌面	426	15.1	处理 NFC NDEF 消息的服务
14.1.1	桌面布局 (CellLayout)	426	15.2	处理 NFC 消息的回调接口
14.1.2	工作空间 (Workspace)	429	15.3	调用处理 NFC 消息的窗口 (Activity)
14.1.3	底座 (Hotseat)	432	15.4	手持设备与蓝牙传输
14.1.4	桌面左右滑动		15.5	继续尝试其他处理 NFC 消息的 可能性
			15.6	通过蓝牙传递 Uri
			15.7	小结

第1章 学习前的准备工作

在这一章主要学习如何安装和配置 Android 的底层开发环境，并且将详细介绍如何编译官方和 CM 的 Android 源代码和 Linux 内核源代码，最后会尝试将编译生成的 ROM 刷到手机上。本书所使用的开发环境是 Ubuntu Linux 12.04 LTS。建议读者使用 root 用户登录 Linux，这样可以拥有最高的权限，从而避免在使用的过程中提升用户权限的麻烦。本章使用的 Android 源代码版本是 4.2，Linux 内核源代码版本是 3.x。

1.1 安装和配置 JDK

无论是编译 Android 源代码，还是编写 Android 系统应用，都需要在 Java 环境下完成，所以必须在 Ubuntu Linux 下安装 JDK。对于开发 Android 应用来说，JDK5、JDK6 和 JDK7 都没问题，不过编译 Android 源代码要求在 JDK5 或 JDK6 下进行，所以本书使用了 JDK6 作为 Java 的编译和运行环境。同时 Google 公司官方要求，Linux 必须是 Ubuntu Linux 10.04 LTS 或以上版本，其他的 Linux 发行版可能无法运行脚本中的某些命令，所以本节会在 Ubuntu Linux 12.04 LTS 下安装 JDK6。

如果读者要使用 JDK6 的最新版本，可以到如下的地址下载。

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

为了方便读者，在随书光盘中已经提供了 JDK 的安装包 (jdk-6u33-linux-i586.bin)，读者可以直接在 Linux 终端中执行该安装文件，并按提示操作即可。最后需要将 JDK 安装路径的 bin 目录添加到 PATH 环境变量中，通常在 /root/.profile 文件中修改 PATH 环境变量。假设 JDK 的安装路径为 /root/jdk6，那么可以按着如下形式设置 PATH 环境变量。

```
export PATH=$PATH:/root/jdk6/bin
```

1.2 安装和配置 Android 开发环境

Android 开发环境主要包括 Eclipse、Android SDK、ADT 和 Android NDK。其中 Eclipse 将作为开发 Android 应用的 IDE；Android SDK 为应用开发包；ADT 为 Eclipse 的插件，用于开发 Android 应用；Android NDK 用于开发 C/C++ 程序，实际上就是标准的 Linux 动态链接库 (.so 文件)，这些文件可以嵌入到 Android 应用程序中 (APK 文件)，并通过 JNI 技术调用。

读者可以到如下的地址下载 Eclipse 的最新版。也可以使用随书光盘中带的 Eclipse (eclipse.7z)，这个 Eclipse 压缩文件包含了最新的 Eclipse4 (juno)，以及 ADT，所以直接解压即可使用，并不需要再次安装 ADT。

<http://www.eclipse.org/downloads>

如果不想单独安装 Eclipse、ADT 和 Android SDK, 也可以到如下地址下载 Android 开发环境的集成版。

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

该集成环境包括如下几部分。

- ☐ Eclipse IDE。
- ☐ ADT 插件。
- ☐ Android SDK 工具集。
- ☐ Android 平台工具集。
- ☐ 最新的 Android SDK。
- ☐ 最新的用于 Android 模拟器的 Android 系统镜像 (system.img)。

如果读者已经有了 Eclipse, 可以使用下面的 Url 在线安装 ADT。

<https://dl-ssl.google.com/android/eclipse>

或者到下面的地址下载 ADT 的离线安装包。

<http://developer.android.com/sdk/installing/installing-adt.html>

最后一个需要安装的是 Android NDK, 读者可以到如下地址下载 Android NDK 的最新版。

<http://developer.android.com/tools/sdk/ndk/index.html>

13 官方 Android 源代码

本书使用了 Android 的最新版本 4.2.2, 可能在读者阅读本书时 Google 公司又发布了更新的 Android 版本, 到时读者只要按着本节的方法下载最新版本并编译即可。

1.3.1 下载 Android 源代码

下载 Android 源代码之前需要先配置 Linux 环境, 其中环境配置主要就是安装 JDK, 以及对于不同 Linux 版本, 可能需要为某些 .so 文件建立符号链接, 当然, 还有一些其他的配置, 不过这些设置大多数时候都不需要, 详细的配置方式 Google 公司官方已经给出, 页面地址如下, 本节不再详细介绍。

<https://source.android.com/source/initializing.html>

配置完环境后就可以直接通过 git 下载 Android 源代码, 不过为了方便, Google 公司提供了一个 Python 脚本 repo, 如果网络不中断, repo 可以用无人值守的方式安装, 即使下载的过程中网络中断了, 再次执行 repo 脚本, 仍然会从中断处开始下载。读者可以使用下面的命令将 repo 脚本文件下载到 ~/bin 目录中, 并将 repo 文件的权限改成可执行。同时将 ~/bin 目录加到 PATH 环境变量中。

```
# mkdir ~/bin
# PATH=~/.bin:$PATH
# curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```



注意

前面的 4 行命令开始的 “#” 并不属于命令本身, 只是由于本书所有的命令都是在 Linux 终端中执行的, 而且 Linux 使用 root 用户登录的, 所以前面的终端提示符是 “#”, 为了表示命令是在终端下执行, 本书所有给出的命令前面都会加 “#”, 但要注意, 当前执行命令的路径不一定在根目录或 /root 目录, 所以 “#” 只表示需要在 Linux 终端下执行该命令, 并不表示当前执行命令的路径。执行路径会在命令前面的描述中提到。如果未提到, 表示可以在任何路径下执行这些命令。

下完 repo 命令后, 可以任意建立一个目录, 并进入该目录, 最后执行下面的命令设置要下载

的 Android 的最新版本。

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

如果要指定下载 Android 的某个版本, 需要在下载链接中专门指定, 例如下载 Android 4.2.2 可以使用下面的命令。

```
# repo init -u https://android.googlesource.com/platform/manifest -b android-4.2.2_r1
```

最后需要执行下面的命令下载指定的 Android 版本。

```
# repo sync
```

下载 Android 源代码的过程比较缓慢, 如果下载的过程中中断, 可再次执行 `repo sync` 命令即可。

1.3.2 编译 Android 源代码

下载完 Android 源代码后, 就可以对源代码进行编译了, 但在编译之前还需要在 Android 源代码根目录中执行如下的命令设置一些 Shell 函数。

```
# source build/envsetup.sh
```

如果执行成功, 根据 Android 版本的不同, 会输出类似如图 1-1 所示的信息。



```
root@dell-pc: /sources/android4.2.2
root@dell-pc:/sources/android4.2.2# source build/envsetup.sh
including device/generic/armv7-a-neon/vendorsetup.sh
including device/generic/armv7-a/vendorsetup.sh
including device/generic/mips/vendorsetup.sh
including device/generic/x86/vendorsetup.sh
including sdk/bash_completion/adb.bash
root@dell-pc:/sources/android4.2.2#
```

▲图 1-1 编译环境安装

在编译 Android 源代码之前还需要做最后一件事, 就是设置编译的目标, 也就是为哪些设备编译 Android 源代码。例如, 如果要让编译后的目标文件在 Android 模拟器或 Android 设备上运行, 可以执行下面的命令。

```
# lunch full-eng
```

如果读者有其他的需要, 还可以指定其他的目标, 直接执行 `lunch` 命令, 会显示当前 Android 支持的所有目标, 如图 1-2 所示。直接输入前面的序号即可选择相应的目标, 例如, `full_x86-eng` 适合于在 X86 架构的计算机上运行。



```
root@dell-pc: /sources/android4.2.2
root@dell-pc:/sources/android4.2.2# lunch
You're building on Linux
Lunch menu... pick a combo:
 1. full-eng
 2. full_x86-eng
 3. vbox_x86-eng
 4. full_mips-eng
 5. mini_armv7a_neon-userdebug
 6. mini_armv7a-userdebug
 7. mini_mips-userdebug
 8. mini_x86-userdebug
Which would you like? [full-eng]
```

▲图 1-2 选择目标

最后可以在 Android 源代码根目录直接执行 `make` 命令编译整个 Android 源代码, 接下来又是一段难熬的等待。通常从下载 Android 源代码到 Android 源代码编译完成, 通常需要一天的时间, 而且还是保证中间不断网的情况下。如果读者的机器是多核 CPU, 可以指定编译时利用的 CPU 核数。例如, 执行下面的代码会利用 4 个 CPU 核。

```
# make -j4
```

但要注意, 如果 CPU 只有 4 个核, 使用上面的命令时可能会造成机器运行缓慢, 无法再处理其他任务的情况。所以读者应根据具体的情况选择参与编译的 CPU 核数。

编译完 Android 源代码后, 会在<Android 源代码根目录>生成一个 `out` 目录, 所以编译生成的目标文件都在该目录的相应子目录中。其中最重要的有 3 个镜像文件 (`ramdisk.img`、`system.img` 和 `userdata.img`), 这 3 个镜像文件都在如下的目录中。在第 2 章会详细介绍如何手动生成和解开这 3 个镜像文件, 以及它们的用途。

```
<Android 源代码本目录>/out/target/product/generic
```

Google 公司已经在官方网站上发布了最新的编译 Android 源代码的方式, 读者可以从如下页面获得更多编译 Android 源代码的细节。

```
https://source.android.com/source/building.html
```

14 官方 Linux 内核源代码

本节将介绍如何为指定的 Android 设备下载 Linux 内核源代码, 并从 Android 设备中获取 Linux 内核的配置文件, 并利用这些配置文件编译 Linux 内核源代码。

1.4.1 Linux 内核支持的 Android 设备

尽管定制 Android ROM 并不需要大量修改 Linux 内核源代码, 不过在很多时候, 还需要依靠 Linux 内核来完成更底层的工作。例如, 获取 ROOT 权限就需要利用 Linux 内核镜像来修改 `default.prop` 文件的内容。所以本节会详细描述如何下载和编译 Linux 内核, 在下一章会介绍如何将生成的 Linux 内核镜像刷到 Android 设备上, 并且对现成的 Linux 内核镜像反编译, 修改其中的内容, 然后再重新生成 Linux 内核镜像, 最后再刷回 Android 设备。

由于 Android 设备的硬件不同, Linux 内核的源代码及其配置文件也有一定的差异, 而且有的差异非常大, 所以下载 Linux 内核源代码一定要清楚是为哪些 Android 设备下载这些源代码。本书主要以 Google Nexus 系列设备为主。因为这些设备的 Linux 内核源代码已经在 Google 公司官方网站发布了, 任何人都可以自由下载、编译和分发。当然, Google 发布的这些 Linux 内核源代码除了支持 Nexus 系列设备外, 还支持一些老的 Android 设备以及 Android 模拟器。目前官方发布的 Linux 内核源代码支持的所有设备如下:

- ❑ Android 模拟器, 也就是金鱼 (goldfish) 内核。
- ❑ ADP1: 也就是 Google 公司发布的第一款 Android 手机 G1, 代号为 Dream。
- ❑ ADP2: 也就是 G2, 代号为 Sapphire。
- ❑ Nexus One。
- ❑ Nexus S。
- ❑ Nexus 4。
- ❑ 熊猫板 (PandaBoard), 一种低功耗的开发板。
- ❑ Galaxy Nexus。
- ❑ Nexus 7。

- ❑ Nexus 10。
- ❑ Xoom: 摩托罗拉推出的第一款 Android 平板电脑。

上述的一些 Android 设备（如 Nexus 4、Nexus 7 和 Nexus 10）都是 Google 公司最近两年新出的 Android 设备，所以 Linux 内核都比较新（都在 Linux 3.1 以上）。本书主要将以这些设备中的 Nexus 7 为主，因为这些设备中只有 Nexus 7 最廉价，而且里面的配置正好是我们需要的。其他的 Nexus 设备的操作方法与 Nexus 7 类似。

1.4.2 下载 Linux 内核源代码

本节将详细介绍如何下载 Google 公司官方发布的 Linux 内核源代码。Google 公司目前提供了 7 套 Linux 内核源代码，那么到底应该下载哪个呢？

下载 Linux 内核源代码要使用 git 命令，下面是下载这 7 套 Linux 内核源代码的命令。

```
# git clone https://android.googlesource.com/kernel/common.git
# git clone https://android.googlesource.com/kernel/exynos.git
# git clone https://android.googlesource.com/kernel/goldfish.git
# git clone https://android.googlesource.com/kernel/msm.git
# git clone https://android.googlesource.com/kernel/omap.git
# git clone https://android.googlesource.com/kernel/samsung.git
# git clone https://android.googlesource.com/kernel/tegra.git
```

我们可以用这 7 套 Linux 内核源代码的 git 文件名来表示它们，也就是 common、exynos、goldfish、msm、omap、samsung 和 tegra。这 7 套 Linux 内核源代码中的后 6 套可以用于上一节给出的一系列的 Android 设备中，具体的对应关系如下。

- ❑ common: 通用的 Linux 内核，后面 6 套 Linux 内核源代码都以该源代码为基础。
- ❑ exynos: 用于使用三星的 Exynos 芯片的 Android 设备，典型的代表是 Nexus 10。
- ❑ goldfish: 用于 Android 模拟器的内核源代码。
- ❑ msm: 用于使用高通 MSM 芯片的 Android 设备，典型的代表是 ADP1 (G1)、ADP2 (G2)、Nexus One 和 Nexus 4。
- ❑ omap: 用于使用德州仪器 (TI) OMAP 芯片的 Android 设备，典型的代表是 PandaBoard 和 Galaxy Nexus。
- ❑ samsung: 用于使用三星 Hummingbird 芯片的 Android 设备，典型的代表是 Nexus S。
- ❑ tegra: 用于使用恩威迪亚 (NVIDIA) Tegra 芯片的 Android 设备，典型的代表是 Xoom 和 Nexus 7。

如果读者恰好使用了上述的设备，就可以下载相应的 Linux 内核源代码了。如果上述设备列表中没有读者使用的设备，就只能到相应厂商的官方网站^①下载 Linux 内核源代码。例如，使用 HTC 手机的读者可以从下面的地址下载用于指定型号设备的 Linux 内核源代码。

<http://www.htcdev.com/devcenter/downloads>

从理论上讲，由于 Linux 内核基于 GPL 协议，所以任何使用 Linux 内核的设备都应该开发 Linux 内核源代码。通常有一定规模的手机厂商会遵守这个协议。不过有很多中小型手机厂商因为要保护知识产权，并未公布自己手机使用的 Linux 内核源代码，所以使用这些 Android 设备的读者将无法通过编译 Linux 内核源代码的方式刷机、获取 ROOT 权限以及完成其他与 Linux 内核相关的工作。

由于本书主要使用 Nexus 7 来讲解与 Linux 内核相关的技术（如获取 ROOT 权限），所以会使用如下的命令下载 Linux 内核。

① 最好到官方网站下载 Linux 内核源代码，因为第三方网站发布的 Linux 内核源代码有可能向其中加入不和谐的东西，有可能会发生意想不到的事情。

尽管前面介绍的几套 Linux 内核源代码都是基于相应芯片的，但这并不意味着使用这些芯片的 Android 设备就可以直接使用这些 Linux 内核源代码。使用这些芯片只是可使用相应 Linux 内核的必要条件，并不是充分条件。还需要对 Linux 内核进一步地调整。但可以在相应 Linux 内核源代码的基础上进行。例如，如果某一部 Android 设备基于 Tegra 芯片，就可以在相应的 Linux 内核源代码的基础上进行调整。

1.4.3 查看 Linux 内核的分支

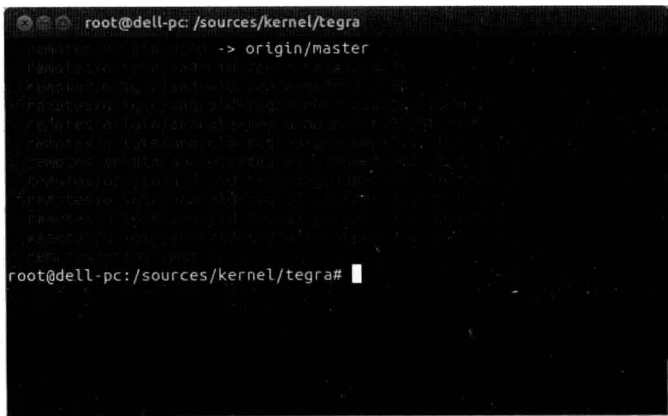
可能对 `git` 命令不熟悉的读者会感到惊讶，为什么下载完 Linux 内核源代码后，在下载目录什么都没有呢，其实所有的 Linux 内核源代码已经下载到本地了，只是都在版本库中。而 `git` 的版本库都在下载目录的“`.git`”子目录中。由于“`.git`”是隐藏目录，所以直接使用 `ls` 命令是看不到“`.git`”目录的，使用 `ls -al` 命令可以查看当前目录中的所有子目录和文件。

每一套 Linux 内核源代码都有多个分支，所谓分支就是在对同一套 Linux 源代码进行微调后产生的不同版本。如果不使用 git 进行管理，就需要备份完整的 Linux 内核源代码，而有了 git，只需要存储修改过的部分即可。

现在进入 Linux 内核源代码的下载目录，然后执行下面的命令查看所有的分支。

```
# git branch -a
```

如果成功执行上面的命令，会输出如图 1-3 所示的内容。



▲图 1-3 当前 Linux 内核源代码的所有分支

可能有很多读者一看到这些分支有些发蒙，从内核版本来看，只有 2.6 和 3.1。从 2.6 前面的 `moto` 字样可以判断，该 Linux 内核用于 Xoom。而后面 6 个分支都可以用于 Nexus 7。不过这 6 个分支的版本都一样，只是分支名结尾处不一样。到底使用哪个 Linux 内核呢？一开始笔者也有些迷惑，通过在网上搜索，发现很多国人和老外也在问同样的问题。最后发现有人回答了句“看看 Google 公司的人在干什么就知道选哪个了”。那么笔者怎么知道 Google 公司内部在干什么？进入 Google 公司总部？哦，应该没这么麻烦。在下一节将告诉读者该如何做。

1.4.4 看看 Google 公司的人在做什么

在 1.4.3 小节留了一个问题，应该使用哪个分支呢？这一点当然 Google 公司会告诉我们。从理论上说，既然 Google 公司公开了 Linux 内核源代码，就应该派人去维护，也就是说应该可以查到维护的日志。

Google 公司已经将 Linux 内核和 Android 源代码的各个分支和维护情况都放到了如下的网站。
<https://android.googlesource.com>

打开该页面, 会看到一个很长的列表。现在需要找到用于 Nexus 7 的 Linux 内核维护日志。在列表右侧的描述 (Description) 中很容易找到“Files specific to Nexus 7”的字样, 左侧对应的是“device/asus/grouper”, 如图 1-4 所示。

Name	Description
Kernel-Projects	
Platform-Projects	Base project for all active
Platform-Unrestricted-Projects	Base project for Android
Public-Projects	
accessories/manifest	
device/asus/grouper	Files specific to Nexus 7
device/asus/tilapia	
device/common	

▲图 1-4 Google Android 源代码维护日志总目录

读者要记住一点, 与 Nexus 7 相关的代码名是 grouper^①, 所以不管是 Linux 内核, 还是 Android 源代码, 或是其他的东西, 凡是带有 grouper 字样的, 都与 Nexus 7 相关。现在单击进入“device/asus/grouper”链接, 会看到有一个长长的日志列表。现在仔细看一下这个日志列表, 找一下规律和出现最多, 并与 Linux 内核分支有关的词汇。从理论上说, 与当前正在使用和维护的 Linux 内核分支的字样会频繁出现在维护日志中, 可能眼尖的读者会发现至少出现了 5 次“jb-mr1.1”^②, 如图 1-5 黑框中所示。而对照上一小节中如图 1-3 所示的分支列表, 正好倒数第二个分支^③包含“jb-mr1.1”。而其他的分支名称不是多一些内容, 就是少一些内容, 所以可以基本断定, Google 公司目前经常维护该分支, 那么很有可能该分支就是 Nexus 7 最新使用的 Linux 内核^④。

Branches <ul style="list-style-type: none"> • master • jb-dev • jb-mr0-release • jb-mr1-dev • jb-mr1-dev-plus-aosp • jb-mr1-release • jb-mr1.1-dev • jb-mr1.1-release • jb-release • tools_r21 	691c86c Merge "Adding TARGET_CPU_VARIANT to config file." by Christopher Ferris - 4 days ago master 6e01ad7 Adding TARGET_CPU_VARIANT to config file. by Christopher Ferris - 5 weeks ago fcc78da More reformatting for uniformity by Jean-Baptiste Queru - 3 weeks ago d962471 Re-format for uniformity with other licenses by Jean-Baptiste Queru - 3 weeks ago 11081d2 Reformat licences to a uniform 77-column format by Jean-Baptiste Queru - 3 weeks ago ad05105 Updated self-extractors for JDP39 by Jean-Baptiste Queru - 4 weeks ago bc28601 Factory images for JDP39 by Jean-Baptiste Queru - 4 weeks ago e8d35a7 Handle ASUS proprietary binaries by Jean-Baptiste Queru - 6 weeks ago 829187c Packages for JDP82 by Jean-Baptiste Queru - 2 months ago 0c975c8 Reconcile with [jb-mr1.1-release] do not merge by The Android Open Source Project - 3 months ago 8f990c4 am f59af94e: am b7eeecdd: grouper: update prebuilt kernel by Ramanan Rajeswaran - 3 months ago 1e49e45 grouper: update prebuilt kernel by Ramanan Rajeswaran - 3 months ago [jb-mr1.1-release] android-4.2.2_r1 f59af94 am b7eeecdd: grouper: update prebuilt kernel by Ramanan Rajeswaran - 3 months ago b7eeecdd: grouper: update prebuilt kernel by Ramanan Rajeswaran - 3 months ago [jb-mr1.1-dev] android-cts-4.2_r2 6a5145c am 2f381992: JOP40D factory images for Grouper by Jean-Baptiste Queru - 3 months ago 2f38199 JOP40D factory images for Grouper by Jean-Baptiste Queru - 3 months ago [jb-mr1-dev-plus-aosp] e8c475a am 8dc0ecbf: Merge "Run fastboot oem unlock before flashing" by Jean-Baptiste Queru - 3 months ago 8dc0ecbf Merge "Run fastboot oem unlock before flashing" by Jean-Baptiste Queru - 3 months ago 2286d70 Reconcile with [jb-mr1.1-release] do not merge by The Android Open Source Project - 3 months ago 0c40ef3 Run fastboot oem unlock before flashing by Jean-Baptiste Queru - 4 months ago
Tags <ul style="list-style-type: none"> • android-cts-4.2_r2 • android-cts-4.1_r2 • android-4.2.2_r1 • android-4.2.1_r1.2 • android-4.2.1_r1.1 • android-sdk-support_r11 • android-cts-4.2_r1 • android-4.2.1_r1 • android-4.2_r1 • android-4.1.1_r6.1 	

▲图 1-5 官方维护 Linux 内核源代码的日志

① 通常为每一个设备单独开发和维护的 Android 源代码和 Linux 内核源代码都会有一个开发代号, 例如, Nexus 7 叫 grouper。在后面的章节会看到更多设备的开发代号。在这里先留个悬念, 看看读者能从书中找到多少种设备的开发代号。

② 可能在读者看到本书时日志内容会改变, 不过查找的方法类似。

③ [remotes/origin/android-tegra3-grouper-3.1-jb-mr1.1](https://android.googlesource.com/remotes/origin/android-tegra3-grouper-3.1-jb-mr1.1)

④ 现在只是基本断定, 只有编译完后刷机成功才能 100%肯定。关于刷 Linux 内核的方法将在下一章详细介绍。

下面开始切换到该分支。

如果是第一次切换到该分支，使用如下的命令。

```
# git checkout -b android-tegra3-grouper-3.1-jb-mr1.1
remotes/origin/android-tegra3-grouper-3.1-jb-mr1.1
```

以后再切换到该分支，可以使用下面的命令。

```
# git checkout android-tegra3-grouper-3.1-jb-mr1.1
```

成功切换分支后，再次执行“git branch -a”命令，会看到输出如图 1-6 所示的信息。其中星号标注的就是当前使用的 Linux 内核源代码分支。



```
root@dell-pc: /sources/kernel/tegra
root@dell-pc:/sources/kernel/tegra# git branch -a
  android-tegra3-grouper-3.1-jb-fr2
  android-tegra3-grouper-3.1-jb-mr0
* android-tegra3-grouper-3.1-jb-mr1.1
  master
  -> origin/master

root@dell-pc:/sources/kernel/tegra#
```

▲图 1-6 当前使用的 Linux 内核源代码分支

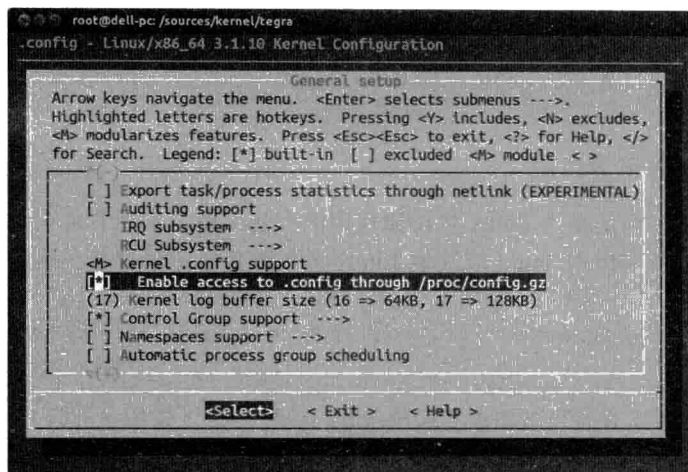
1.4.5 获取 Linux 内核的配置文件

Linux 内核被设计成可插拔式的，也就是说 Linux 内核的所有组件（包括驱动、内存管理、I/O 管理等）都可以很容易移除和添加，而完成组装和下载任务的就是一个叫“.config”的配置文件。该配置文件在 Linux 内核源代码的根目录。尽管 tegra 可以用于 Nexus 7，但直接编译该 tegra 的源代码还是会有一些错误的，原因就是“.config”文件中的某些配置有问题。当然，我们可以通过自己修改“.config”文件来解决问题，但比较麻烦，所以可以找一部已经刷了最新版本（Linux 内核版本需要是 3.1.10）的 Nexus 7，进入 Nexus 7 的 Shell，在“/proc”目录下寻找一个叫 config.gz 的文件。该文件就是已经配置好的“.config”文件的压缩版本，将该文件从 Nexus 7 上下载到本地，并将 config.gz 解压，会得到 config 文件，最后用 config 文件覆盖“.config”文件即可^①。

覆盖“.config”文件后，执行 make 命令编译 Tegra Linux 内核时就不会有任何错误了，编译完后会在<Linux 内核源代码根目录>/arch/arm/boot 目录中生成一个 zImage 文件，该文件就是 Linux 内核源代码编译后生成的二进制版本，但需要将 zImage 文件制作成内核镜像文件（boot.img）才能刷到 Nexus 7 上。这些内容会在下一章详细介绍。

可能的读者并未在/proc 目录中找到 config.gz 文件。实际上，config.gz 文件是否存在与 Linux 内核的配置有关。如果要想在 zImage 文件中包含 config.gz 文件，必须在“.config”文件中将 CONFIG_IKCONFIG_PROC 设为“y”，否则 zImage 是不会在/proc 目录生成 config.gz 文件的。当然，也可以执行 make menuconfig 命令查看 Linux 内核的配置菜单，进入“General setup”菜单项，找到“Enable access to .config through /proc/config.gz”菜单项，按空格键，将该菜单项设为“[*]”，最后退出界面并保存配置即可，配置界面如图 1-7 所示。

① 如果要保留 Linux 内核源代码旧的配置，可以在覆盖“.config”文件之前先备份该文件。



▲图 1-7 允许生成/proc/config.gz 文件

除了可以通过直接覆盖“.config”文件和使用 make menuconfig 重新保存配置文件的方式外，还可以将 config 文件放到如下的目录，并将该文件改名为 new_defconfig。然后执行 make new_defconfig 命令用 new_defconfig 文件覆盖“.config”文件的内容，然后再执行 make 命令编译 Linux 内核源代码即可。

<Linux 内核源代码根目录>/arch/arm/configs



注意

文件名 new_defconfig 分为两部分，前一部分是“new”，这部分名称可以随便起，而后一部分“_defconfig”是固定的，也就是说所有用于 Linux 内核配置的文件名都必须以“_defconfig”结尾。

本节涉及的 config.gz 和 config 文件已经包含在随书光盘中，读者可以直接使用这两个文件，而且带的 Tegra Linux 内核源代码 (tegra.gz) 也已经使用 config 文件替换了“.config”文件，所以可以直接编译 Tegra Linux 内核的源代码。

1.4.6 安装交叉编译器

由于我们需要将编译的 Linux 内核放到 Nexus 7 或其他 Android 设备上运行，并且这些设备都是 ARM 架构的 CPU，所以就不能使用 Linux 下的 gcc 对 Linux 内核源代码进行编译。因此要使用一种称为“交叉编译器”的工具。这里所说的交叉编译器就是可以在 X86 架构的 PC 上编译生成可以在 ARM 架构的 Android 设备^①上运行的二进制文件。这里的二进制文件主要指 Linux 内核的 zImage 文件。

其实 Android 源代码本身已经带了一套交叉编译器（在 prebuilt 目录中），虽然 Android 带的交叉编译器可以完美地编译 Linux 内核，但由于 Linux 内核源代码对编译环境的要求非常苛刻，编译器的版本、配置文件 (.config) 等条件只要有一点差异，都可能无法成功在 Nexus 7 上运行，所以需要了解当前 Linux 内核版本需要使用哪个版本的交叉编译器进行编译。

Google 公司官方编译用于 Nexus 7 的最新 Linux 内核源代码使用的交叉编译器版本是 4.4.3，所以需要使

用下面的命令下载该版本的交叉编译器。

```
# git clone https://android.googlesource.com/platform/prebuilt
```

① ARM 架构的不光是 Android 设备，其他设备，例如，微软的 Surface，也是 ARM 架构的。只是本身的主题是 Android，所以这里特指 Android 设备。

为了方便，读者可以将交叉编译器的 bin 子目录加到 PATH 环境变量中，也可以在编译 Linux 内核源代码的终端中执行下面的命令，临时将 bin 目录加到 PATH 环境变量中。这里假设交叉编译器正好在当前目录的 prebuilt 子目录中。

```
# export PATH=$(pwd)/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH
```

1.4.7 编译 Linux 内核源代码

前面几个小节做的都是编译 Linux 内核源代码的准备工作。本小节将正式编译 Linux 内核源代码。首先在 Linux 终端进入 Tegra 或其他 Linux 内核源代码的根目录，然后执行下面的命令设置必要的环境变量。

```
# export PATH=$(pwd)/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH
# export ARCH=arm
# export CROSS_COMPILE=arm-eabi-
```

其中第 1 行命令用于设置交叉编译器编译命令的路径；第 2 条命令告诉 Linux 内核要为 ARM 架构的设备生成 Linux 内核（zImage 文件）；第 3 条命令设置交叉编译器的前缀，在编译 Linux 内核源代码的过程中可能会调用一系列编译命令，这些编译命令文件名的前缀都是“arm-eabi-”。读者可以进入交叉编译器的\$(pwd)/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin 看一下，该目录中的文件名几乎都是以“arm-eabi-”作为前缀的。

最后执行 make 命令即可完美编译 Linux 内核源代码。完整编译的时间大概 20 分钟左右。编译完后，就会在<Linux 内核源代码根目录>/arch/arm/boot 目录中生成一个 zImage 文件。这就是在 Nexus 7 上运行的 Linux 内核的二进制版本。

1.5 Cyanogenmod(CM)源代码

尽管使用官方的 Android 源代码和 Linux 内核源代码可以紧跟 Google 的步伐，第一时间体验 Android 的最新版本，不过官方的这些源代码只适合 Nexus 系列的设备。而其他很多流行的 Android 设备都无法直接使用官方的源代码制作 ROM。由于很多读者使用的都是非 Nexus 系列的 Android 设备，所以本书为了满足这些读者的需求，大多数内容将以另外一个经过第三方组织 CM 修改的 Android 源代码和 Linux 内核源代码为主，尽管 CM 发布修改版源代码的时间可能比官方发布相应源代码的时间晚一些（一般也晚不了多长时间），但使用 CM 源代码制作的 ROM 可以运行在更多的 Android 设备上，例如，HTC、三星等国内主流的机型都在 CM 的支持范围内。如果读者对 CM 还不太了解，强烈建议仔细阅读本节的内容。因为通过本节的学习，就足以使读者通过 CM 提供的源代码制作一个可以完美刷机的 ROM。当然，在后面的章节还会更深入地研究 ROM 的制作过程。

1.5.1 什么是 CM

经常刷机的读者肯定对 CM ROM 不陌生。CM 是 Cyanogenmod 的缩写。CM 是目前最受关注的 Android ROM 适配团队^①。目前世界上绝大多数第三方 Android ROM 团队出品的 ROM 都是以 CM ROM 为基础制作的。例如，小米 ROM、还有老罗的锤子 ROM 都是在 CM ROM 的基础上修改而来的。

大家都知道，Android 系统本质上由 Android 和 Linux 内核两部分组成。而决定 Android 系统

^① 甚至有很多人认为，如果 CM 不支持某一款 Android 设备，有可能会影响该款 Android 设备的销量。不过大家也别当真，这也有些夸大的成分。但至少可以看出，CM 团队在 Android 领域的影响不可小觑！

是否能在 Android 设备上成功运行的主要因素就是 Linux 内核中的驱动^①。而 Google 公司官方提供的 Linux 内核源代码只支持数量有限的几种 Nexus 系统的 Android 设备，所以 CM 团队的主要工作就是在 Google 公司官方提供的 Linux 内核源代码^②的基础上，添加适合各种 Android 设备的驱动，而 CM 对 Android 源代码的改动不大，UI 仍然保持着原生 Android 的风格。只是修改了其中一些系统应用程序，并添加了一些自己的应用程序，例如，手电筒、从 CM 升级 ROM、Root 等功能。所以 CM 团队的主要成果就是为不同的 Android 设备适配 Linux 内核中的驱动。

CM 的这些工作也为其他的 Android ROM 团队提供了方便之门。因为有了 CM 适配的 Linux 内核，定制新的 ROM，主要的工作就是修改 CM 提供的 Android 源代码。尽管每一款 Android 设备都有一套自己独立的 Linux 内核源代码，但 Android 源代码的大多数部分却是共享的，也就是说，对于某个版本的 CM Android 源代码（如 CM 10.1），只需要下载一套，并且和相应 Android 设备的 Linux 内核源代码搭配即可制作成在不同 Android 设备上运行的 ROM。而我们要做的只是维护一套 Android 源代码，多套 Linux 内核源代码（这些源代码大多数由 CM 团队负责维护）。这样将大大减轻制作 Android ROM 的复杂度和适配各种 Android 设备的烦恼。

CM 的官方地址是 <http://www.cyanogenmod.org>。读者可以从 CM 官网的相应页面下载适合不同 Android 设备的 Android 源代码和 Linux 内核源代码。Android 源代码和 Linux 内核源代码都在 github (<http://www.github.com>) 上托管，本章后面的部分会详细介绍如何下载、编译 CM 源代码，并且从 CM 源代码制作我们的第一个完美的 ROM。如果读者要想在真机上刷这个 ROM，只需要有一个 CM 支持的 Android 设备即可，要想了解 CM 支持哪些 Android 设备，请看下一小节的内容。

1.5.2 CM 支持哪些 Android 设备

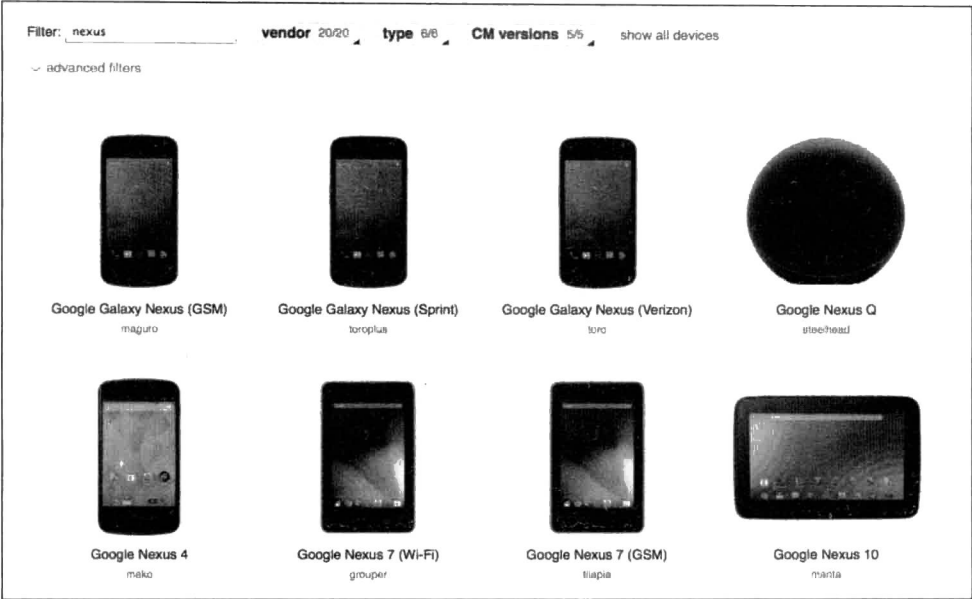
在下载和使用 CM 源代码之前了解 CM 支持哪些设备非常必要。因为如果 CM 如果不支持您的 Android 设备，下载的 CM 源代码生成的 ROM 即使可以成功刷机，也很可能启动不了或遇到其他莫名其妙的现象。不过读者也不用担心，CM 支持的 Android 设备种类是非常多的。国外很多常见的品牌都在支持之列，例如，三星、摩托罗拉、索爱、HTC 的大多数型号都支持。甚至华为等国内厂商的部分型号的 Android 设备也在支持之列。当然，Nexus 系列就更不在话下了。

要想知道 CM 是否支持自己的 Android 设备，可以进入 CM 的主页面，然后单击右上角的“Devices”链接（如果因网页改版没有该链接，可以直接访问 <http://wiki.cyanogenmod.org/w/Devices>），进入设备列表页面。在页面的中上部找到如图 1-8 左上角所示的“Filter”过滤框。先单击右侧的“show all devices”链接，然后在“Filter”过滤框输入要查找的关键词，例如要查找支持哪些 Nexus 设备，可以输入“nexus”，这时在下方就会显示如图 1-8 所示的相关 Android 设备列表。

如果在图 1-8 所示的设备列表中找到自己的 Android 设备，可以直接单击该设备，进入下一个页面。在该页面中会告诉你如何下载 Android 源代码和 Linux 内核源代码，以及其他相关的资源。

① Android 设备和 PC 一样，不同厂商的 Android 设备使用的硬件不同，所以要求不同的 Linux 驱动。

② 有的 CM 修改版的 Linux 内核源代码可能不是来自 Google 公司官方，而是其他厂商发布的 Linux 内核源代码，不过这已经不重要了，我们直接享用 CM 的成果就好了！



▲图 1-8 查找 CM 支持的 Android 设备

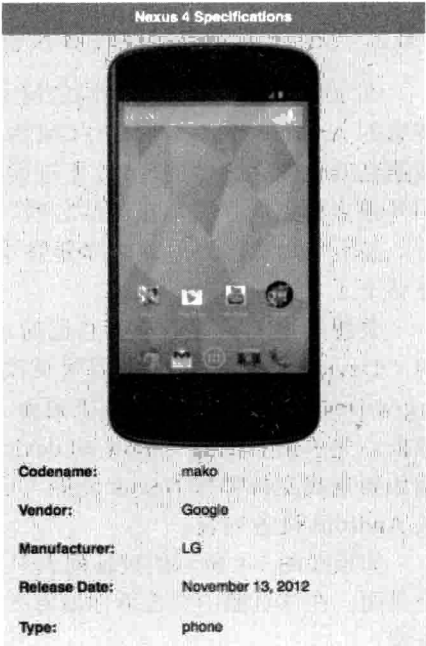
1.5.3 与 Android 设备对应的 Codename 和 CM 版本

在下载和编译 CM Android 源代码之前需要知道如下两个信息。

- ❑ 当前 Android 设备对应的 Android 源代码的代码名称（Codename），也就是开发代号。
- ❑ 支持当前 Android 设备的 CM 源代码版本。

代码名称在编译 Android 源代码（选择 target）和下载对应的 Linux 内核源代码的过程中非常重要。例如，Nexus 7 Wi-Fi 版对应的代码名称是 grouper，Nexus 4 对应的代码名称是 mako。也就是说，如果读者看到 grouper 的字样，就说明对应的 Android 源代码和 Linux 内核源代码是为 Nexus 7 Wi-Fi 版准备的，而看到 mako 的字样，就是为 Nexus 4 准备的。那么对于不熟悉的设备，怎么知道与其对应的代码名称呢？

如果读者按着上一小节的方法找到自己的 Android 设备，并进入与该设备相关页面。可以在页面的右侧看到一个设备规范列表，例如，进入的是 Nexus 4 设备的页面，会看到“Nexus 4 Specifications”列表，在设备图像下面有若干与该设备相关的参数，如图 1-9 所示。通常第一个参数就是代码名称（Codename），最后一个或倒数第二个参数（CM Support）就是 CM 的版本（由于参数列表太长，图 1-9 未显示全，具体的参数值请查看实际的页面）。例如，适合 Nexus 4 的只有 CM 10.1，对应 Android 4.2 和 Linux 3.1.10^①。



▲图 1-9 Nexus 4 的规范列表

① 同一个 CM 版本，对应的 Linux 内核版本不一定相同。例如，对于 CM 10.1 来说，Nexus 7 Wi-Fi 版的 Linux 内核版本是 3.1.10，而 Nexus S 的 Linux 内核版本是 3.0.50。

1.5.4 下载 CM Android 源代码

下载 CM Android 源代码与下载官方 Android 源代码的方法及其类似，同样也是用 `repo` 脚本（要保证当前 Ubuntu Linux 已经安装了 `git`）下载，只是下载的地址不同。下面看一下详细的下载步骤。

第 1 步：下载 repo 脚本文件

```
# mkdir -p ~/bin
# curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
```

第 2 步：设置 repo 脚本文件的路径

```
export PATH=${PATH}:~/bin
```

为了方便，可以将上述代码添加到 `~/.profile` 文件中，这样就不需要每次使用新的 Linux 终端，或重启机器后都要重新设置 `repo` 脚本文件的路径了。

第 3 步：初始化下载地址和版本

```
# mkdir -p /sources/cyanogenmod
# cd /sources/cyanogenmod
# repo init -u git://github.com/CyanogenMod/android.git -b cm-10.1
```

第 4 步：开始下载

```
# repo sync
```

从这 4 个下载步骤可以看出，除了第 3 步与下载官方 Android 源代码不同外，其他的步骤完全一样（只是将 CM 源代码下载到了“`/sources/cyanogenmod`”目录）。

在第 3 步的一个关键是了解需要下载哪一个版本的 CM Android 源代码。为了了解这些信息，首先需要确定读者手中 Android 设备的型号。例如，如果是 Nexus S，需要按着上一节的方法进入 Nexus S 设备页面，然后在右下方找到“CM Support”参数项。从该参数的值可以看出，支持 Nexus S 的 CM 版本是 7、9、10、10.1，共 4 个版本。当然，下载哪个版本都可以，不过要想使用最新的 Android 版本，就需要下载 10.1，该版本对应的 Android 版本是 4.2。如果读者手中的 Android 设备是 Nexus 4，那么只有 CM 10.1 支持该设备，所以只能下载 CM 10.1。一旦确定了要下载哪个 CM 版本，就需要在执行 `repo init` 命令之前指定要下载的具体分支，例如，本例要下载 CM 10.1，所以需要使用下面的命令执行分支。

```
# repo init -u git://github.com/CyanogenMod/android.git -b cm-10.1
```

不过要注意的是，在 CM 10.1 之前的版本，指定分支通常使用有意义的名称，例如，比较古老的 HTC Hero (G3) 只有 CM7 支持，所以需要使用下面的命令初始化要下载的分支。

```
# repo init -u git://github.com/CyanogenMod/android.git -b gingerbread
```

要想知道与某一 Android 设备对应的 CM 版本分支，首先按着 1.5.2 小节的方法找到该设备，并进入该设备的页面。如果在右侧的 Specifications 部分的最下方找到“Latest CM version”字段，则“-b”后面直接跟该字段值即可。如果没有“Latest CM version”字段，那就是 CM 10.1 或更新的版本，所以“-b”后面直接跟 `cm-10.1` 即可。

如果读者要想了解更详细的下载过程，可以查看下面的页面（这里 Codename 需要替换成相应的字段值）。

```
http://wiki.cyanogenmod.org/w/Build_for_Codename
```

例如，要想进入 HTC Hero 的下载描述页面，首先要知道 HTC Hero 的 Codename 是 `hero`，然后进入如下的页面。

```
http://wiki.cyanogenmod.org/w/Build_for_hero
```


❗注意

下载 CM Android 尽管需要考虑具体的 Android 设备，但这只是为了确定要下载哪个 CM 版本，而每一个 CM 版本可以用在一系列的 Android 设备中。所以 CM Android 源代码与 CM Linux 内核源代码的主要版本差异就是前者共享于多款 Android 设备，而后者对于每一款 Android 设备都是独立的。例如，Nexus 4 和 Nexus 7 Wi-Fi 都可以使用 CM 10.1 的 Android 源代码，但这两款设备的 Linux 内核源代码是各自独立的，不可以互换。所以为多款 Android 设备制作 ROM，Android 源代码只需要一套，而每一款 Android 设备都需要有独立的一套 Linux 内核源代码。

1.5.5 下载经过 CM 适配的 Linux 内核源代码

CM Linux 内核源代码在 github 上托管，在 CM 官网上可以找到相应的下载链接。首先进入 CM 官网与某一 Android 设备相关的页面，例如，Nexus S 的页面地址如下：

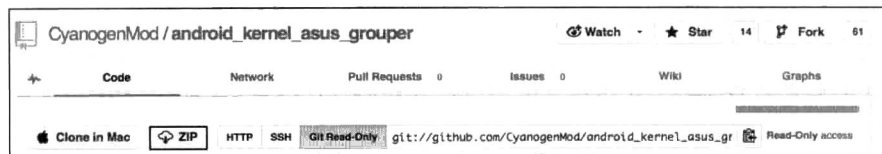
http://wiki.cyanogenmod.org/w/Crespo_Info

在页面的中部可以找到如图 1-10 所示的两个链接，其中 Kernel 后面的链接就是 github 上的 Linux 内核源代码下载地址。

Source Repositories:	
Device	http://www.github.com/cyanogenmod/android_device_samsung_crespo
Kernel	http://www.github.com/cyanogenmod/android_kernel_samsung_crespo

▲图 1-10 CM Linux 内核源代码下载链接

单击该链接进入 github 下载页面，可以看到在下载页面上方会有一个如图 1-11 所示的 git 下载地址。如果读者不想通过 git 下载，也可以单击左侧的“ZIP”按钮下载 zip 压缩版本。



▲图 1-11 CM Linux 内核源代码在 github 上的下载页面

CM Linux 内核源代码与官方提供的 Linux 内核源代码在编译方法上完全一样，读者可以参阅 1.4.7 小节的 Linux 内核源代码的编译方法。

1.5.6 编译 Android 源代码生成 Recovery ROM

尽管 Android 源代码和 Linux 内核源代码都可以单独下载和编译，不过要想利用它们单独编译生成的二进制文件^①制作 ROM 还是比较麻烦的。CM 团队为了更方便地制作 ROM，允许将 Android 源代码与 Linux 内核源代码放在一起，并通过统一的命令进行编译，最后生成的目标文件就是一个完整的刷机包（zip 文件）。这个刷机包是 Recovery ROM^②，所以可以自己进入 Recovery 模式进行刷机。

现在开始利用 1.5.4 小节下载的 CM Android 源代码进一步准备制作 ROM 的环境。读者可以按照如下几步进行操作。

① 编译 Android 源代码生成的二进制文件主要是一些镜像文件，例如，system.img、userdata.img 等。编译 Linux 内核源代码生成的二进制文件主要是 zImage 文件。如果要通过手工方式制作完整的 Android ROM，需要对这些文件做进一步的处理，还需要编写一些脚本。

② Android 设备的刷机包(ROM)有两种：Bootloader 和 Recovery。其中 Bootloader ROM 必须要有一台 PC 或笔记本电脑与 Android 设备通过 USB 线相连才能刷机，而 Recovery ROM 既可以通过 USB 线进行刷机，也可以将刷机包（通常为 zip 压缩文件）复制到 Android 设备的 SD 卡中，完全脱离计算机进行刷机。在后面的章节会详细讨论这两种 ROM 的制作和刷机方法。

第1步：下载一些必要的文件

编译 Android 源代码之前需要下载一些必要的 Library 和 apk 文件。这些文件也不需要自己一个个下载，只需要按照如下方式执行 get-prebuilts 脚本文件即可。

```
# cd /sources/cyanogenmod/vendor/cm
# ./get-prebuilts
```

如果成功执行上面的命令，会看到在 Linux 终端输出如图 1-12 所示的信息，这些信息表明了 get-prebuilts 脚本下载了哪些文件。

```
root@dell-pc: /sources/cyanogenmod/vendor/cm
root@dell-pc:/sources/cyanogenmod/vendor/cm# ./get-prebuilts
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             k   B           k   B   mm:ss  mm:ss   mm:ss Left   Speed
100 178 100 178 0 0 242 0 --:--:-- --:--:-- --:--:-- 511
100 391k 100 391k 0 0 74599 0 0:00:05 0:00:05 --:--:-- 93767
Archive: ./proprietary/Term.apk
  inflating: ./proprietary/lib/armeabi/libjackpal-androidterm4.so
  inflating: ./proprietary/lib/mips/libjackpal-androidterm4.so
  inflating: ./proprietary/lib/x86/libjackpal-androidterm4.so
root@dell-pc:/sources/cyanogenmod/vendor/cm#
```

▲图 1-12 执行 get-prebuilts 脚本文件输出的信息

第2步：下载与 Android 设备相关的源代码

这里的源代码主要指的是 Linux 内核源代码，只不过这一步会将 Linux 内核源代码与 Android 源代码放到一起，编译时统一处理。在这一步需要使用 Android 设备的 Codename，如果读者使用的是 Nexus S，需要执行下面的命令初始化安装环境，如果是第一次执行这些命令，会下载相应的 Linux 内核源代码。要注意，当前的 Linux 终端应该在 Android 源代码的根目录。

```
# source build/envsetup.sh
# breakfast crespo
```

其中 breakfast 命令是在 envsetup.sh 脚本文件中定义的函数，用于根据 Codename 检测相应的 Linux 内核源代码是否存在，如果不存在，就直接从相应的地址下载。如果读者的机器上以前就有 repo 脚本文件，建议使用 repo selfupdate 升级到最新版本。正在下载 Linux 内核源代码的效果如图 1-13 所示。这些输出信息说明正在下载 Nexus 7 Wi-Fi 的 Linux 内核源代码 (Codename = grouper)。下载完后，会在 Android 源代码根目录生成一个 kernel 目录，并且在 kernel 目录中还会根据不同的厂商建立相应的子目录。例如，Nexus S 是三星代工的，所以 Nexus S 对应的 Linux 内核源代码目录是 <Android 源代码根目录>/kernel/samsung/crespo，而 Nexus 7 Wi-Fi 是由华硕代工的，所以对应的 Linux 内核源代码目录是 <Android 源代码根目录>/kernel/asus/grouper。

```
root@dell-pc: /sources/cyanogenmod
root@dell-pc:/sources/cyanogenmod# repo selfupdate
repo version v1.12.2 is current
root@dell-pc:/sources/cyanogenmod# breakfast grouper
including vendor/cm/vendorsetup.sh
ls: 无法访问 device/*/*grouper/cm.mk: 没有那个文件或目录
build/core/product_config.mk:234: *** Cannot locate config makefile for product
"cm_grouper". 停止。
Device grouper not found. Attempting to retrieve device repository from Cyanogen
Mod Github (http://github.com/CyanogenMod).
Found repository: android_device_asus_grouper
Default revision: cm-10.1
Checking branch info
Adding dependency: CyanogenMod/android_device_asus_grouper -> device/asus/groupe
r
Using default branch for android_device_asus_grouper
Syncing repository to retrieve project.
remote: Counting objects: 1644, done.
remote: Compressing objects: 100% (845/845), done.
Receiving objects: 92% (1527/1644), 6.40 MiB | 174 KiB/s
```

▲图 1-13 下载 Linux 内核源代码

第 3 步: 下载与设备有关的私有 (proprietary) 文件

尽管这一步与第 1 步一样, 也需要下载文件。不过这里的下载可不是从 Internet 上下载, 而是从 Android 设备上下载到 PC 上。也就是说要完成这一步的操作就需要有一部 Android 设备。而且还必须为哪款 Android 设备制作 ROM, 就需要哪一款 Android 设备。例如, 要为 Nexus S 制作 ROM, 就需要一部 Nexus S 手机, 并且要通过 USB 线与 PC 相连。

注意

这一步需要 adb 命令与 Android 设备交互, 而这时 Android 源代码还没有编译, 在 out 目录的相应子目录还没有生成 adb 命令, 所以进行这一步之前还需要安装 Android SDK, 并且需要将<Android SDK 根目录>/platform-tools 目录添加到 PATH 环境变量中。

在 PC 与 Android 设备通过 USB 线连接好后, 执行如下的命令下载相应的私有文件。这里下载的是 Nexus S 的私有文件。如果下载其他 Android 设备的私有文件, 需要进入相应的目录, 同样是执行 extract-files.sh 脚本文件。device 目录中与各个设备相关的子目录的命名规则与 Linux 内核源代码目录的命名规则相同。

```
# cd /sources/cyanogenmod/device/samsung/crespo
# ./extract-files.sh
```

第 4 步: 打开缓存 (Cache), 加速编译

尽管这一步不是必须的, 但可能会获得更快的编译速度, 执行下面的命令可以打开用于编译的缓存。或者将该命令加入/root/.profile 文件中。

```
# export USE_CCACHE=1
```

第 5 步: 编译

在这一步将编译 Android 源代码和 Linux 内核源代码, 并最终生成一个可刷机的 ROM (zip 压缩文件)。编译源代码也需要指定具体设备的 Codename, 因为除了编译 Android 源代码, 还要编译 Linux 内核源代码。执行下面的命令可进行编译, 并为 Nexus S 生成 ROM 压缩包。

```
# croot
# brunch crespo
```

其中 croot 是 envsetup.sh 脚本文件中定义的函数, 用于回到 Android 源代码的根目录。brunch 是编译命令, 后面要跟 Codename。例如, 要为 Nexus 7 Wi-Fi 编译并生成 ROM 压缩包, 需要使用 brunch grouper。

编译的过程非常缓慢, 请耐心等待。编译完后, 对于 Nexus S, 会在<Android 源代码根目录>/out/target/product/crespo 目录中生成一个 cm-10.1-20130418-UNOFFICIAL-crespo.zip 文件 (或叫类似的名, 因为文件名包含编译的日期)。如果是其他的 Android 设备, 会在<Android 源代码根目录>/out/target/product 目录以相应的 Codename 命名生成目录, 例如, Nexus 7 Wi-Fi 的刷机包在 grouper 目录中。接下来就是刷机了, 在 1.5.8 小节将详细介绍如何将这个生成的 ROM 刷到相应的 Android 设备上。

答疑解惑: 为什么 APK 程序不带 odex 文件

也许很多读者会提出这样的问题, 其他很多 Android 设备的/system/app 目录中的系统应用都是由“APK 文件”+“同名的 odex 文件”组成的, 为什么自己用 CM Android 源代码编译生成的 APK 程序 (在<Android 源代码根目录>/packages/apps 目录中) 只有 APK 文件呢? odex 文件到哪去了?

实际上, CM Android 源代码默认生成的 APK 程序是 userdebug 模式的, 也就是说并没有生成经过优化的 odex 文件 (是对 dex 文件优化后生成的文件)。不过这对于开发状态并没有任何影响。但在开发完成后, 需要编译生成最终的 ROM。在这种情况下, 最好使用 user 模式进行编译。由

于 lunch 命令显示的编译模式列表（也可称为 Target 列表）并没有 user，所以可以修改<Android 源代码根目录>/build/core/main.mk 文件的内容来达到这个目的。现在打开该文件，找到如下的内容。

```
ifneq (true,$(DISABLE_DEXPREOPT))
  ifeq ($(user_variant),user)
    ifeq ($(HOST_OS),linux)
      WITH_DEXPREOPT := true
    endif
  endif
endif
```

其实上面代码的核心就是 WITH_DEXPREOPT := true。如果将 WITH_DEXPREOPT 属性设为 true，则使用 brunch 命令编译 Android 源代码，就会提取 APK 程序中的 classes.dex，并将其优化成 odex 文件。现在将上面代码中前两个条件语句注释掉，如下面代码所示。

```
#ifneq (true,$(DISABLE_DEXPREOPT))
  #ifeq ($(user_variant),user)
    #ifeq ($(HOST_OS),linux)
      WITH_DEXPREOPT := true
    endif
  #endif
#endif
```

现在重新执行 brunch crespo 命令，就会为 Nexus S 重新生成经过优化的 ROM。查看其中的 APK 程序，已经全部由 APK 和 odex 两个文件组成了。

1.5.7 单独编译 CM Linux 内核源代码

尽管 CM 提供的 Linux 内核源代码与 Android 源代码放在了一起，但实际上，在编译 Android 源代码时只是生成了一个内核文件（kernel），并没有完整地编译 Linux 内核源代码。如果读者要修改 Linux 内核源代码，例如，添加或修改某个驱动程序，还需要重新编译 Linux 内核源代码，以便生成 zImage 文件。

CM Linux 内核源代码的编译方法与官方 Linux 内核源代码的编译方法类似。首先需要从特定的 Android 设备的 /proc 目录获取 config.gz 文件，并将其解压，然后覆盖 Linux 内核源代码根目录的 config 文件。最后再执行 make 命令编译 Linux 内核源代码。详细的操作方法见 1.4.5 小节的内容。

经测试发现，CM Linux 内核源代码在编译时可能会有一些小错误。例如，在编译 Nexus S 的 Linux 内核源代码时会出现如下的错误。其大意是 fbmem.c 文件中包含了 s3cfb.h 头文件，但却没找到该文件。

```
drivers/video/fbmem.c:39:27: fatal error: samsung/s3cfb.h: No such file or directory.
...

```

为了排除这个编译错误，先来看一下 fbmem.c 文件的源代码，找到第 39 行，该行前后与错误相关的代码如下：

```
#ifdef CONFIG_FB_S3C
#include <samsung/s3cfb.h>
#endif
```

很明显，samsung/s3cfb.h 文件在搜索路径中不存在。这个 samsung 目录实际上是相对于<Linux 内核源代码>/include 目录的，查看 include 目录，并没有找到 samsung 目录，这就更谈不上 s3cfb.h 文件了。其中 samsung 目录被放到了<Linux 内核源代码>/drivers/video 目录中，只要将该目录中的 samsung 目录复制到<Linux 内核源代码>/include 目录即可成功编译 Linux 内核源代码。编译生成的 zImage 文件（该文件与编译 Android 源代码生成的 kernel 文件是一样的）可以按照第 4 章的方法生成 boot.img 镜像文件，然后在 Bootloader 模式下使用 fastboot 命令重新刷内核镜像（boot.img）。

1.5.8 刷机！刷机！

现在即将迎来激动人心的时刻：刷机。这回刷机可和传统意义上的刷机不同。一般的刷机都是从网上下载现成的 ROM，然后按说明一步步刷机。而这回可是直接从 Android 源代码和 Linux 内核源代码编译而成的，尽管我们没有修改这些代码，但对于第一次用源代码制作 ROM 的读者来说仍然足够兴奋，因为这将是成为一个 Geek 的第一步。

可能很多从未亲手刷过机的读者会认为刷机存在一些危险，可能会将自己心爱的手机刷成“砖”。没错，刷机的确有风险，不过风险远没有炒股大，但回报却可能比炒股大。学会自己从最原始状态（源代码）制作 ROM 将会对自己创业或找高薪的工作有非常大的帮助。而刷机带来的风险只要注意一下就基本上可以忽略不计。在后面的章节会详细介绍 ROM 的各种组成部分，不过这里先提一下。一个完整的 ROM 主要由 system、recovery、userdata、boot 和 bootloader 五部分组成。其中 system 和 userdata 属于应用层部分。recovery、boot 和 bootloader 属于系统层部分。这五部分除了 bootloader 外，其他的可以任意修改。不会有任何危险，但要注意，在刷机之前，最好有一个保证好用的 ROM 作为备份，万一刷坏了，可以很容易恢复，否则手机就用不了了（硬件没坏，只是软件的问题）。尽管 bootloader 的源代码也随 Android 源代码一起发布，但尽量不要自己修改 bootloader。Bootloader 就像 PC 的 BIOS，一旦坏了，手机就真的成砖了。因为即使要恢复 bootloader，也需要进入 bootloader 模式，一旦 bootloader 模式进不去，就不能通过常规的方法刷机了（需要利用特殊的设备，但大多数人没有这种设备）。所以读者记住一点，只要 bootloader 不动（官方提供的 bootloader 除外），其他的随便刷，毫无危险，即使最终自己不知道怎么恢复。淘宝上有一些商家提供了恢复的服务，几十元就可恢复如初，所以头一次刷机的读者并不需要有任何担心。

在刷机之前需要保证 bootloader 是解锁的。如果未解锁，需要在 Android 设备正常开机的情况下执行 `adb reboot bootloader` 命令进入 bootloader 模式，然后执行 `fastboot oem unlock` 命令解锁。如果成功解锁，会在 bootloader 模式界面的最后显示红色的“UNLOCKED”。

由于上一节生成的是 Recovery ROM，所以需要进入 Recovery 模式刷机。读者可以在 bootloader 模式下直接进入 recovery 模式，或正常启动 Android 设备，然后执行 `adb reboot recovery` 命令进入 Recovery 模式。建议读者使用第三方的 ClockworkMod Recovery 刷机（简称 CWM Recovery）。虽然可以直接到 <http://www.clockworkmod.com> 下载最新版的 Recovery。但并不需要这么麻烦，CM Android 源代码带的 Recovery 就是 ClockworkMod Recovery，而且在刷机包所在的目录生成了一个 `recovery.img` 镜像文件。只要按着前面的方法进入 bootloader 模式，然后在 Linux 终端进入 `recovery.img` 文件所在的目录，并执行如下的命令即可刷这个 Recovery。

```
# fastboot flash recovery recovery.img
```



注意

刷 Recovery 时要注意，Recovery 的镜像与 Linux 内核类似，也是与具体的 Android 设备有关的。不要将其他 Android 设备的 Recovery 镜像刷到自己的 Android 设备上，否则可能进不了 Recovery 模式。当然，刷错了也没关系。进入 bootloader 模式重新刷正确的 Recovery 镜像即可。

刷完 Recovery 镜像后，正常启动 Android 设备，然后将 ROM 文件复制到 SD 卡的根目录（也可以是 SD 卡的其他目录），并且备份手机中的重要数据。现在执行 `adb reboot recovery` 命令进入 Recovery 模式，先选择“wipe data/factory reset”和“wipe cache partition”，清空手机上的数据，然后选择“install zip from sdcard”，在 SD 卡上找到 ROM 压缩文件，然后开始刷机。刷完后，选择“reboot system now”重启手机，现在就可以体验新成果了！

关于此电子书的说明

本人由于一些便利条件，可以为您提供各种中文图书的PDF电子版，保证质量清晰。只要图书不是太新，文学、法律、计算机、经济、医学、工业、学术等方面的图书，都可以帮您制作，如果您有这方面的需求，可以通过QQ联系我，我的QQ号是 [3330972307](#)。