Android Notes

www.theqiong.com

2014 年 10 月 26 日

Contents

1 Introduction **11**

1.1 Overview 11

1.2 Histroy 12

1.3 Releases 12

1.4 License 13

2 Installation **15**

3 Update **17**

4 Activity **19**

4.1 Introduction 19

5 UI **21**

6 Fragment **23**

7 Broadcast Receiver **25**

8 Storage **27**

8.1 File Storage 27

8.2 Shared Preferences 27

8.3 Database Storage 27

9 Content Provider **29**

10 Multimedia **31**

10.1 Streaming Media 31

10.2 Camera . . . 31
10.3 Audio . . . 31
10.4 Video . . . 31
10.5 Debug . . . 31

11 Service **33**
11.1 Multithreading . . . 33
11.2 Multitasking . . . 33

12 Network **35**
12.1 Web Browser . . . 35
12.2 Wireless . . . 35
12.3 Screen Shot . . . 35

13 Location **37**

14 Sensor **39**

15 Architecture **41**
15.1 Linux Kernel . . . 41
15.2 Library . . . 42
15.2.1 JVM . . . 43
15.2.2 HAL . . . 43
15.3 Application . . . 44
15.4 Hardware . . . 45
15.4.1 Bluetooth . . . 46
15.5 Multi-Point Touch . . . 46
15.6 Voice . . . 46

16 Development **47**
16.1 Init Tool . . . 47
16.2 Build Tool . . . 47
16.2.1 Apache Ant . . . 47
16.2.2 Apache Maven . . . 48
16.2.3 Gradle . . . 49
16.3 Debug . . . 50

16.3.1 Lint 50

17 Security **51**

17.1 Android Virus 51

17.2 Root 51

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Android 是一个以 Linux 为基础的开放源代码移动设备操作系统，主要用于智能手机和平板电脑，目前由 Google 成立的 Open Handset Alliance（OHA，开放手持设备联盟）持续领导与开发中。

Android 系统最初由 Andy Rubin 等人进行开发，最初开发这个系统的目的是创建一个数码相机的先进操作系统。不过，后来发现市场需求不够大，加上智能手机市场快速成长，于是 Android 被改造为一款面向智能手机的操作系统，并于 2005 年 8 月被 Google 收购。

2007 年 11 月，Google 与 84 家硬件制造商、软件开发商及电信营运商成立开放手持设备联盟来共同研发改良 Android 系统。随后，Google 以 Apache 免费开放源代码许可证的授权方式，发布了 Android 的源代码，让生产商推出搭载 Android 的智能手机，Android 操作系统后来更逐渐拓展到平板电脑及其他领域。

- Android Wear 是专为智能手表等可穿戴式设备所设计的一个 Android 系统分支。
- Android TV 是专为家用电视所设计的一个 Android 系统分支。
- Android Auto 是专为汽车所设计的一个 Android 系统功能。

Android 执行于 Linux kernel 之上，但并不是 GNU/Linux。在一般 GNU/Linux 里支持的功能，Android 大都没有支持，包括 Cairo、X11、Alsa、FFmpeg、GTK、Pango 及 Glibc 等都被移除。

Android 去除了 Linux 中的本地 X Window System，也不支持标准的 GNU 库，这使得 Linux 平台上的应用程序移植到 Android 平台上变得困难。另外，Android 又以 bionic 取代 Glibc、以 Skia 取代 Cairo、再以 opencore 取代 FFmpeg 等。

1.2 Histroy

2003 年 10 月，Andy Rubin 在美国加利福尼亚州帕洛阿尔托创建了 Android 科技公司（Android Inc.），并与 Rich Miner、Nick Sears、Chris White 共同发展这家公司。

2005 年 8 月 17 日，Google 收购了 Android 科技公司并正式进入移动领域。Android 科技公司成为 Google 所拥有的全资子公司，所有 Android 科技公司的员工都被并入 Google。

Google 的合作平台为 Android 提供了广阔的市场，Google 给予各大硬件制造商、软件开发商一个灵活可靠的系统升级承诺，并保证将给予它们最新版本的操作系统。

2007 年 11 月 5 日，在 Google 的领导下成立了开放手持设备联盟（Open Handset Alliance），最早的一批成员包括 Broadcom、HTC、Intel、LG、Marvell 等公司。

开放手持设备联盟的创建目的是为了创建一个更加开放自由的移动电话环境。

在开放手持设备联盟创建的同一日，联盟就对外展示了他们的第一个产品：一部搭载了以 Linux 2.6 为核心基础的 Android 操作系统的智能手机。世界上第一部真正意义上使用 Android 操作系统的设备是 2008 年 10 月 22 日发布的 HTC Dream。

同时，一个负责持续发展 Android 操作系统的开源代码项目成立了 AOSP（Android Open Source Project）。除了开放手持设备联盟之外，Android 还拥有全球各地开发者组成的开源社区来专门负责开发 Android 应用程序和第三方 Android 操作系统来延长和扩展 Android 的功能和性能。

1.3 Releases

Android 1.0 beta 发布于 2007 年 11 月 5 日，它作为一个面向开发者的软件开发包（SDK）进行发布，至今已经发布了多个更新。这些更新版本都在前一个版本的基础上修复了 Bug 并且添加了前一个版本所没有的新功能。

从 2009 年 5 月开始，Android 操作系统改用甜点来作为版本代号，这些版本按照大写字母的顺序来进行命名：纸杯蛋糕（Cupcake）、甜甜圈（Donut）、闪电泡芙（Éclair）、凍酸奶（Froyo）、姜饼（Gingerbread）、蜂巢（Honeycomb）、冰激凌三明治（Ice Cream Sandwich）、果冻豆（Jelly Bean）、奇巧（KitKat）、棒棒糖（Lollipop）。

此外，Android 操作系统还有两个预发布的内部版本，它们分别是原子小金刚（Astro）和机器人班亭。

1.4 License

Android 操作系统使用开放免费代码许可证，一切代码为公开免费的。Google 将 Android 的大部分以 Apache 开源条款 2.0 发布，剩下的 Linux 内核部分则继承 GPLv2 许可，AOSP 包括了智能手机网络和电话协议栈等智能手机所必需的功能。

Google 也不断发布问卷和开放修改清单、更新情况和代码来让任何人看到并且提出他们的意见和评论，以便按照用户的要求改进 Android 操作系统。

Android 操作系统是完全免费开源的，任何厂商都不须经过 Google 和开放手持设备联盟的授权随意使用 Android 操作系统。但是，制造商不能在未授权下在产品上使用 Google 的标志和应用程序，例如 Google Play 等。除非 Google 证明其生产的产品设备符合 Google 兼容性定义文件（CDD），这才能在智能手机上预装 Google Play Store、Gmail 等 Google 的私有应用程序，并且获得 CDD。

此外，智能手机厂商也可以在其生产的智能手机上印上“With Google”的标志。

Chapter 2

Installation

Chapter 3

Update

Android Studio 本身提供了在线增量更新功能，不过也可以手动进行更新。

首先，查看当前 Android Studio 的版本信息。

```
Android Studio (Beta) 0.8.9
Build #AI-135.14.4660, built on September 3, 2014
JRE:1.80_20-b26 amd64
JVM: Java HotSpot(TM) 64-bit Server VM by Oracle Corporation
```

访问studio-patches-updates可以看到最新的 build number 和版本号。

```
<channel id="AI-0-beta" name="Android␣Studio␣updates" status="beta" url="
    http://tools.android.com/recent" feedback="https://code.google.com/p/
    android/issues/entry?template=Android+Studio+bug" majorVersion="0">
  <build number="135.1404660" version="0.8.9">
    <message>
      <![CDATA[
        <html> A new Android Studio 0.8.9 is available in the beta channel
            .<p/> </html>
      ]]>
    </message>
    <button name="Download" url="http://developer.android.com/sdk/
        installing/studio.html" download="true"/>
    <button name="Release␣Notes" url="http://tools.android.com/recent"/>
    <patch from="135.1245622" size="11"/>
    <patch from="135.1248636" size="11"/>
    <patch from="135.1267975" size="11"/>
    <patch from="135.1281642" size="7"/>
    <patch from="135.1295215" size="7"/>
    <patch from="135.1331330" size="4"/>
```

```
      <patch from="135.1339820" size="4"/>
      <patch from="135.1371801" size="1"/>
      <patch from="135.1391024" size="1"/>
    </build>
  </channel>
```

可以手动下载更新补丁并安装，例如：

```
$ wget http://dl.google.com/android/studio/patches/AI-135.1404660-135.15254
$ cd android-studio
$ cp ~/AI-135.1404660-135.1525417-patch-unix.jar .
$ java -classpath AI-135.1404660-135.1525417-patch-unix.jar com.intellij.u
$ rm AI-135.1404660-135.1525417-patch-unix.jar
```

或者，可以使用 Android Studio 提供的脚本来手动升级。

为了彻底解决 Android Studio 在线更新时出现的“Connection failed. Please check your network connection and try again.”问题，可以在 studio.vmoptions 或 studio64.vmoptions 中增加如下两行配置：

或者，通过如下的命令直接修改环境变量 $REQUIRED_JVM_ARGS：

如果不需要进行更新，那么将会出现如下的提示：

Chapter 4

Activity

4.1 Introduction

Activity 等同于 J2ME 的 MIDlet，一个 Activity 类别负责创建视窗，一个活动中的 Activity 就是在 foreground（前景）模式，背景执行的程序叫做 Service。两者之间通过由 ServiceConnection 和 AIDL 连接，达到复数程序同时执行的效果。如果执行中的 Activity 全部画面被其他 Activity 取代时，该 Activity 便被停止，或者被系统清除。

View 等同于 J2ME 的 Displayable，程序人员可以通过 View 类别与“XML layout”档将 UI 放置在视窗上，并可以利用 View 实现所谓的 Widgets，其实 Widget 只是 View 的一种，所以可以使用 xml 来设计 layout。

ViewGroup 是各种 layout 的基础抽象类别，ViewGroup 之内还可以有 ViewGroup。

View 的构造函数不需要在 Activity 中调用，但是 Displayable 的是必须的，在 Activity 中，要通过 findViewById() 来从 XML 中取得 View，Android 的 View 类的显示很大程度上是从 XML 中读取的。

View 与事件息息相关，两者之间透过 Listener 结合在一起，每一个 View 都可以注册 event listener。例如，当 View 要处理用户触碰的事件时，就要向 Android 框架注册 View.OnClickListener。另外还有 Image 等同于 J2ME 的 BitMap。

Chapter 5

UI

Android 操作系统支持多语言。

Android 操作系统支持更大的分辨率，VGA，2D 显示，3D 显示都给予 OpenGL ES 3.0 标准规格（4.3 版本开始支持 OpenGL ES 3.0），并且支持传统的智能手机。

Chapter 6

Fragment

碎片（fragment）是自 Android 3.0 之后引入的全新概念，目前已经广泛应用于 Android 手机和平板中。

Chapter 7

Broadcast Receiver

作为原设计给智能手机使用的操作系统，Android 操作系统原生支持短信和邮件，并且支持所有的云端信息和服务器信息。

Chapter 8

Storage

8.1 File Storage

8.2 Shared Preferences

8.3 Database Storage

Android 操作系统内置 SQLite 小型关联式资料库管理系统来负责存储数据。

Chapter 9

Content Provider

Chapter 10

Multimedia

Android 操作系统本身支持以下格式的音频/视频/图片媒体：WebM、H.263, H.264（in 3GP or MP4 container）、MPEG-4 SP、AMR, AMR-WB（in 3GP container）、AAC, HE-AAC（in MP4 or 3GP container）、MP3、MIDI、Ogg Vorbis、FLAC、WAV、JPEG、PNG、GIF、BMP。如果用户需要播放更多格式的媒体，可以安装其他第三方应用程序。

10.1 Streaming Media

Android 操作系统支持 RTP/RTSP（3GPP PSS, ISMA）的流媒体以及（HTML5 <video>）的流媒体，同时还支持 Adobe 的 Flash，在安装了 RealPlayer 之后，还支持苹果公司的流媒体。

10.2 Camera

10.3 Audio

10.4 Video

10.5 Debug

Chapter 11

Service

11.1 Multithreading

11.2 Multitasking

Android 操作系统支持本地的多任务处理。

Chapter 12

Network

Android操作系统支持所有的网络制式(包括GSM/EDGE、IDEN、CDMA、TD-SCDMA、EV-DO、UMTS、Bluetooth、Wi-Fi、LTE、NFC和WiMAX等)。

12.1 Web Browser

Android操作系统中内置的网页浏览器基于WebKit核心，并且采用了Chrome V8引擎。在Android 4.0内置的浏览器测试中，HTML5和Acid3故障处理中均获得了满分。

Android从2.2版及之后就能原生支持Flash，4.0版本后去除对Flash的支持。

12.2 Wireless

Android操作系统支持用户使用本机实现“无线热点”，并且将本机的网络共享给其他智能手机，其他机器只需要通过WiFi查找到共享的无线热点就可以上网。

在Android OS v2.2版本之前的操作系统则需要通过第三方应用或者其他定制版系统来实现这个功能。

12.3 Screen Shot

从Android OS v4.0版本开始，Android操作系统便支持截图功能，该功能允许用户直接抓取智能手机屏幕上的任何画面，用户还可以通过编辑功能对截图进行处理，还可以通过蓝牙/E-mail/微博/共享等方式发送给其他用户或者上传到网络上，也可以拷贝到电脑中。

Chapter 13

Location

Chapter 14

Sensor

Chapter 15

Architecture

15.1 Linux Kernel

Android 操作系统的核心属于 Linux 内核的一个分支，具备典型的 Linux 调度和功能。除此之外，Google 为了能让 Linux 在移动设备上良好的运行，对其进行了修改和扩充。

除了核心之外，则是中间层、数据库和用 C/C++ 编写的 API 以及应用程序框架。

2008 年，Patrick Brady 于 Google I/O 演讲“Anatomy & Physiology of an Android”，并提出的 Android HAL 架构图。

HAL 以 *.so 档的形式存在，可以把 Android framework 与 Linux kernel 隔开，这种中间层的方式使得 Android 能在移动设备上获得更高的运行效率。

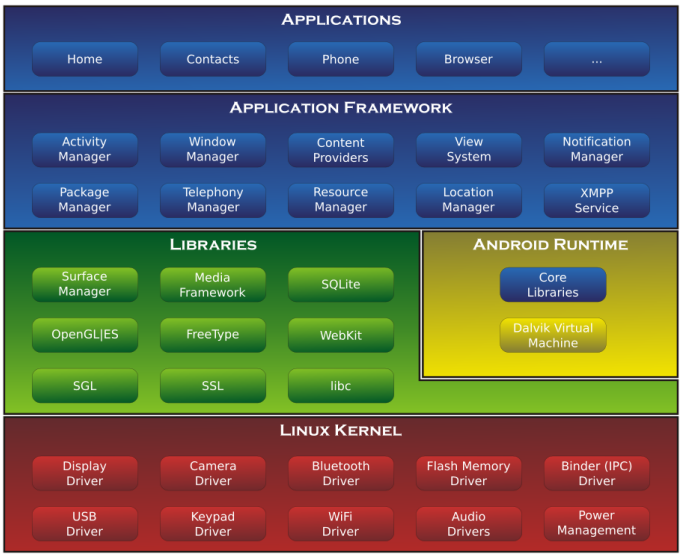
Android 的 Linux kernel 控制包括安全、存储器管理、进程管理、网络堆叠、驱动程序模型等。

Google 在 Android 的核心中加入了移动设备电源管理功能“wakelocks”，该功能用于管理移动设备的电池性能。不过，wakelocks 功能并没有被加入到 Linux 内核的主线开放和维护中，因为 Linux 内核维护者认为 Google 没有向他们展示这个功能的意图和代码。

2010 年 2 月 3 日，由于 Google 在 Android 核心开发方面和 Linux 社区方面开发的不同步，Linux 内核开发者 Greg Kroah-Hartman 将 Android 的驱动程序从 Linux 内核“状态树”（“staging tree”）上除去。

2010 年 4 月，Google 宣布将派遣 2 名开发人员加入 Linux 内核社区，以便重返 Linux 内核。

2010 年 9 月，Linux 内核开发者 Rafael J. Wysocki 添加了一个修复程序来确保 Android 的“wakelocks”可以轻松地与主线 Linux 内核合并，在接下来的 Linux 3.3 中大部分代码



的集成完成。

15.2 Library

Android 库是操作系统与应用程序的沟通桥梁，可以划分为函数层和虚拟机器。

Android 使用的工具链 Bionic Libc 是 Android 改良 libc 的版本。Google 希望用 Bionic libc 来取代 glibc，它的发展目标是达到轻量化以及高运行速度。

bionic/libc/kernel/ 并非标准的内核头文件（kernel header files），而且 Android 的内核头文件是利用工具由 Linux 内核的头文件所产生的，这样做是为了保留常数、数据结构与宏。

Android 同时包含了 Webkit，并使用 Surface flinger 将 2D 或 3D 的内容显示到屏幕上。

Android 采用 OpenCORE 作为基础多媒体框架，OpenCORE 可分 7 大块：PVPlayer、PVAuthor、Codec、PacketVideo Multimedia Framework（PVMF）、Operating System Compatibility Library（OSCL）、Common、OpenMAX。

Android 使用 Skia[1] 为核心图形引擎，Skia 与 Linux Cairo 功能相当，以搭配 OpenGL/ES 使用。

Android 的多媒体数据库采用 SQLite 数据库系统。数据库又分为共用数据库及私用数据库。用户可通过 ContentResolver 类别取得共用数据库。

[1] 2005 年 Skia 公司被 Google 收购，2007 年初，Skia GL 源码被公开，目前 Skia 也是 Google Chrome 的图形引擎。

15.2.1 JVM

Android 的中间层多以 Java 实现，并且采用特殊的 Dalvik 虚拟机。

大多数虚拟机（包括 JVM）都是一种堆栈机，而 Dalvik 虚拟机则是寄存器机，因此变量皆存放于暂存器中，并且减少了虚拟机的指令。

一般而言，两种架构各有优劣，其中基于堆栈的机器需要更多指令，而基于寄存器的机器指令更长。

- Dalvik 虚拟机早期并没有使用即时编译[2]（JIT）技术。
- Dalvik 虚拟机有自己的字节码, 并非使用 Java 字节码。
- Dalvik 基于暂存器，而 JVM 基于堆栈。
- Dalvik VM 通过 Zygote 进行类的预加载，Zygote 会完成虚拟机的初始化，也是与 JVM 不同之处。

现在，作为 Android 操作系统的核心组成部分之一，Dalvik 虚拟机可以支持已转换为.dex[3]（即“Dalvik Executable”）格式的 Java 应用程序的运行。

Dalvik 虚拟机可以有多个实例，每个 Android 应用程序都用一个自属的 Dalvik 虚拟机来执行，让系统在执行程序时可达到优化。Dalvik 虚拟机并非执行 Java 字节码，而是执行一种称为.dex 格式的文件。

Dalvik 解释器采用预先算好的 Goto 地址，每个指令对内存的访问都在 64 字节边界上对齐。这样可以节省一个指令后进行查表的时间。为了强化功能, Dalvik 还提供了快速翻译器（Fast Interpreter）。

在 Android 应用开发中，使用 dx 工具来将 Java .class 转换为 DEX 格式代码。一个 dex 文件通常会有多个.class，各个.class 中重复的字符串和其他常数只在 DEX 中存放一次，以节省空间。不过，dex 有时必须进行优化，因此会使文件大小增加 1-4 倍，以 ODEX 结尾。

Java 字节码（bytecode）被转换成 Dalvik 虚拟机所使用的替代指令集，一个未压缩 dex 文件通常稍小于一个已经压缩的.jar 文件。

当安装到 Android 设备时，Dalvik 可执行文件可能会被修改。例如，为了获得进一步优化，虚拟机可能会调整文件内部分数据的端序、内联一些函数和简单的结构体，并短路掉一些不必要的操作。

15.2.2 HAL

Android 的硬件抽像层是以封闭源码形式提供的硬件驱动模块。

[2]从 Android 2.2 开始, Dalvik 虚拟机也支持 JIT。

[3].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。

通过硬件抽象层，可以把 Android framework 与 Linux kernel 隔开，让 Android 不至过度依赖 Linux kernel，以实现“内核独立”（kernel independent），也让 Android framework 的开发能在不考量驱动程序实现的前提下进行。

HAL stub 是一种代理人的概念，stub 是以 *.so 档的形式存在。Stub 向 HAL“提供”操作函数，并由 Android runtime 向 HAL 取得 stub 的操作，再回调这些操作函数。HAL 里包含了许多的 stub（代理人）。Runtime 只要说明“类型”（即 module ID），就可以取得操作函数。

Android 为了达到商业应用，必须移除被 GNU GPL 授权证所约束的部份，因此 Android 并没有用户层驱动（user space driver）。所有的驱动还是在内核空间中，并以 HAL 避开版权问题。

15.3 Application

Android 操作系统中的应用程序大部分都是由 Java 编写的，但是 Android 却是以转换为 Dalvik executables 的文件在 Dalvik 虚拟机上运行的。

绝大多数 Android 设备都允许用户安装 APK 格式的文件来使用应用程序。具体来说，运行程序时，应用程序的代码会被实时转变为 Dalvik dex-code（Dalvik Executable），然后 Android 操作系统通过使用实时编译的 Dalvik 虚拟机来执行程序。

Android 应用程序包文件 (APK) 是一种 Android 操作系统上的应用程序安装文件格式，其英文全称为“application package file”。

为了在 Android 设备上运行应用程序，必须先进行编译，然后被打包成为一个被 Android 系统所能识别的文件才可以被运行，而这种能被 Android 系统识别并运行的文件格式便是“APK”。

APK 文件基于 ZIP 文件格式，它与 JAR 文件的构造方式相似，其互联网媒体类型是 application/vnd.android.package-archive。

具体来说，APK 文件内包含被编译的代码文件 (.dex 文件)、文件资源（resources）、assets、证书（certificates）、和清单文件（manifest file）。

- META-INF 文件夹
  - MANIFEST.MF: 清单信息（Manifest file）
  - CERT.RSA: 保存着该应用程序的证书和授权信息。
  - CERT.SF: 保存着 SHA-1 信息资源列表，比如：

```
Signature-Version: 1.0
 Created-By: 1.0 (Android)
```

```
SHA1-Digest-Manifest: wxqnEAI0UA5nO5QJ8CGMwjkGGWE=
...
Name: res/layout/exchange_component_back_bottom.xml
SHA1-Digest: eACjMjESj7Zkf0cBFTZ0nqWrt7w=
...
Name: res/drawable-hdpi/icon.png
SHA1-Digest: DGEqylP8W0n0iV/ZzBx3MW0WGCA=
```

- res：APK 所需要的资源文件夹。
- AndroidManifest.xml：一个传统的 Android 清单文件，用于描述该应用程序的名字、版本号、所需权限、注册的服务、链接的其他应用程序。该文件使用 XML 文件格式，可以编译为二进制的 XML，使用的工具为 AXMLPrinter2 或 apktool。
- classes.dex：classes 文件通过 DEX 编译后的文件格式，用于在 Dalvik 虚拟机上运行的主要代码部分。
- resources.arsc

Android 中并不自带 Java 虚拟机，因此无法直接运行 Java 程序。不过 Android 平台上提供了多个 Java 虚拟机供用户下载使用，安装了 Java 虚拟机的 Android 系统可以运行 Java ME 的程序。

当 Android 启动时，Dalvik VM 监视所有的程序（APK），并且创建依存关系树，为每个程序优化代码并存储在 Dalvik 缓存中。Dalvik 第一次加载后会生成 Cache 文件，以提供下次快速加载，所以第一次会很慢。

15.4 Hardware

Android 操作系统大多搭载在使用了 ARM 架构的硬件设备上，但是同样也有支持 X86 架构的 Android 操作系统（比如 Google 的 Google TV 就是使用一个特别定制的 X86 架构版本的 Android 操作系统）。

Android 操作系统支持识别并且使用视频/照片摄像头，多点电容/电阻触摸屏，GPS，加速计，陀螺仪，气压计，磁强计，键盘，鼠标，USB Disk，专用的游戏控制器，体感控制器，游戏手柄，蓝牙设备，无线设备，感应和压力传感器，温度计，加速 2D 位位块传输（硬件方向，缩放，像素格式转换）和 3D 图形加速。

15.4.1 Bluetooth

Android 支持 A2DP、AVRCP、发送文件（OPP)、访问电话簿（PBAP)、语音拨号和发送智能手机之间的联系，同时支持键盘、鼠标和操纵杆（HID)。

15.5 Multi-Point Touch

Android 支持本地的多点触摸，而且该功能是内核级别（为了避免对苹果公司的触摸屏技术造成侵权)。

15.6 Voice

除了支持普通的电话通话之外，Android 操作系统从最初的版本开始就支持使用语音操作来使用 Google 进行网页搜索等功能。

从 Android OS v2.2 开始，语音功能还可以用来输入文字、语音导航等功能。

Chapter 16

Development

在早期的 Android 应用程序开发中，通常通过在 Android SDK 中使用 Java 作为编程语言来开发应用程序。开发者亦可以通过在 Android NDK 中使用 C 语言或者 C++ 语言来作为编程语言开发应用程序。

16.1 Init Tool

Repo 是 Android 用来辅助 Git 工作的一个工具，用以初始化源码。

16.2 Build Tool

16.2.1 Apache Ant

Apache Ant 是一个将软件编译、测试、部署等步骤联系在一起加以自动化的一个工具，大多用于 Java 环境中的软件开发。

默认情况下，它的 buildfile(XML 文件) 名为 build.xml。每一个 buildfile 含有一个 <project> 和至少一个默认的 <target>，这些 targets 包含许多 task elements。每一个 task element 有一个用来被参考的 id，此 id 必须是唯一的。

Listing 16.1: build.xml

```
<?xml version="1.0" ?>
<project name="Hello␣World" default="execute">
  <target name="init">
    <mkdir dir="build/classes"/>
    <mkdir dir="dist"/>
```

```
    </target>
    <target name="compile" depends="init">
      <javac srcdir="src" destdir="build/classes"/>
    </target>

    <target name="compress" depends="compile">
      <jar destfile="dist/HelloWorld.jar" basedir="build/classes"  />
    </target>

    <target name="execute" depends="compile">
      <java classname="HelloWorld" classpath="build/classes"/>
    </target>

  </project>
```

16.2.2 Apache Maven

Apache Maven 曾是 Jakarta 项目的子项目，现为独立 Apache 项目。

作为一个软件（特别是 Java 软件）项目管理及自动构建工具，Maven 基于项目对象模型概念以利用一个中央信息片断能管理一个项目的构建、报告和文档等步骤。

Maven 项目使用项目对象模型（Project Object Model，POM）来配置，并且项目对象模型存储在命名为 pom.xml 的文件中。

Listing 16.2: pom.xml

```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0</version>

  <!-- library dependencies -->

  <dependencies>
    <dependency>
```

```
        <!-- coordinates of the required library -->

        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>3.8.1</version>

        <!-- this dependency is only used for running and compiling tests -->

        <scope>test</scope>

      </dependency>
    </dependencies>
</project>
```

16.2.3 Gradle

Gradle 是一个基于 Apache Ant 和 Apache Maven 概念的项目自动化建构工具，使用一种基于 Groovy 的特定领域语言来声明项目设置，而不是传统的 XML。

Gradle 当前支持 Java、Groovy 和 Scala 语言。

Gradle 与 Ant 有很紧密集成，因此可以在构建时直接导入 Ant 构建脚本。例如，下面的例子展示了一个简单的 Ant target 被引入为一个 Gradle task。

Listing 16.3: build.xml

```
<project>
  <target name="ant.target">
    <echo message="Running ant.target!"/>
  </target>
</project>
```

Listing 16.4: build.gradle

```
ant.importBuild 'build.xml'
```

运行 gradle ant.target 将显示如下结果：

```
$ gradle ant.target
:ant.target
[ant:echo] Running ant.target!

BUILD SUCCESSFUL
```

16.3 Debug

16.3.1 Lint

在计算机科学中，lint 是一种工具程序的名称，它用来标记源代码中的某些可疑的、不具结构性（可能造成 bug）的段落。

lint 最早于 1979 年，在贝尔实验室发表的 UNIX 第七版中出现，派生自可移植 C 编译器（pcc）。

Chapter 17

Security

Android 操作系统使用了沙箱（sandbox）机制，所有的应用程序都会先被简单地解压缩到沙箱中进行检查，并且将应用程序所需的权限提交给系统，并且将其所需权限以列表的形式展现出来，供用户查看。例如一个第三方浏览器需要“连接网络”的权限，或者一些软件需要拨打电话，发送短信等权限。用户可以根据权限来考虑自己是否需要安装，用户只有在同意了应用程序权限之后，才能进行安装。

美国国家安全局在 2012 年 1 月发布 SE Android（Security Enhanced Android，后改名为 SE for Android，Security Enhancements for Android）开源项目和代码，使 Android 系统支持强制访问控制（Mandatory Access Control）以增加系统安全性。

17.1 Android Virus

2010 年 8 月，卡巴斯基病毒实验室报告称发现了 Android 操作系统上首个木马程序，并将其命名为“Trojan-SMS.AndroidOS.FakePlayer.a”，这是一个通过短信方式感染智能手机的木马，并且已经感染了一定数量的 Android 设备。

除了短信感染方式，这些 Android 木马还可以伪装成一些主流的应用程序，并且还可以隐藏在一些正规的应用程序之中。

17.2 Root

Android 系统的 root 与 Apple iOS 系统的越狱类似，从而使得用户可以获取 Android 操作系统的超级用户权限。

通常情况下，用户可以通过 root 来越过手机制造商的限制，使得用户可以卸载手机

制造商预装在手机中某些应用程序，以及运行一些需要超级用户权限的应用程序。

root 的基本原理就是利用系统漏洞，将 su 和对应的 Android 管理应用复制到/system 分区（例如/system/xbin/su），并用 chmod 命令为其设置可执行权限和 setuid 权限。

手机制造商原始出厂的手机并未开放 root 权限，获取 root 的方法都是不受官方支持的，因此目前获取 root 的方法都是利用系统漏洞实现的。

不同手机厂商可能存在的漏洞不同，也就导致了不同手机 root 的原理可能不同。不过，不管采用什么原理实现 root，最终都需要将 su 可执行文件复制到 Android 系统的/system 分区。

目前最广泛利用的系统漏洞是 zergRush，该漏洞适用于 Android 2.2-2.3.6 的系统，其它的漏洞还有 Gingerbreak、psneuter 等。

zergRush 漏洞必须在 adb shell 下运行，而 adb shell 只能将手机用 USB 数据线与 PC 连接之后才能在 PC 上打开，因此目前常用的 root 工具都是 PC 客户端程序，通过 Android 系统的 adb shell 运行漏洞利用程序。

为了让用户可以控制 root 权限的使用，防止其被未经授权的应用所调用，通常还有一个 Android 应用程序来管理 su 程序的行为。

在 Android 设备上直接运行的 root 工具通常以 App 的形式在各类应用商店（非 Google 官方）上发布，但是利用这些应用获取 root 权限之后，应用本身就会成为 root 权限的授权者，其他应用使用 root 权限时都需要通过此授权者的允许。例如，有些应用会在通知栏推送广告，或者在获取 root 权限之后将自己变成系统应用。