【eoe 特刊】第十期： Android2.2 特色详解

目录

【本期简介】.....3

【新书推荐】

《It's Android Time—Google Android 创赢路线》.....4

【Android2.2 特色详解】

1.1 11 项重点升级 Android 2.2 全方位解析.....5

1.2 开发者或想使用的 7 个 Android2.2 新特性.....11

1.3 详解 android2.2 中全新的 JIT 内核编译器.....13

【android 行业研究报告】

2.1 NPD 统计 Android 美国本土份额在 2010 一季度超越 iPhone OS.....15

2.2 ComScore：Android 手机在美市场份额上升至 13%.....16

2.3 admob 5 月统计数据.....17

【android 技术盛宴】

3.1 滑动式抽屉(SlidingDrawer)的使用.....20

3.2 Android 启动过程详解.....27

3.3 Android 消息系统.....31

【其他】

4.1 BUG 提交.....36

4.2 关于 eoeAndroid.....36

4.3 新版优亿市场上线.....36

本期简介：

本期特刊主要是介绍 android 最新的 2.2 系统。全面介绍 android2.2 中的重点升级，重点介绍 JIT 内核编译器，本期亮点是推出了三份最近的 android 行业研究报告，技术盛宴版块里有丰富的内容，给大家慢慢研究，最后有最新的优亿市场介绍。如果你对特刊有更好的建议，请联系 eoeandroidteam@eoemobile.com

【新书推荐 】 《It's Android Time—Google Android 创赢路线》

作者： eoeAndroid 开发者社区

出版社：电子工业出版社

ISBN：9787121111556

上架时间：2010-7-26

出版日期：2010 年 7 月

开本：16 开

页码：588

内容简介

It's Android Time！我们深信这个时代很快就会到来，我们需要做的就是早早进入这个行业，对其行业的趋势做相关的判断，对其中各式各样的产品及其方向都应该有所了解。本书对 Android 相关的产品定义和方向进行了详细的调查和分析，以实例的形式循序渐进地引导大家进一步了解 Android 的知识。

本书深入 Android 底层讲述如何进行底层开发，同时会站在更高的层面和方向上看待和剖析 Android 及其开发相关的内容。本书总体的策划思路是：我们将现在看到的或者想到的产品方向进行汇总和归纳，评估每个方向的市场容量，然后选取市场容量足够大的方向用一个或者几个例子讲述如何在这个方向上进行产品的规划、设计、开发和发布等。我们力求选取的方向清晰，又保证选取的方向能被实践证明具有可操作性。

马上订购：http:/www.china-pub.com/196898
进入讨论区：http:/www.eoeandroid.com/thread-26455-1-1.html
试读下载：http:/www.eoeandroid.com/thread-21853-1-1.html

【Android2.2 特色详解】

1.1 11 项重点升级 Android 2.2 全方位解析

北京时间 5 月 20 日 22:30 分，谷歌终于在开发者峰会 Google I/O 大会上公布了 Android 2.2（开发代号为 Froyo）版操作系统，除了此前我们论坛曝光的几项特征，Android 2.2 还为用户带来了更多全新体验和提升，无论是谷歌声称的“Froyo 具有世界上最快的浏览”，还是最高 5 倍的性能提高，都使 Froyo 成为众人关注的焦点。为了让用户体验 Android 2.2 的强大升级，我们在此总结了 11 项 2.2 版比较重要的提升和改变，希望能够给期待 Froyo 的用户带来一些指导。

一、系统速度提高 2 到 5 倍

Android 2.2 采用了一个 Just In Time (JIT)编译器，这个编译器可以使系统提高 2 到 5 倍的速度。当应用程序启动之后，JIT 编译器会将虚拟机的字节码汇编成本机代码。通过现场演示，我们能够发现 JIT 编译器使得系统运行的速度提高了 2.2 倍。

系统速度提高 2 到 5 倍

二、新增网络共享功能

谷歌 2.2 版 Android 操作系统将支持 USB tethering(网络共享功能)，从而实现手机与笔记本电脑等其他设备共享网络连接。另外 2.2 版本还将支持 Wi-Fi hotspot 功能，这意味着新系统可以让用户的 Android 手机变成一个移动的 Wi-Fi 热点，进而对附近设备进行 Wi-Fi 网络分享。

网络共享功能

三、具有更快的浏览器

谷歌对 Android 2.2 版本中浏览器作出了进一步的优化，在加入 V8 JavaScript 脚本后，Android 2.2 的 Web 浏览器体验非常出色。与 2.1 版 Android 相比，能够提高 2 到 3 倍的性能，因此谷歌声称 Android 2.2 具有世界上最快的手机浏览器。

更快的浏览器

四、语音识别功能

谷歌在语音命令方面的技术已经日趋成熟，因此在谷歌也将这一功能延伸到 Android 2.2 版系统之中。在现场语音输入的演示中，我们必须承认其表现非常出众。对于演示人员说出的长句，设备能够快速、准确的识别，并进行处理。目前用户可以通过语音输入的方式进行搜索、翻译以及开启应用程序等其他操作。

语音翻译功能

五、Flash 升级至 10.1 版本

除了语音输入方面的优势之外，Android 2.2 优于 iPhone 操作系统的另外一项特点，就是对 Flash 功能的支持。2.2 版 Android 系统在升级至 Flash 10.1 功能之后，用户可以在访问 Flash 网站和 Flash 游戏时获得更好的视觉体验。

六、支持应用程序存至存储卡及程序自动更新功能

即将到来的 2.2 版本将允许在 SD 卡中存储应用程序，这表明 2.2 版 Android 将在很大程度上改善手机内部存储过小问题。另外 2.2 版本还将支持应用程序自动更新功能，省去了用户定期对众多应用手动更新的麻烦。不过这一功能也会给用户增加很多数据流量。

应用程序存至存储卡及程序自动更新功能

七、新增向开发商发送错误报告功能

为了方便解决用户的各种问题，2.2 版本还支持向开发者发送错误报告，并且操作十分简单。用户只需要在发生错误时点击一个按键，就可以将错误信息发送给开发商，以帮助开发商更快地解决问题，缩小应用发展的进程。

八、增添云接口备份应用数据

此前 Android 手机的应用程序一直没有数据备份功能，这让很多用户颇为烦恼，好在 2.2 版终于解决了这一问题现，在新 Android 系统新增了云接口，这意味着用用户可以将手机数据资料通过云接口备份，此后应用程序服务将通过信息服务发到 Android 设备上。

云接口备份应用数据

九、增加企业用户相关功能

企业用户也将得益于 Android 2.2 版本，因为 2.2 版本将更好的支持 Exchange 功能，它将支持自动发现、安全政策和 GAL 查找功能，此外设备还能通过更好、更有效的方式管理。此外，设备管理 API 还允许开发人员编写应用程序，以此来控制设备屏幕锁定超时或者是密码设置方面的安全功能。

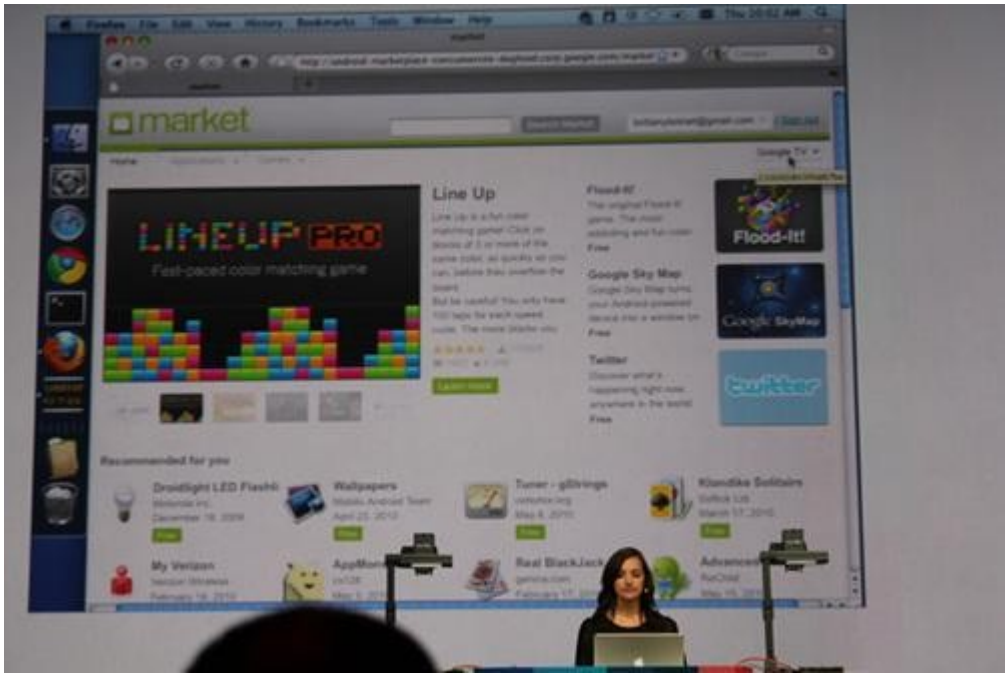
增加企业功能

十、简化 PC Android 应用商城下载步骤

此前 Android 用户通过 PC 方式获得应用软件，都需要先下载到 PC 上，然后通过 PC 将软件移至手机上。而 2.2 版本的 Android 则简化了这一繁琐的步骤，

在登录 PC 之后，Android 商城将自动识别用户的手机信息，在经过确认之后，应用软件将通过无线直接下载到手机中，而无需 PC 的中转。

同样的提升还体现在音乐方面，谷歌将推出一项新产品，能够让用户将 PC 中的音乐通过网络传输到手机中。



PC 浏览 Android 应用商城

十一、新增应用程序快速搜索功能

另外 Android 2.2 还提供了应用程序的快速搜索框，用户可以通过输入关键字的方式更为轻松的下载应用程序，同时开发人员也可以将应用程序增添到这一选项中。

总结：以上就是 Andrid 2.2 版本为我们带来的 11 项比较重要的升级，简单来说 2.2 版本的确给我们带来了很多惊喜。我们很高兴能够看到 Froyo 在系统性能、浏览器性能方面的提高；应用程序在手机、PC 方面搜索、安装、备份、更新等一系列的全面改善；Flash 10.1、语音输入、企业功能的添加；以及网络共享等最新的前端应用的体验。简单来说，Andrid 2.2 将是一个功能更丰富，操作更人性化的系统，用户体验将大大提高。

1.2 开发者或想使用的 7 个 Android2.2 新特性

虽然从版本号来看，Android 2.2(代号 Froyo)只能算是一次小的升级，但事实却不是这样的，这次升级将给开发人员和最终用户带来期待已久的许多功能，笔者参加了 Google I/O 大会，亲眼见证了 Froyo 的发布，整理了 7 个我认为是开发人员急于想使用的新特性。

1、 Flash 10.1 和 AIR 支持

关于 Flash 在移动平台上的去留是最近的一个热门话题，主要是由苹果掌门 Jobs 发起的抵制 Flash 运动，但令人惊奇的是 Android 2.2 中内置了对 Flash 的支持，不管未来 Flash 是否会走向衰落，但目前仍然是主流，互联网上到处充斥着 Flash 应用，剔除 Flash 会让很多社交应用都无法进入智能手机。

从 Froyo 开始，Android 用户可以从 Android Market 下载 Flash 10.1 Beta 版本以及 AIR 支持，这个决定大大扩展了 Android 用户可以访问的 Web 应用程序和网站的数量，拓宽了 Android 社区开发的范围。

但这对 Android 开发人员来说可能是把双刃剑，它将对 Android Market 产生多大的影响?过不了多久就会产生大量的 Flash 应用程序(我们也许很快就会看到德克萨斯扑克出现在 Android Market 中)，也许很多非 Flash 应用程序将会被逐渐淘汰，这对开发人员来说究竟是好事还是坏事呢?我们拭目以待吧!

2、 消息推送

开发人员现在可以利用另一个 Google 服务，Android 云到设备的消息框架(Android Cloud to Device Messaging (C2DM) Framework)，通过这个框架提供的服务，开发人员可以给 Android 设备开启有限的消息推送功能，可以去 Google Labs 网站去注册申请使用该服务，Android Market Web 版将会提供这个功能，用户通过电脑桌面购买 Android 应用程序，Android Market 通过推送功能将应用程序发送给指定的移动设备，这一技术能够克服因下载量大造成的网络堵塞。

3、 新的企业特性

Android 终于将自己定位为重要的企业移动应用平台，Android 2.2 SDK 包括新的设备管理 API(android.app.admin)，可以对设备实现远程管理和保护，也有专门针对设备安全保护的 API，包括强制密码策略，远程锁定和擦除设备数据的功能。例如，如果一名雇员丢失了他/她保存有敏感数据的手机，可以立即远程锁定，并擦除数据。

4、 性能改善

开发人员和用户都会受益于 Froyo 做出的大量性能改进，Google 采取了最严格的措施来提升系统性能和质量。

性能的提升主要得益于为 Dalvik VM 引入了 JIT(即时)编译器，根据 Google Android 开发团队的说法，Froyo 运行时性能是上一个版本的 2-5 倍，你可以在应用程序的 Android Manifest 文件中禁用 JIT 优化。Android 浏览器引入 V8 JavaScript 引擎后，速度也明显提高了很多，与 Android 2.1 浏览器相比，性能至少提高了 2-3 倍。

5、 音频和多媒体 API 改进

Android 多媒体 API 存在的大量问题在 Android 2.2 中得到了解决，例如，增加了 Audio Focus API 管理音频的播放，SoundPool API 也得到了更新，增加了回调功能，当一个项目完全载入后，可以暂停和恢复所有的流，应用程序不用再独立跟踪每个流。这项改进简化了应用程序的编码，提高了工作效率。

6、 全面的 SDK 增强

Android 2.2 SDK 中增加了许多 API，图形和游戏开发人员期待的 OpenGL ES 2.0 和 ETC1 纹理压缩得到了支持，另外语音识别服务(android.speech)也进行了大幅升级，周边 API，如对照相机和摄像机的支持也得到了极大的改进，新的 UI 模式管理器(android.app.UIModeManager)服务也为夜间模式，汽车驾驶模式和桌面模式做了调整。

在 Android 2.2 上，应用程序的安装路径不再受限制，支持安装到外部存储，如 SD 卡上，同时提供了新的数据备份服务，允许用户在 Android 设备之间无缝地过渡。

另外布局属性 fill_parent 改名为 match_parent 了，但对旧应用程序没有影响。

7、 Android Market 更新

Froyo 为 Android Market 带来了大量的更新，其中最有用的是内置的错误报告功能，如果你的应用程序在某个用户的手机上崩溃了，通过错误报告功能可以向开发人员反馈完整的信息(如设备配置和堆栈跟踪信息)，这样开发人员可以更有效地找出问题的根源，避免程序错误给差评留下借口。

OK，这就是我现在已经迫不及待地想体验的 Android 2.2 的 7 大特性，你同意我的观点吗?

1.3 详解 android2.2 中全新的 JIT 内核编译器

大家知道 Android 2.2 的新特性中有条是“使用了全新的 JIT 内核编译器”，不过我们之前并没有得到更多的消息，最新消息显示，经测试新版的 JIT 编译器为 2.2 版系统带来高达 500%效能提升。

Armor Games 公司的应用开发人员 Ian Douglas 展示了 Nexus One 运行 Android 2.2 版系统的测试结果，使用 Linpack 的 Benchmark 获取分值是目前大屏幕智能手机中运用比较多的基准测试之一。从结果中可以看到，Nexus One 安装了使用新版 JIT 编辑器的 Android 2.2 系统后，Benchmark 分数由 6-7 MFLOPS 提升至 38-40 MFLOPS，整体分值至少提升了五倍以上。

在 Google I/O 大会中，来自 JIT 编译负责团队的 Ben Cheng 和 Bill Buzbee 将会进行一场 A JIT Compiler for Android’s Dalvik VM 演示，相信新版系统大幅提升效能将会成为现实。

虽然这些测试是在 Nexus One 上进行的，但是新版的 JIT 编译器应该会在 Android 2.2 版中得到普及，届时所有 Android 手机升级系统之后都将能体验到性能提升带来的快感。不过目前仍不能评估这项更新见效的时间，大多数应用程序都将针对新编译器的优势进行测试和调整，而且手机厂商还要针对每一款机型开发不同的升级包。

JIT Compiler(Just-in-time Compiler) 即时编译

最早的 Java 建置方案是由一套转译程式（interpreter），将每个 Java 指令都转译成对等的微处理器指令，并根据转译后的指令先后次序依序执行，由于一个 Java 指令可能被转译成十几或数十几个对等的微处理器指令，这种模式执行的速度相当缓慢。

针对这个问题，业界首先开发出 JIT（just in time）编译器。当 Java 执行 runtime 环境时，每遇到一个新的类别（class：类别是 Java 程式中的功能群组），类别是 Java 程式中的功能群组－JIT 编译器在此时就会针对这个类别进行编译（compile）作业。经过编译后的程式，被优化成相当精简的原生型指令码（nativecode），这种程式的执行速度相当快。花费少许的编译时间来节省稍后相当长的执行时间，JIT 这种设计的确增加不少效率，但是它并未达到最顶尖的效能，因为某些极少执行到的 Java 指令在编译时所额外花费的时间可能比转译器在执行时的时间还长，针对这些指令而言，整体花费的时间并没有减少。

基于对 JIT 的经验，业界发展出动态编译器（dynamic compiler），动态编译器仅针对较常被执行的程式码进行编译，其余部分仍使用转译程式来执行。也就是说，动态编译器会研判是否要编译每个类别。动态编译器拥有两项利器：一是转译器，另一则是 JIT，它透过智慧机制针对每个类别进行分析，然后决定使用这两种利器的哪一种来达到最佳化的效果。动态编译器针对程式的特性或者是让程式执行几个循环，再根据结果决定是否编译这段程式码。这个决定不见得绝对正确，但从统计数字来看，这个判断的机制正确的机会相当高。事实上，动态编译器会根据「历史资料」做决策，所以程式执行的时间愈长，判断正确的机率就愈高。以整个结果来看，动态编译器产生的程式码执行的速度超越以前的 JIT 技术，平均速度可提高至 50%。

即时编译（Just-in-time Compilation，JIT），又称动态转译（Dynamic Translation），是一种通过在运行时将字节码翻译为机器码，从而改善字节码编译语言性能的技术。即时编译前期的两个运行时理论是字节码编译和动态编译。

在编译为字节码的系统如 Limb 编程语言，Smalltalk，UCSD P-System, Perl，GNU CLISP，和 Java 的早期版本中， 源代码被翻译为一种中间表示即字节码。 字节码不是任何特定计算机的机器码， 它可以在多种计算机体系中移植。 字节码被解释着运行在虚拟机里。

动态编译环境是一种在执行时使用编译器的编译环境。 例如， 多数 Common Lisp 系统有一个编译函数，他可以编译在运行时创建的函数。

在即时编译环境下， 字节码的编译是第一步， 它将源代码递归到可移植和可优化的中间表示。字节码被部署到目标系统。 当执行代码时，运行时环境的编译器将字节码翻译为本地机器码。 基于每个文件或每个函数：函数仅仅在他们要被执行时才会被编译。

目标是要组合利用本地和字节码编译的多种优势：多数重量级的任务如源代码解析和基本性能的优化在编译时处理，将字节码编译为机器码比起从源代码编译为机器码要快得多。部署字节码是可移植的，而机器码只限于特定的系统结构。从字节码到机器码编译器的实现更容易，因为大部分工作已经在实现字节码编译器时完成。

好，说白了，JIT 在 Android 方面其实现实际上是在 Dalvik vm 这一层，与内核是分离的，直观的表现就是几个库文件。

然后，JIT 到底能提升机器多大的性能？我也不知道——不要跟我说 Linpack 跑多少分，学计算机的都知道，Linpack 只是一个基准测试程序，测试的是机器的浮点计算能力、向量性能和高速缓存性能。Linpack 只是众多基准测试的一种，它的分数代表不了任何东西。它只有一个局部的计算性能参考

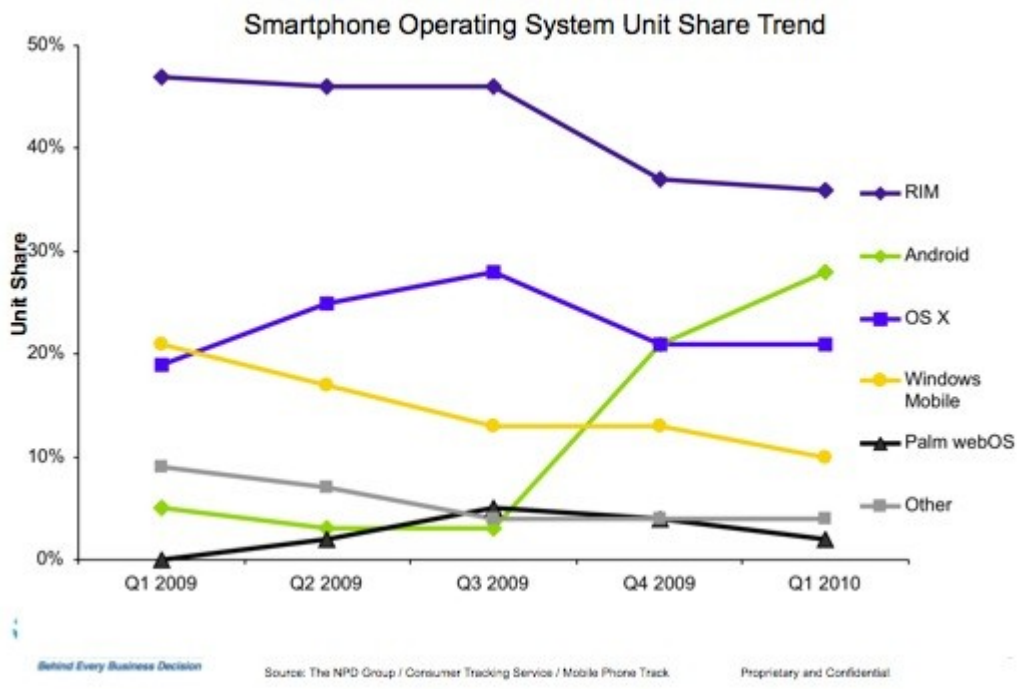
然后，为什么使用了 JIT 之后，Linpack 的分数就能提高很多？OK，看 Linpack 是干什么的——有一项是测试高速缓存的，那么，使用 JIT 之后，一部分 Dalvik 的字节码（apk 程序包的逻辑部分，如*.dex 和*.odex）会被转译成手机 CPU 的指令代码，那好，这部分代码就可以进行重用，减少了编译的时间，这就相当于一个“缓存”，使得访问 Dalvik 字节码的次数大大降低，也就是减少了解释执行字节码的次数。要知道，解释执行字节码的速度是比较慢的，所以直接体现就是 Linpack 分数的提高。

那么，JIT 对我们使用的影响大么？

准确的说，有，但不是很大，根本达不到 Froyo 2.2 宣传的那样（我恨 IT 媒体）几倍的提升。为什么？因为我们用的系统不光有逻辑（程序执行），还有 UI（图形渲染），以及其他很多因素（线程调度、I/O 等等），单独的 JIT 技术只是提高了程序代码的执行效率（还不是全部），并未对系统有全局优化的功效。如果可以的话，可以针对 Linpack 专门设计一个优化的虚拟机，让 G2 这样的机器跑到 30 分也是没问题的。但是实际使用的体验不会好很多。

【android 行业研究报告】

2.1 据 NPD 统计 Android 美国本土份额在 2010 一季度超越 iPhone OS



尽管全球还在为争抢 iPad 发疯，但是 Apple 在美国本土智能手机的市场份额却悄悄被 Android 超越。来自 NPD 对美国本土智能手机市场的统计，Android 在 2010 年第一季度以 28%首次超越了 iPhone OS 的 21%，落后于 RIM 黑莓的 36%成为全美第二大受欢迎的智能手机操作系统。

NPD 说，在过去，运行商的分销和促销决定着智能手机的市场份额，为了击败 iPhone，Verizon 扩展他们以前针对黑莓“买一送一活动”到所有 Verizon 销售的智能手机之上。Droid、Droid Eris 和黑莓 Curve 通过这种促销保证 Verizon 相对 AT&T 在 2010 第一季度强劲的销售势头。根据 NPD 的移动电话追踪系统来看，AT&T 销售的智能手机大概占到整个智能手机市场的 32%，之后是 Verizon 的 30%、T-Mobile 的 17%和 Sprint 的 15%。

文章来源：http://android.google.org.cn/posts/android-surpasses-the-iphone-in-the-united-states.html

2.2 ComScore：Android 手机在美市场份额上升至 13%

Top Smartphone Platforms 3 Month Avg. Ending May 2010 vs. 3 Month Avg. Ending Feb. 2010 Total U.S. Age 13+ Source: comScore MobiLens			
	Share (%) of Smartphone Subscribers		
	Feb-10	May-10	Point Change
Total Smartphone Subscribers	*100.0%*	*100.0%*	*N/A*
RIM	42.1%	41.7%	-0.4
Apple*	25.4%	24.4%	-1.0
Microsoft	15.1%	13.2%	-1.9
Google	9.0%	13.0%	4.0
Palm	5.4%	4.8%	-0.6

北京时间 7 月 9 日消息，据国外媒体报道，ComScore 刚发布的 3 月至 5 月智能手机市场份额数据显示，不出所料，依然持续了之前几个月的趋势。5 月 Android 手机的市场份额增长显著，上升了 4.0 个百分点至 13%。

尽管 Android 在快速增长，但 RIM 和苹果依然垄断着智能手机市场。RIM 在美国智能手机市场的份额为 41.7%，其次是苹果为 24.4%。前五名中还有微软（13.2%）和 Palm（4.8%）。3-5 月美国共有 4910 万智能手机用户，比 2 月结束的三个月里增长 8.1%。

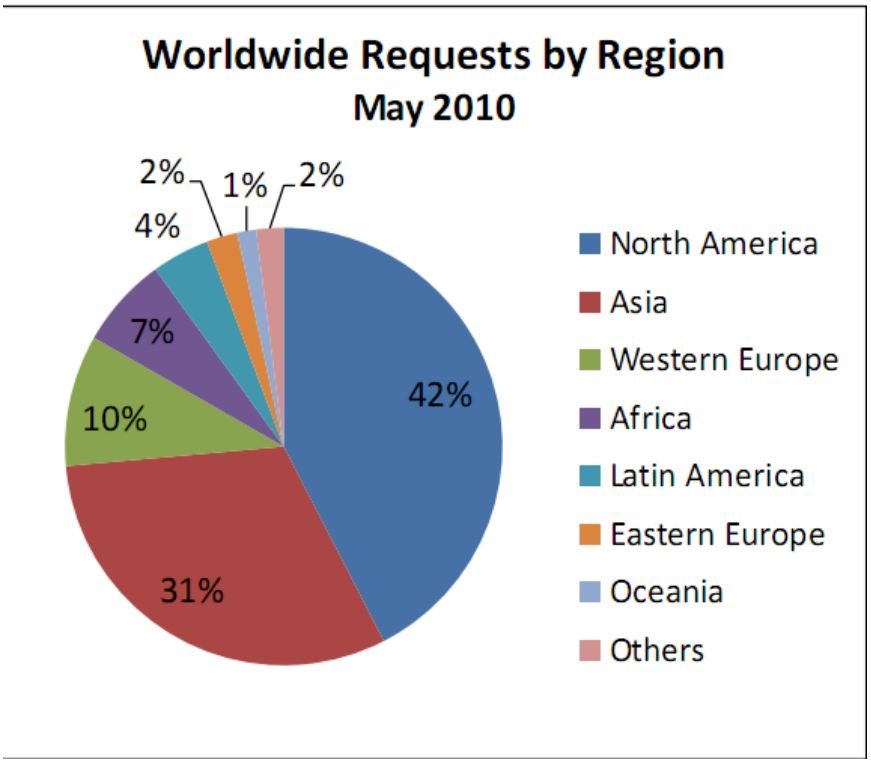
在制造商方面，三星是最大的智能手机制造商，市场占有率为 22.4%。LG 电子排名第二，市场份额为 21.5%。随后是摩托罗拉（21.2%）、RIM（8.7%，上升 0.5 个百分点）和诺基亚（8.1%）。

comScore 报告称，3 月至 5 月 13 岁以上的美国移动设备用户达到 2.34 亿。comScore 的数字显示，美国手机用户与手机的互动更为频繁。5 月美国移动用户使用短信的比例占到 65.2%，比之前 3 个月上升 1.4 个百分点；移动上网的比例也提高到 31.9%（上升 2.3 个百分点）。下载应用软件的移动用户占 30%，提高 2.1 个百分点。访问社交网站和博客的用户比例也出现增长，提高 2.6 个百分点至 20.8%。

虽然 Android 智能手机是唯一市场份额上升的手机，不过该平台的用户群仍然落后于苹果。Android 在市场份额上与苹果竞争还有段路要走，但令人鼓舞的是，该平台的用户群正在慢慢扩大。

文章来源：http://finance.qq.com/a/20100709/000630.htm

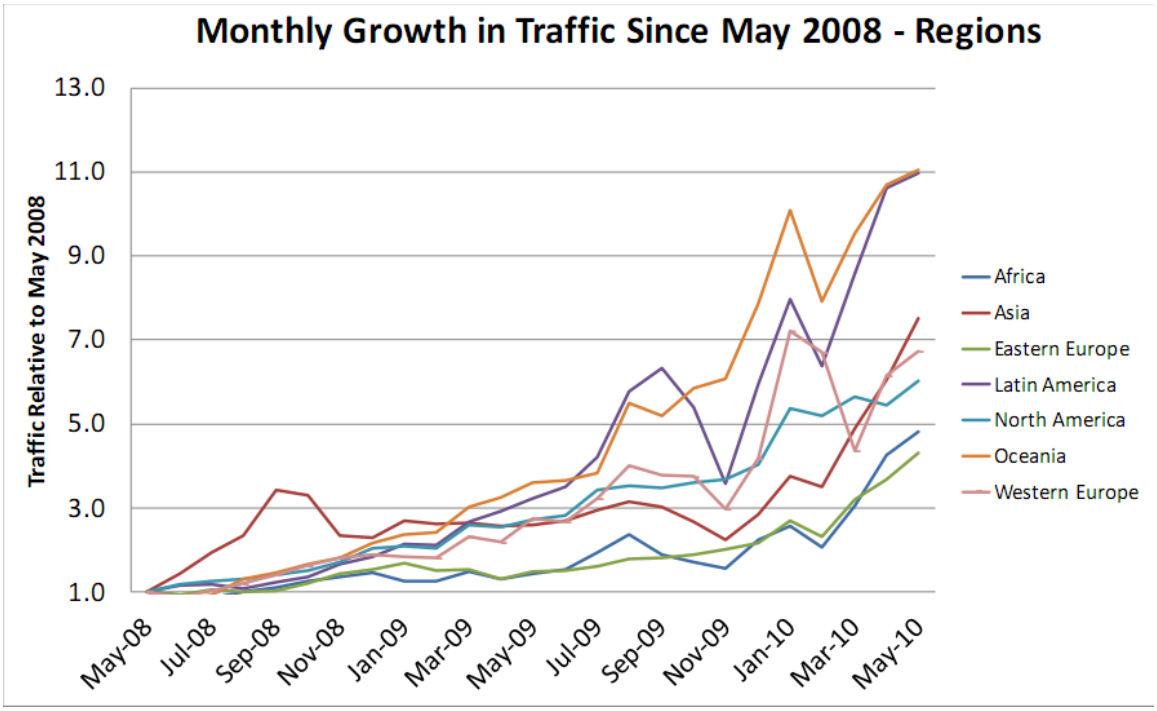
2.3 admob 5 月统计数据



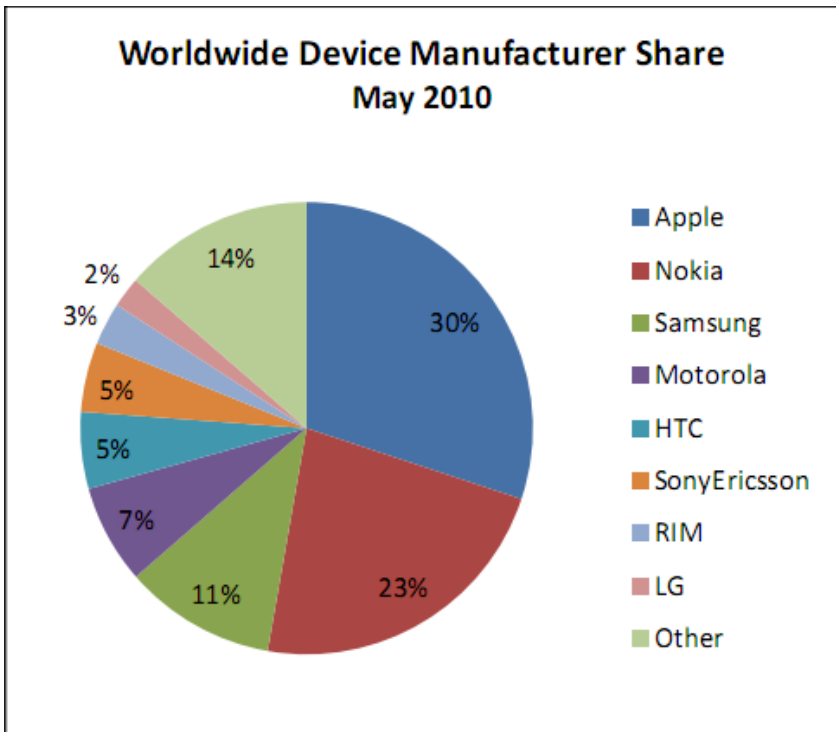
全球移动互联网使用率

Top Countries by Ad Requests
May 2010

Country	% of Requests
United States	40.3%
India	9.9%
Indonesia	5.3%
United Kingdom	3.0%
Canada	2.2%
Japan	2.1%
France	1.9%
China	1.9%
Mexico	1.7%
Vietnam	1.6%
Other Countries	30.1%
Total	**100.0%**



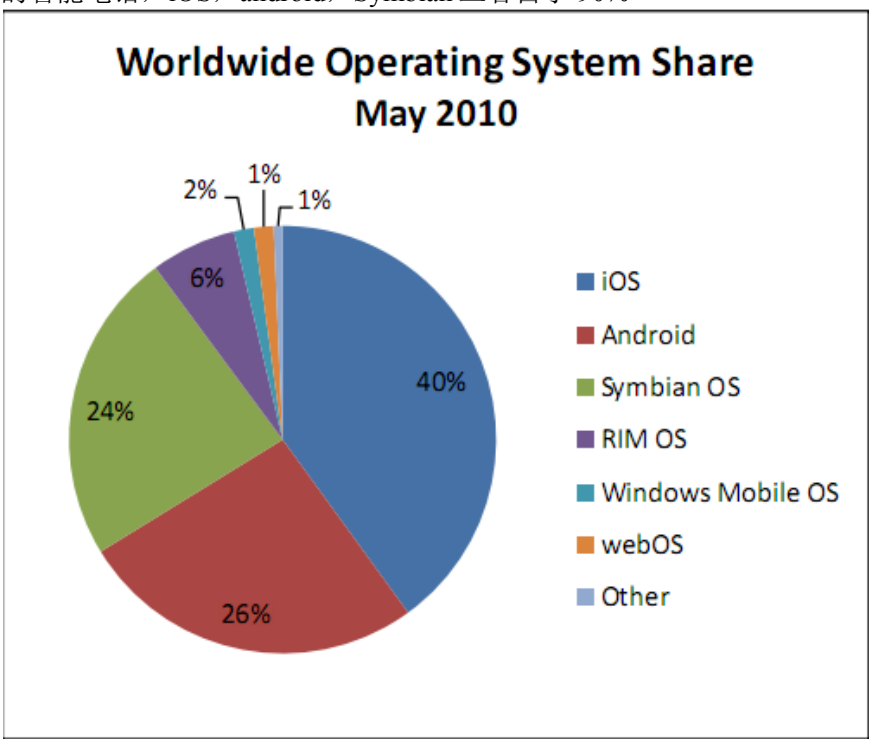
Nokia 在非洲，亚洲，东欧是手机设备制造商的领导者，Apple 在北美，大洋洲，西欧是领导者。

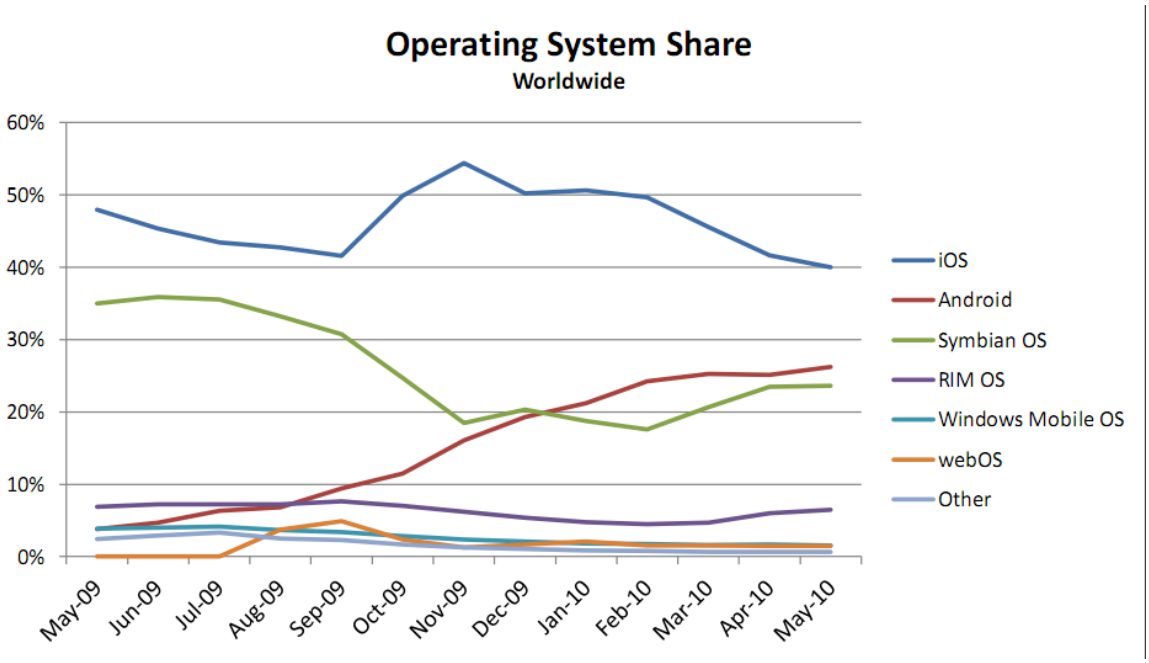


Top Devices, May 2010

Brand	Model	% of Requests
Apple	iPhone	18.8%
Apple	iPod touch	11.0%
Motorola	Droid	3.2%
Samsung	SCH R350	1.6%
Nokia	5130	1.5%
HTC	Magic	1.4%
Nokia	3110c	1.4%
Nokia	N70	1.2%
HTC	Hero	1.1%
Nokia	6300	1.1%
Total		**42.2%**

世界上所有的智能电话，iOS，android，Symbian 三者占了 90%





【android 技术盛宴】

3.1 滑动式抽屉(SlidingDrawer)的使用

SlidingDrawer 是自 SDK 1.5 才新加入的成员，也许你已曾经在 Android 手机上看过。按下一个按钮，就能展开一个"程序集"菜单，里面包含了各式各样的程序，而 SlidingDrawer Widget 正是为了这样的效果所准备的，当你在布局有限的 UI Layout 时，可以应用 SlidingDrawaer 来在可视范围内放置更多组件，在需要的时候才拉出"抽屉"里的"子功能图标"。SlidingDrawer 配置上采用了水平展开或垂直展开两种（android:orientation）方式，在 XML 里必须指定其使用的 android:handle 与 android:content，前者委托要展开的图片（Layout 配置），后者则是要展开的 Layout Content。

运行结果如下

SlidingDrawer 的使用，并不需复杂的设置才能启用，关键在于 XML 里的属性设置，稍后将会看到 XML 里的描述。在主程序中所建立的 SlidingDrawer 对象，为了捕捉"打开完毕"与"已经关闭"这两个事件，所设置的 Listener 为 SlidingDrawer.setOnDrawerOpenListener()与 SlidingDrawer.setOnDrawerCloseListener()。

```
1.    /* import 程序略 */
2.    import android.widget.GridView;
3.    import android.widget.ImageView;
```

```
4.   import android.widget.SlidingDrawer;
5.
6.   public class EX04_27 extends Activity
7.   {
8.     private GridView gv;
9.     private SlidingDrawer sd;
10.    private ImageView im;
11.    private int[] icons={R.drawable.alarm,R.drawable.calendar,
12.                         R.drawable.camera,R.drawable.clock,
13.                         R.drawable.music,R.drawable.tv};
14.    private String[] items=
15.    {
16.      "Alarm","Calendar","Camera","Clock","Music","TV"
17.    };
18.
19.    /** Called when the activity is first created. */
20.    @Override
21.    public void onCreate(Bundle savedInstanceState)
22.    {
23.      super.onCreate(savedInstanceState);
24.      /* 加载 main.xml Layout */
25.      setContentView(R.layout.main);
26.      /* 初始化对象 */
27.      gv = (GridView)findViewById(R.id.myContent1);
28.      sd = (SlidingDrawer)findViewById(R.id.drawer1);
29.      im=(ImageView)findViewById(R.id.myImage1);
30.
31.      /* 使用自定义的 MyGridViewAdapter 设置 GridView 里面的 item 内容 */
32.      MyGridViewAdapter adapter=new MyGridViewAdapter(this,items,icons);

33.      gv.setAdapter(adapter);
34.
35.      /* 设置 SlidingDrawer 被打开的事件处理 */
36.      sd.setOnDrawerOpenListener
37.      (new SlidingDrawer.OnDrawerOpenListener()
```

```
38.       {
39.         @Override
40.         public void onDrawerOpened()
41.         {
42.           im.setImageResource(R.drawable.close);
43.         }
44.       });
45.       /* 设置 SlidingDrawer 被关闭的事件处理 */
46.       sd.setOnDrawerCloseListener
47.       (new SlidingDrawer.OnDrawerCloseListener()
48.       {
49.         @Override
50.         public void onDrawerClosed()
51.         {
52.           im.setImageResource(R.drawable.open);
53.         }
54.       });
55.     }
56.   }
```

MyGridViewAdapter 在本范例中，是为了"拉开 SlindingDrawer"所要显示的 GridView 配置的图标，以下是自定义继承自 BaseAdapter 的类。

```
1.    /* import 程序略 */
2.
3.    /* 自定义 Adapter，继承 BaseAdapter */
4.    public class MyGridViewAdapter extends BaseAdapter
5.    {
6.      private Context _con;
7.      private String[] _items;
8.      private int[] _icons;
9.      /* 构造函数 */
10.     public MyGridViewAdapter(Context con,String[] items,int[] icons)
11.     {
12.       _con=con;
13.       items=items;
14.       _icons=icons;
```

```
15.     }
16.
17.     @Override
18.     public int getCount()
19.     {
20.       return _items.length;
21.     }
22.
23.     @Override
24.     public Object getItem(int arg0)
25.     {
26.       return _items[arg0];
27.     }
28.
29.     @Override
30.     public long getItemId(int position)
31.     {
32.       return position;
33.     }
34.
35.     @Override
36.     public View getView(int position, View convertView, ViewGroup parent)
37.     {
38.       LayoutInflater factory = LayoutInflater.from(_con);
39.       /* 使用 grid.xml 为每一个 item 的 Layout */
40.       View v = (View) factory.inflate(R.layout.grid, null);
41.       /* 取得 View */
42.       ImageView iv = (ImageView) v.findViewById(R.id.icon);
43.       TextView tv = (TextView) v.findViewById(R.id.text);
44.       /* 设置显示的 Image 与文字 */
45.       iv.setImageResource(_icons[position]);
46.       tv.setText(_items[position]);
47.       return v;
48.     }
49.   }
```

res/layout/main.java

这支 XML 中的 SlidingDrawer TAG，通过 Activity 里 onCreate() 方法中的 setContent-View(R.layout.main)之后，就会存在于布局（Layout）当中，即使主程序（EX04_27.java）没有编写相关的程序，依然可以正常运行，关键在于当中 android:handle 指定要显示的 ImageView（小圆图）作为开关，android:content 则是按下这个 ImageView（开关）之后，所要展开抽屉显示的布局（Layout）。

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <RelativeLayout
3.    xmlns:android="http://schemas.android.com/apk/res/android"
4.    android:layout_width="fill_parent"
5.    android:layout_height="fill_parent"
6.  >
7.    <TextView
8.      android:layout_width="fill_parent"
9.      android:layout_height="wrap_content"
10.     android:text="@string/hello"
11.     android:textSize="16sp"
12.   />
13.   <SlidingDrawer
14.     android:id="@+id/drawer1"
15.     android:layout_width="fill_parent"
16.     android:layout_height="fill_parent"
17.     android:handle="@+id/layout1"
18.     android:content="@+id/myContent1"
19.     android:orientation="horizontal"
20.   >
21.     <LinearLayout
22.       android:id="@id/layout1"
23.       android:layout_width="35px"
24.       android:layout_height="fill_parent"
25.       android:background="@drawable/black"
26.       android:gravity="center"
27.     >
28.       <ImageView
29.         android:id="@+id/myImage1"
30.         android:layout_width="wrap_content"
```

```
31.           android:layout_height="wrap_content"
32.           android:src="@drawable/open"
33.         />
34.       </LinearLayout>
35.       <GridView
36.         android:id="@id/myContent1"
37.         android:layout_width="wrap_content"
38.         android:layout_height="wrap_content"
39.         android:numColumns="2"
40.         android:background="@drawable/black"
41.         android:gravity="center"
42.       />
43.     </SlidingDrawer>
44.   </RelativeLayout>
```

扩展学习

上面的 XML 程序代码之中是配置以水平方式来打开抽屉，我们另外通过修改 android:orientation="vertical"，就能让 SlidingDrawer 以垂直的方式打开。

```
1.   <SlidingDrawer
2.     android:id="@+id/drawer1"
3.     android:layout_width="fill_parent"
4.     android:layout_height="fill_parent"
5.     android:handle="@+id/layout1"
6.     android:content="@+id/myContent1"
7.     android:orientation="vertical"
8.   >
```

运行结果如图

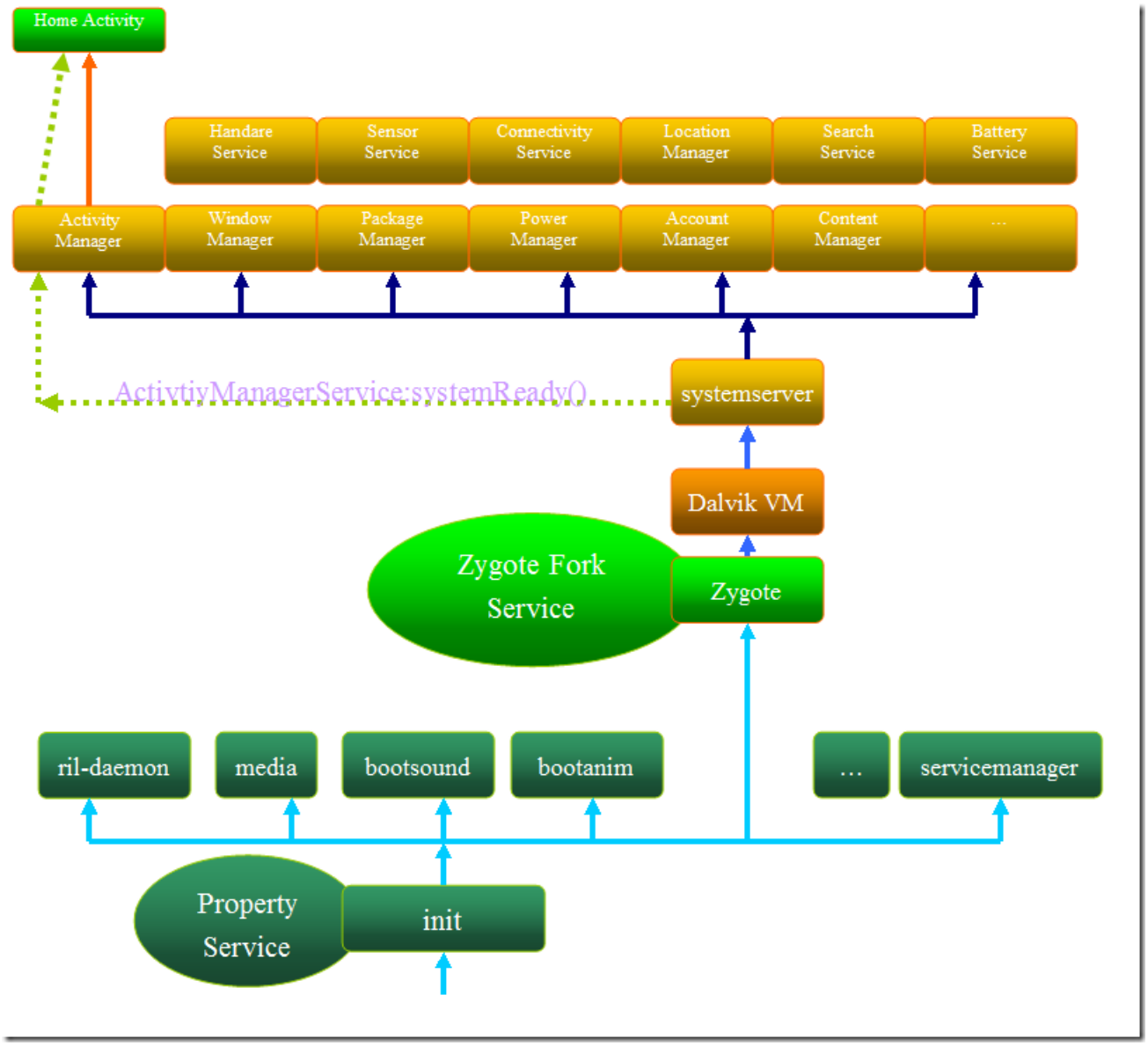
文章来源：http://book.51cto.com/art/201007/212101.htm

3.2 Android 启动过程详解

Android 从 Linux 系统启动有 4 个步骤；

(1) init 进程启动
(2) Native 服务启动
(3) System Server，Android 服务启动
(4) Home 启动

总体启动框架图如：

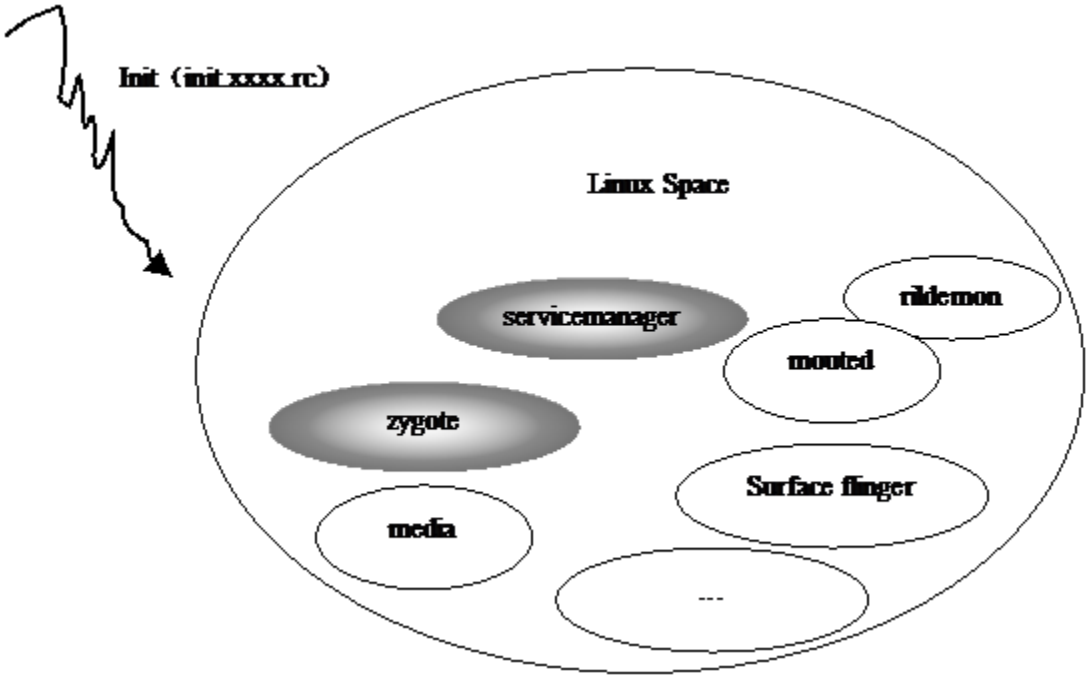


第一步：initial 进程(system\core\init)

init 进程，它是一个由内核启动的用户级进程。内核自行启动（已经被载入内存，开始运行，并已初始化所有的设备驱动程序和数据结构等）之后，就通过启动一个用户级程序 init 的方式，完成引导进程。init 始终是第一个进程.

Init.rc

Init.marvell.rc



Init 进程一起来就根据 init.rc 和 init.xxx.rc 脚本文件建立了几个基本的服务：

servicemanamger
zygote
。。。

最后 Init 并不退出，而是担当起 property service 的功能。

1.1 脚本文件
init@System/Core/Init

Init.c： parse_config_file(Init.rc)

@parse_config_file(Init.marvel.rc)

解析脚本文件：Init.rc 和 Init.xxxx.rc(硬件平台相关)

Init.rc 是 Android 自己规定的初始化脚本(Android Init Language, System/Core/Init/readme.txt)

该脚本包含四个类型的声明：

Actions
Commands
Services
Options.
1.2 服务启动机制

我们来看看 Init 是这样解析.rc 文件开启服务的。

（1）打开.rc 文件，解析文件内容@ system\core\init\init.c
将 service 信息放置到 service_list 中。@ system\core\init parser.c
（2）restart_service()@ system\core\init\init.c

service_start

execve(…).建立 service 进程。

第二步 Zygote
Servicemanager 和 zygote 进程就奠定了 Android 的基础。Zygote 这个进程起来才会建立起真正的 Android 运行空间，初始化建立的 Service 都是 Navtive service.在.rc 脚本文件中 zygote 的描述：

service zygote /system/bin/app_process -Xzygote /system/bin --zygote --start-system-server

所以 Zygote 从 main(…)@frameworks\base\cmds\app_main.cpp 开始。

(1) main(…)@frameworks\base\cmds\app_main.cpp

建立 Java Runtime
runtime.start("com.android.internal.os.ZygoteInit", startSystemServer);

(2) runtime.start@AndroidRuntime.cpp

建立虚拟机
运行：com.android.internal.os.ZygoteInit：main 函数。

（3）main()@com.android.internal.os.ZygoteInit//正真的 Zygote。

registerZygoteSocket();//登记 Listen 端口
startSystemServer();
进入 Zygote 服务框架。
经过这几个步骤，Zygote 就建立好了，利用 Socket 通讯，接收 ActivityManangerService 的请求，Fork 应用程序。

第三步 System Server

startSystemServer@com.android.internal.os.ZygoteInit 在 Zygote 上 fork 了一个进程: com.android.server.SystemServer.于是 SystemServer@(SystemServer.java）就建立了。Android 的所有服务循环框架都是建立 SystemServer@(SystemServer.java）上。在 SystemServer.java 中看不到循环结构，只是可以看到建立了 init2 的实现函数，建立了一大堆服务，并 AddService 到 service Manager。

main() @ com/android/server/SystemServer

```
{
        init1();
}
```

Init1()是在 Native 空间实现的（com_andoird_server_systemServer.cpp）。我们一看这个函数就知道了，init1->system_init() @System_init.cpp

在 system_init()我们看到了循环闭合管理框架。

```
{
    Call "com/android/server/SystemServer", "init2"

    ……
    ProcessState::self()->startThreadPool();

      IPCThreadState::self()->joinThreadPool();
}
```

init2()@SystemServer.java 中建立了 Android 中所有要用到的服务。

这个 init2（）建立了一个线程，来 New Service 和 AddService 来建立服务

第三步 Home 启动
在 ServerThread@SystemServer.java 后半段，我们可以看到系统在启动完所有的 Android 服务后，做了这样一些动作：

（1） 使用 xxx.systemReady()通知各个服务，系统已经就绪。

(2) 特别对于 ActivityManagerService.systemReady(回调)

Widget.wallpaper,imm(输入法)等 ready 通知。

Home 就是在 ActivityManagerService.systemReady()通知的过程中建立的。下面是 ActivityManagerService.systemReady()的伪代码：

systemReady()@ActivityManagerService.java

resumeTopActivityLocked()

startHomeActivityLocked();//如果是第一个则启动 HomeActivity。
startActivityLocked（。。。）CATEGORY_HOME

文章来源：http://blog.csdn.net/maxleng/archive/2010/04/20/5508372.aspx

3.3 Android 消息系统

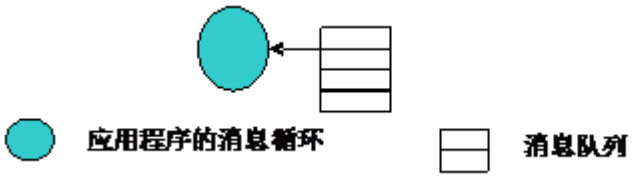
我们要理解 Android 的消息系统，Looper，Handle，View 等概念还是需要从消息系统的基本原理及其构造这个源头开始。从这个源头，我们才能很清楚的看到 Android 设计者设计消息系统之意图及其设计的技术路线。

1．消息系统的基本原理

从一般的系统设计来讲，一个消息循环系统的建立需要有以下几个要素：

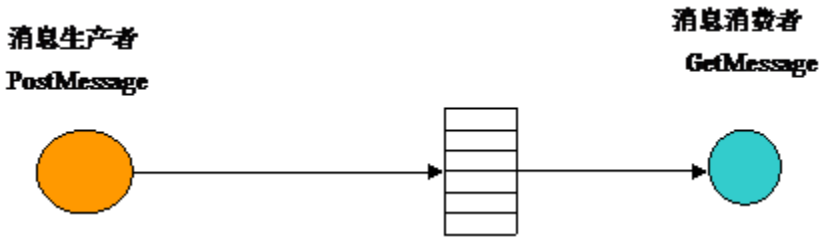
- 消息队列
- 发送消息
- 消息读取
- 消息分发
- 消息循环线程

首先来研究一下消息驱动的基本模型，我使用如下的图形来表示一个消息系统最基本构成：

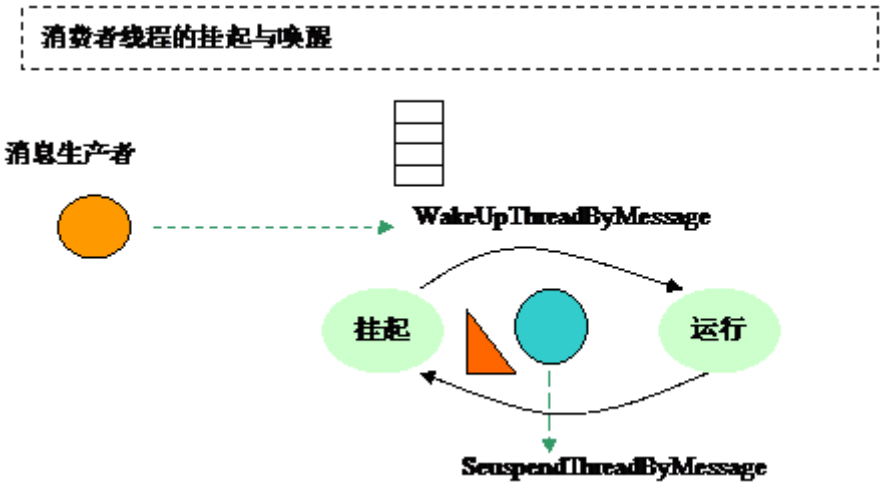


上面的模型代表应用程序一直查询自己的消息队列,如果有有消息进来,应用消息处理函数中根据消息类型及其参数来作相应的处理。

消息系统要运作起来,必定有消息的产生和消费。我们可以从下图看到消息生产和消费的一个基本的链条，这是一个最基本的，最简单的消息系统。



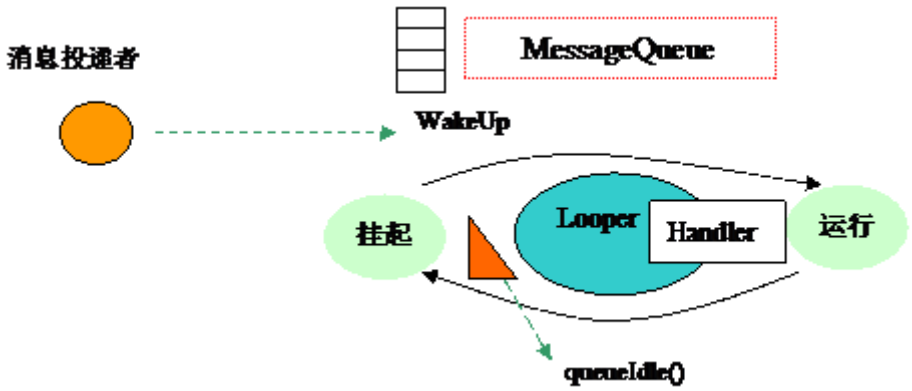
生产线程将消息发送到消息队列，消息消费者线程从消息队列取出消息进行相应的处理。但是这样简单的模型对实际运行的系统来说是不够的，例如对系统资源的消耗等不能很好的处理，我们就需要一个有旗语的消息系统模型，在上面的消息系统模型中加入了一个旗语，让消息消费者线程在没有消息队列为空时，等待旗语，进入到挂起状态，而有消息到达时，才被唤醒继续运行。当然生产者同时也可以是消费者。



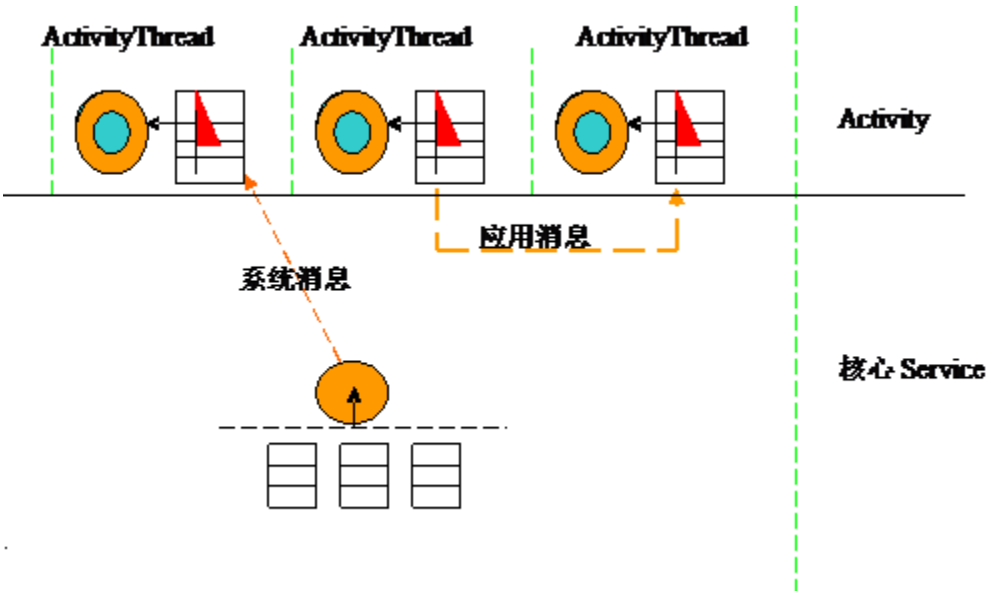
2. Android 的消息模型

Android 要建立一个消息系统使用了 Looper，MessageQueue,Handler 等概念，从上节的原理我们可以知道这些都是概念包装,本质的东西就是消息队列中消息的分发路径的和消息分发处理方式的设计。Android 巧妙的利用了对象抽象技术抽象出了 Looper 和 Handler 的概念。在 Looper 和 Handler 两个概念的基础上,通过 View 的处理函数框架，Android 十分完美的达到消息分发的目的。

参照基本消息系统描述模型，我给出了 Android 消息系统整体框架,表示如下：

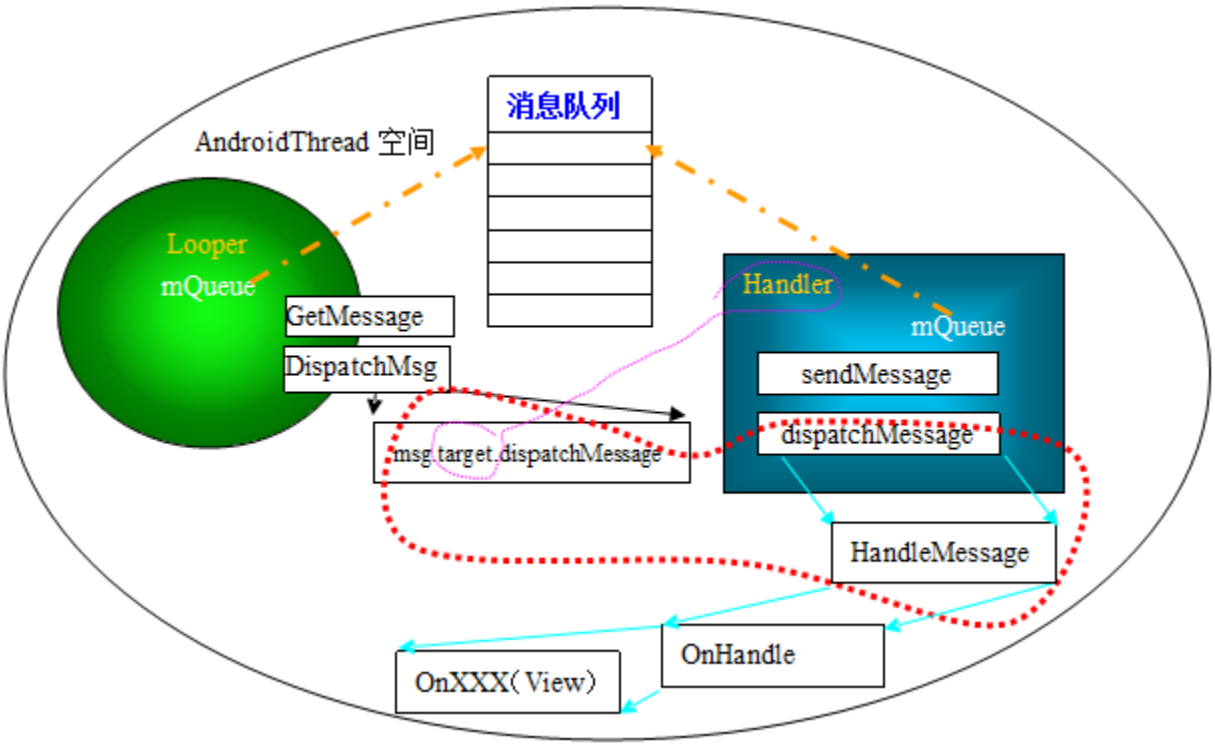


Android 消息系统消息分发框架



2 、Looper,Handler 详解

Looper 只是产生一个消息循环框架，首先 Looper 创建了消息队列并把它挂接在 Linux 的线程上下文中，进入到取消息，并分发消息的循环当中。Handler 对象在同一个线程上下文中取得消息队列，对消息队列进行封装操作，最主要的就是 SendMessage 和担当起 dispatchMessage 这个实际工作。外部系统需要向某个 Android 线程发送消息，必须通过属于该 AndroidThread 的 Handler 这个对象进行。



Handler 属于某个线程，取决 Handlerd 对象在哪个线程中建立。Handler 在构建时做了如下的默认动作：

- 从线程上下文取得 Looper。
- 通过 Looper 获取到消息队列并记录在自己的成员 mQueue 变量中

Handler 使用消息队列进行对象封装，提供如下的成员函数：

- 通过 post(Runnable r)发送。Runnable 是消息处理的回调函数，通过该消息的发送，引起 Runable 的回调运行，Ｐｏｓｔ消息放置消息队列的前面。Message.callback=Runable.
- 通过 sendMessage 发送。放置在所有的 Post 消息之后，sendMessage 发送消息.
- dispatchMessage 分发消息。消息带有回调函数，则执行消息回调函数，如何没有则使用默认处理函数：handleMessage。而 handleMessage 往往被重载成某个继承 Handler 对象的新的特定的 handleMessage。

几乎所有的 Message 发送时，都指定了 target。Message.target=(this).

Looper 运行在 Activity 何处？我们现在可以从代码堆栈中纵观一下 Looper 的位置。

NaiveStart.main()

ZygoteInit.main

ZygoteInit$MethodAndArgsCall.run

Method.Invoke

method.invokeNative

ActivityThread.main()

Looper.loop()

ViewRoot$RootHandler().dispatch()

handleMessage

....

这样我们就更清楚的了解到 Looper 的运行位置。

文章来源：http://blog.csdn.net/maxleng/archive/2010/05/03/5552976.aspx

【其他】

4.1 BUG 提交

如果你发现文档中翻译不妥的地方，请到如下地址反馈，我们会定期更新、发布更新后的版本
http://www.eoeandroid.com/thread-26982-1-1.html

4.2 关于 eoeandroid

eoeandroid 团队是一个有远大的梦想，有充沛的激情，有高效执行力的团队。北京易联致远无线技术有限公司于 2009 年 8 月成立。eoeandroid 的核心成员有在摩托罗拉、卓望科技、T3g 和手机 design house 的工作经历和相关丰富的行业经验。eoeandroid 致力于让移动互联网的软件开发变得容易，发布变得更加方便，传播变得更加迅捷,让用户以最快的速度获取最适合自己的移动互联网应用。

4.3 新版优亿市场上线

优亿市场（eoeMarket）经过 eoe 团队无数个白天黑夜的奋斗，数个版本的调整和升级，我们迎来了一个更加优优秀的新版本（V0.2.2 内测版）；其本土化、纯中文支持；高质量、分类清晰的应用，无论你是狂热的 android 玩家，还是初涉 android 江湖的小白，你都可以在上面轻而易举的找到自己想要的软件和游戏。而这一切都是免费的。
新版优亿市场（eoeMarket） V0.2.2 内测版特性如下：

- 全新设计的炫酷界面
- 智能检测已安装应用的新版本
- 更加精准的智能推荐
- 更加丰富的应用专题
- web 端和客户端收藏列表一键同步
- 前所未有的稳定性

使用方法：

第一步:准备工作

1. 手机在主屏幕状态 按下“MENU”键，选择“设置”菜单
2. 在设置界面中选择“应用程序”
3. 在应用程序设置界面勾选“未知来源”选项，允许安装非电子市场提供的应用程序

第二步:下载并安装 eoe 应用商店 2

a.通过手机浏览器安装

1.输入网址 eoemarket.com/a，按下“转到”按钮

2.下载完成后，点击“eoeMarket2.apk”,启动安装程序

b.通过 PC 安装

1.浏览器输入 eoemarket.com/a 或者点击本页的'立即下载'

2.下载完成后，将“eoeMarket2.apk”拷贝到手机 SD 卡中

3.安装