Android 入门 FAQ

目　录

1　序言.....3
1.1　文档简介.....3
1.2　编写目的.....3
1.3　阅读对象.....3
2　Android 新手入门 FAQ.....3
2.1　Q：什么是 Android?.....3
2.2　Q：Android 就业前景?.....3
2.3　Q：Android 的特点都有哪些？.....4
2.4　Q：Android 平台手机优势？.....5
2.5　Q：学习 android 需要哪些基础？.....6
2.6　Q：Android 上编程用什么语言?.....6
2.7　Q：Android 编程环境需要哪些?.....7
2.8　Q：什么是 APK？.....7
2.9　Q：什么是 SDK？.....7
2.10　Q：什么是 API？.....7
2.11　Q：什么是 TCP 协议和 UDP 协议？.....8
2.12　Q：Android 开发者应该先看什么文档?.....8
2. 13　Q：Android 系统架构有哪些？.....8
2.14　Q：什么是开源？.....10
2.15　Q：如何搭建编程环境？.....11
2.16　Q：什么是 NDK？.....12
2.17　Q：怎样提高 Android 应用程序的速度？.....12
2.18　Q：Android 运行库有哪些？.....13
2.19　Q：Android 都支持哪些 Java 特性.....13
2.20　Q：Android 最简单播放 GIF 动画方法是什么？.....14
2.21　Q：线程在 Android 应用当中的作用？.....14
2.22　Q：如何成为高手？.....14
2.23　Q：JNI 是什么？.....15
2.24　Q：Android 和 Linux 的区别？.....15
2.25　Q：SDK 升级了，如何更新 SDK？.....16
2.26　Q：如何进入 Recovery 模式.....16
2.27　Q：Android bionic 移植需要注意的事项.....16
2.28　Q：Android 源码下 vendor 目录下文件的作用？.....18
2.29　Q：编译 Android 源码和编译 Android 内核有什么区别？.....18
2.30　Q：如何获取手机和存储卡上的图片？.....18
2.31　Q：如何使用双缓冲？.....19

Android 入门 FAQ

1 序言

1.1 文档简介

该文档介绍了 Android 新手入门需要知道的一些基础知识。

1.2 编写目的

该文档用于 xxx 框架-应用加速器(Android)开发人员，在开发之前需要了解和准备的一些基础知识。

1.3 阅读对象

本文档阐述的内容是 Android 入门相关问题的一个解答和基础知识普及，凡是想了解和应用 Android 平台开发的开发人员和感兴趣的朋友。

2 Android 新手入门 FAQ

2.1 Q：什么是 Android?

A：Android 一词的本义指“机器人”，同时也是 Google 于 2007 年 11 月 5 日宣布的基于 Linux 平台的开源手机操作系统的名称，该平台由操作系统、中间件、用户界面和应用软件组成，号称是首个为移动终端打造的真正开放和完整的移动软件。简单来说是个开源的手机操纵系统。

2.2 Q：Android 就业前景?

A：Android 是 Google 开发的基于 Linux 平台的开源移动操作系统。它包括操作系统、用户界面和应用程序----移动电话工作所需的全部软件，而且不存在任何以往阻碍移动产业创新的专有权障碍，号称是首个为移动终端打造的真正开发和完整的移动软件。

国外 Android 市场正在如日中天的扩展，据市场调研机构最近发布的一份报告称，今年第一季度基于 Android 操作系统的智能手机在美国智能手机总销量中所占比例达到 28%，首度超过苹果 iphone，苹果 iphone 约为 21%，相信在不久的将来会有更多的用户选择 Android 系统的手机或是无线终端设备。

2.3 Q：Android 的特点都有哪些？

1）应用程序框架：支持组件的复用和更换

2）Dalvik 虚拟机：专门为移动设备进行过优化

3）集成的浏览器：基于开源的 WebKit 引擎 ，TV 上会内置 Chrome 浏览器

4）优化的图形机制：自定义的 2D 图形库，基于 OpenGL ES 1.0 规范的 3D 图形实现（本项为硬件加速器可选）

5）SQLite：轻量级的数据库，支持结构化数据的存储

6）媒体支持：面向常见的音频、视频以及静态图形档案格式 (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF)

7）GSM 技术 GSM：global system for mobile communications (依赖硬件支持)

8）Bluetooth, EDGE, 3G, and WiFi (依赖硬件支持)

9）Camera, GPS, compass, and accelerometer (依赖硬件支持)

10）Rich development environment：丰富的开发环境，包含一套硬件仿真器，一些用于程序调试、内存和性能剖析的工具，以及支持 Eclipse 集成开发环境的插件（ADT ）。

2.4 Q：Android 平台手机优势？

一、开放性

在优势方面，Android 平台首先就是其开发性，开发的平台允许任何移动终端厂商加入到 Android 联盟中来。显著的开放性可以使其拥有更多的开发者，随着用户和应用的日益丰富，一个崭新的平台也将很快走向成熟。

开发性对于 Android 的发展而言，有利于积累人气，这里的人气包括消费者和厂商，而对于消费者来讲，随大的受益正是丰富的软件资源。开放的平台也会带来更大竞争，如此一来，消费者将可以用更低的价位购得心仪的手机。

二、挣脱运营商的束缚

在过去很长的一段时间，特别是在欧美地区，手机应用往往受到运营商制约，使用什么功能接入什么网络，几乎都受到运营商的控制。从去年 iPhone 上市 ，用户可以更加方便地连接网络，运营商的制约减少。随着 EDGE、HSDPA 这些 2G 至 3G 移动网络的逐步过渡和提升，手机随意接入网络已不是运营商口中的笑谈，当你可以通过手机 IM 软件方便地进行即时聊天时，再回想不久前天价的彩信和图铃下载业务，是不是像噩梦一样？互联网巨头 Google 推动的 Android 终端天生就有网络特色，将让用户离互联网更近。

三、丰富的硬件选择

这一点还是与 Android 平台的开放性相关，由于 Android 的开放性，众多的厂商会推出千奇百怪，功能特色各具的多种产品。功能上的差异和特色，却不会影响到数据同步、甚至软件的兼容，好比你从诺基亚 Symbian 风格手机 一下改用苹果 iPhone ，同时还可将 Symbian 中优秀的软件带到 iPhone 上使用、联系人等资料更是可以方便地转移，是不是非常方便呢？

四、不受任何限制的开发商

Android 平台提供给第三方开发商一个十分宽泛、自由的环境，不会受到各种条条框框的阻扰，可想而知，会有多少新颖别致的软件会诞生。但也有其两面性，血腥、暴力、情色方面的程序和游戏如可控制正是留给 Android 难题之一。

五、无缝结合的 Google 应用

如今叱诧互联网的 Google 已经走过 10 年度历史，从搜索巨人到全面的互联网渗透，Google 服务如地图、邮件、搜索等已经成为连接用户和互联网的重

要纽带，而 Android 平台手机将无缝结合这些优秀的 Google 服务。

2.5 Q：学习 android 需要哪些基础？

A：学习 Android 一定要有 Java 基础，最差也要类似 Java 的其他面对像对象语言的基础。很多朋友从来没有搞过编程，听说 Android 能赚钱，就疯了一样冲过来要学习。这明显是不靠谱的，甚至有的朋友一个字母一个字母的按照我的例子去敲代码，然后非常努力的把代码背下来。像背英文单词一样的背代码，编程学成这个样子，还真是让小生佩服啊！不管怎样，你最少也要掌握 Java 语言的如下知识点才能够开始学 Android：

a)Java 基本数据类型及其特点

b)Java 分支语句和循环语句的使用

c)类和对象的创建和使用方法

d)函数的使用

e)抽象类和接口

f)继承和实现

g)对象的多态性

h)包和访问权限

i)异常的处理

j)类集框架

k)eclipse 的基本用法

2.6 Q：Android 上编程用什么语言?

A：Android 应用基于 Java，支持 SQL，由于底层是 Linux 所以底层支持 C/C++。

目前有两种编程:

1）基于 ADT 的 JAVA 编程

2）基于 NDK 的 C 编程

2.7 Q：Android 编程环境需要哪些?

A：编程环境：Ide 、Android SDK、JDK、Java

官方推荐用 JDK+ECLIPSE+ADT+ADK(WINDOWS 开发环境下。

2.8 Q：什么是 APK?

A：APK 是 Android Package 的缩写，即 Android 安装包(anapk)。APK 是类似 Symbian Sis 或 Sisx 的文件格式。通过将 APK 文件直接传到 Android 模拟器或 Android 手机中执行即可安装。

apk 文件和 sis 一样最终把 android sdk 编译的工程打包成一个安装程序文件格式为 apk。 APK 文件其实是 zip 格式，但后缀名被修改为 apk，通过 UnZip 解压后,可以看到 Dex 文件,Dex 是 Dalvik VM executes 的全称,即 Android Dalvik 执行程序，并非 Java ME 的字节码而是 Dalvik 字节码。

2.9 Q：什么是 SDK?

A：SDK，Software Development Kit 的缩写，中文即“软件开发工具包”。广义上指辅助开发某一类软件的相关文档、范例和工具的集合。

SDK 是一些被软件工程师用于为特定的软件包、软件框架、硬件平台、操作系统等创建应用软件的开发工具的集合，一般而言 SDK 即开发 Windows 平台下的应用程序所使用的 SDK。它可以简单的为某个程序设计语言提供应用程序接口 API 的一些文件，但也可能包括能与某种嵌入式系统通讯的复杂的硬件。一般的工具包括用于调试和其他用途的实用工具。SDK 还经常包括示例代码、支持性的技术注解或者其他的为基本参考资料澄清疑点的支持文档。

2.10 Q：什么是 API?

A：API（Application Programming Interface）其实就是操作系统留给应用程序的一个调用接口，应用程序通过调用操作系统的 API 而使操作系统去执行应用程序的命令（动作）。

其实早在 DOS 时代就有 API 的概念，只不过那个时候的 API 是以中断调用的形式（INT 21h）提供的，在 DOS 下跑的应用程序都直接或间接的通过中断

调用来使用操作系统功能，比如将 AH 置为 30h 后调用 INT 21h 就可以得到 DOS 操作系统的版本号。而在 Windows 中，系统 API 是以函数调用的方式提供的。同样是取得操作系统的版本号，在 Windows 中你所要做的就是调用 GetVersionEx() 函数。

2.11 Q：什么是 TCP 协议和 UDP 协议？

A：TCP/IP 协议的名称中只有 TCP 这个协议名，但是在 TCP/IP 的传输层同时存在 TCP 和 UDP 两个协议。Tranfer Control Protocol 的简称，是一种面向连接的保证可靠传输的协议。通过 TCP 协议传输，得到的是一个顺序的无差错的数据流。发送方和接收方的成对的两个 socket 之间必须建立连接，以便在 TCP 协议的基础上进行通信，当一个 socket（通常都是 server socket）等待建立连接时，另一个 socket 可以要求进行连接，一旦这两个 socket 连接起来，它们就可以进行双向数据传输，双方都可以进行发送或接收操作。

UDP 协议是 User Datagram Protocol 的简称，是一种无连接的协议，每个数据报都是一个独立的信息，包括完整的源地址或目的地址，它在网络上以任何可能的路径传往目的地，因此能否到达目的地，到达目的地的时间以及内容的正确性都是不能被保证的。

2.12 Q：Android 开发者应该先看什么文档?

A：对开发者来说,最重要的是概念,而 DOCS 中的 Dev Guide 里面对其解释十分清晰,请仔细阅读。并可以到公司的 SVN 中获取相关文档和学习资料。

2.13 Q：Android 系统架构有哪些？

A：Android 的系统架构和其操作系统一样，采用了分层的架构。从架构图看，android 分为四个层，从高层到低层分别是应用程序层、应用程序框架层、系统运行库层和 linux 核心层。

1） 应用程序

Android 会同一系列核心应用程序包一起发布，该应用程序包包括 email 客户端，SMS 短消息程序，日历，地图，浏览器，联系人管理程序等。所有的应用程序都是使用 JAVA 语言编写的。

2) 应用程序框架

开发人员也可以完全访问核心应用程序所使用的 API 框架。该应用程序的架构设计简化了组件的重用;任何一个应用程序都可以发布它的功能块并且任何其它的应用程序都可以使用其所发布的功能块(不过得遵循框架的安全性限制)。同样，该应用程序重用机制也使用户可以方便的替换程序组件。

隐藏在每个应用后面的是一系列的服务和系统, 其中包括;

* 丰富而又可扩展的视图(Views)，可以用来构建应用程序， 它包括列表(lists)，网格(grids)，文本框(text boxes)，按钮(buttons)，甚至可嵌入的 web 浏览器。

* 内容提供器(Content Providers)使得应用程序可以访问另一个应用程序的数据(如联系人数据库)， 或者共享它们自己的数据

* 资源管理器(Resource Manager)提供 非代码资源的访问，如本地字符串，图形，和布局文件(layout files)。

* 通知管理器 (Notification Manager) 使得应用程序可以在状态栏中显示自定义的提示信息。

* 活动管理器(Activity Manager) 用来管理应用程序生命周期并提供常用的导航回退功能。

有关更多的细节和怎样从头写一个应用程序，请参考 如何编写一个 Android 应用程序.

3) 系统运行库

程序库

Android 包含一些 C/C++库，这些库能被 Android 系统中不同的组件使用。它们通过 Android 应用程序框架为开发者提供服务。以下是一些核心库：

* 系统 C 库 - 一个从 BSD 继承来的标准 C 系统函数库(libc)，它是专门为基于 embedded linux 的设备定制的。

* 媒体库 - 基于 PacketVideo OpenCORE;该库支持多种常用的音频、视频格式回放和录制，同时支持静态图像文件。编码格式包括 MPEG4,

H.264, MP3, AAC, AMR, JPG, PNG 。

* Surface Manager - 对显示子系统的管理，并且为多个应用程序提 供了 2D 和 3D 图层的无缝融合。

* LibWebCore - 一个最新的 web 浏览器引擎用，支持 Android 浏览器和一个可嵌入的 web 视图。

* SGL - 底层的 2D 图形引擎

* 3D libraries - 基于 OpenGL ES 1.0 APIs 实现;该库可以使用硬件 3D 加速(如果可用)或者使用高度优化的 3D 软加速。

* FreeType -位图(bitmap)和矢量(vector)字体显示。

* SQLite - 一个对于所有应用程序可用，功能强劲的轻型关系型数据库引擎。

Android 运行库

Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。

每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。 Dalvik 虚拟机执行(.dex)的 Dalvik 可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的，所有的类都经由 JAVA 编译器编译,然后通过 SDK 中 的 "dx" 工具转化成.dex 格式由虚拟机执行。

Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

4) Linux 内核

Android 的核心系统服务依赖于 Linux 2.6 内核，如安全性，内存管理，进程管理， 网络协议栈和驱动模型。 Linux 内核也同时作为硬件和软件栈之间的抽象层。

2.14 Q：什么是开源？

A：开源，意为开放源代码，由 Bruce Perens（曾是 Debian 的创始人之一）定义如下：

1)自由再散布（Free Distribution）：获得源代码的人可自由再将此源代码散布。

2)源代码（Source Code）：程式的可执行档在散布时，必需随附完整源代码或是可让人方便的事后取得源代码。

3)衍生著作（Derived Works）：让人可依此源代码修改后，在依照同一授权条款的情形下再散布。

4)原创作者程式源代码的完整性（Integrity of The Author's Source Code）：意即修改后的版本，需以不同的版本号码以与原始的程式码做分别，保障原始的程式码完整性。

5)不得对任何人或团体有差别待遇（No Discrimination Against Persons or Groups）：开放源代码软件不得因性别、团体、国家、族群等设定限制，但若是因为法律规定的情形则为例外（如：美国政府限制高加密软件的出口）。

6)对程式在任何领域内的利用不得有差别待遇（No Discrimination Against Fields of Endeavor）：意即不得限制商业使用。

7)散布授权条款（Distribution of License）：若软件再散布，必需以同一条款散布之。

8)授权条款不得专属于特定产品（License Must Not Be Specific to a Product）：若多个程式组合成一套软件，则当某一开放源代码的程式单独散布时，也必需要符合开放源代码的条件。

9)授权条款不得限制其他软件（License Must Not Restrict Other Software）：当某一开放源代码软件与其他非开放源代码软件一起散布时（例如放在同一光碟片），不得限制其他软件的授权条件也要遵照开放源代码的授权。

10)授权条款必须技术中立（License Must Be Technology-Neutral）：意即授权条款不得限制为电子格式才有效，若是纸本的授权条款也应视为有效。

2.15 Q：如何搭建编程环境？

A：开发平台推荐 Eclipse。VS 基本不要想，除非你特别钟爱于传统的 vim 或其他编程环境并且打算让自己每天敲一堆命令，否则还是用 Eclipse 吧。

它是官方推荐的，目前支持的还算中规中距。而且 Linux，windows 都能用。

具体环境搭建请参看移动应用研发组编写的《开发环境搭建手册 V1.0.doc》。

2.16 Q：什么是 NDK？

A：1）NDK 是一系列工具的集合。

NDK 提供了一系列的工具，帮助开发者快速开发 C（或 C++）的动态库，并能自动将 so 和 java 应用一起打包成 apk。这些工具对开发者的帮助是巨大的。NDK 集成了交叉编译器，并提供了相应的 mk 文件隔离 CPU、平台、ABI 等差异，开发人员只需要简单修改 mk 文件（指出“哪些文件需要编译”、“编译特性要求”等），就可以创建出 so。NDK 可以自动地将 so 和 Java 应用一起打包，极大地减轻了开发人员的打包工作。

2）NDK 提供了一份稳定、功能有限的 API 头文件声明。

Google 明确声明该 API 是稳定的，在后续所有版本中都稳定支持当前发布的 API。从该版本的 NDK 中看出，这些 API 支持的功能非常有限，包含有：C 标准库（libc）、标准数学库（libm）、压缩库（libz）、Log 库（liblog）

2.17 Q：怎样提高 Android 应用程序的速度？

A：首先， 我们要先明白“加快”是有两层意思的，第一层是代码执行所需要的时间，第二层意思是用户需要等待用户界面响应的时间。下面是提高 Android 应用程序运行速度的几条原则。

1）不要让 UI 线程等待

2）耗时操作不可取

3）模拟器和真实的设备有不同

4）通知用户，要注意用户体验。

2.18 Q：Android 运行库有哪些？

A：Android 包括了一个核心库，该核心库提供了 JAVA 编程语言核心库的大多数功能。每一个 Android 应用程序都在它自己的进程中运行，都拥有一个独立的 Dalvik 虚拟机实例。Dalvik 被设计成一个设备可以同时高效地运行多个虚拟系统。Dalvik 虚拟机执行（.dex）的 Dalvik 可执行文件，该格式文件针对小内存使用做了优化。同时虚拟机是基于寄存器的,所有的类都经由 JAVA 编译器编译，然后通过 SDK 中的"dx" 工具转化成.dex 格式由虚拟机执行。Dalvik 虚拟机依赖于 linux 内核的一些功能，比如线程机制和底层内存管理机制。

2.19 Q：Android 都支持哪些 Java 特性

A：针对于 Java SE 或 Java EE 的程序员想转到 Android 平台上进行开发，有以下几点常规的支持:

1）目前来看 JDK 的高级特性均支持，比如说 1. Java 的反射、2. NIO (New I/O)、3. JNI (Java Native Interface)相对而言 对于 OpenGL 和 SQLite 的支持比较强大，但是 AWT 和 JDBC 这些东西都不支持。

2）在 Xml 解析上，兼容 DOM、XmlPull 和 SAX，同时数据交换格式上，Android 虽然不支持 LINQ 但对于 Java 来说 JSON 同样支持。

3）对于 Http 处理方面，提供了轻量级的 Http 处理类，以及更完善的 Apache 库支持。

4）音频方面 Android 比较强大，使用了 OpenCore 库，很多地方我们可以自己编写编码、解码器进行扩展。

5）Android 在文件系统上基本山和 Java 是相同的，不过对于高效率的内存影射文件而言提供了 android.os.MemoryFile 这个类。

总体而言，Java 程序员转入 Android 开发只需要了解平台特有的，

Intent，Service，Receiver 和 Activity 就差不多了，深入了解下 AIDL 和 UI 控件和自定义 Widget 基本上可以胜任常规的工作。

2.20 Q：Android 最简单播放 GIF 动画方法是什么？

A:GIF 动画的原理就是逐帧播放,在 Android 中提供了 AnimationDrawable 类可以实现,GIF89A 的解码方法在过去的 J2ME 平台移植到 Android 平台也能用，其实在 Google Android 上面开发目前 2.2 以后的固件支持的方法除了 Flash Player 外，更好的兼容方法就是使用万能的 webkit 浏览器了。直接在工程中内嵌一个 webView,当然了路径大家可以换成本地的,对于浏览器使用本地资源 url 为 file://开头。

不过 webView 的资源消耗也不小，开个 webView 对象可能占用了至少 8MB 的 RAM 吧，保守估计，当然更多的要看插件和以及 html 的复杂程度所决定的。

2.21 Q：线程在 Android 应用当中的作用？

A：1）动态更新 UI 如 AsyncTask 类，在开发 Android 应用时必须遵守单线程模型的原则： Android UI 操作并不是线程安全的并且这些操作必须在 UI 线程中执行

2）SOCKET 之间的通信

3）文件的下载

4）与服务端之间的交互

5）复杂数据和逻辑的处理

2.22 Q：如何成为高手？

A：成为一名真正的 Android 高手必须掌握和遵循的一些准则：

1）学会懒惰

a Don’t Reinvent the Wheel（不要重复发明轮子）。

b Inventing the Wheel(发明轮子)。

c Don't Reinvent the Wheel（不要重复发明轮子)。

“轮子理论”，也即“不要重复发明轮子”，这是西方国家的一句谚语，原话是：Don't Reinvent the Wheel。“不要重复发明轮子 ”意思是企业中任何一项工作实际上都有人做过，我们所需要做的就是找到做过这件事情的人。拿到软件领域中就是指有的项目或功能，别人已经做过，我们需要用的时候，直接拿来用即可，而不要重新制造。

2）精通 Android 体系架构、MVC、常见的设计模式、控制反转（IoC）

3）编写可重用、可扩展、可维护、灵活性高的代码

4）高效的编写高效的代码

5）学会至少一门服务器端开发技术

2.23 Q：JNI 是什么？

A:JNI 是 Java Native Interface 的缩写,中文为 JAVA 本地调用。从 Java1.1 开始，Java Native Interface(JNI)标准成为 java 平台的一部分，它允许 Java 代码和其他语言写的代码进行交互。JNI 一开始是为了本地已编译语言，尤其是 C 和 C++而设计的，但是它并不妨碍你使用其他语言，只要调用约定受支持就可以了。

使用 java 与本地已编译的代码交互，通常会丧失平台可移植性。但是，有些情况下这样做是可以接受的，甚至是必须的，比如，使用一些旧的库，与硬件、操作系统进行交互，或者为了提高程序的性能。JNI 标准至少保证本地代码能工作在任何 Java 虚拟机实现下。

2.24 Q：Android 和 Linux 的区别？

A：Android 是一个专门针对移动设备的软件集，包括一个操作系统，中间件和一些重要的应用程序。Android SDK 提供了在 Android 平台使用 java 语言进行 android 应用程序开发必须的工具和 API 接口。Android 系统架构除了 Linux2.6 内核之外，还提供了丰富的 lib 和适用于 java 的运行时环境(很重要

的一点是 Dalvik 虚拟机，类似于 JVM)、应用程序框架和核心应用。在此基础之上可以快速开发应用程序。

linux 内核像是核心技术支持，而 google 将其商业化于移动设备上。android 上开发应用程序和 Windows MFC 开发有些类似。

2.25 Q：SDK 升级了，如何更新 SDK？

A:更新 SDK 是非常棘手的。当一个新的 SDK 发布，必须是 plugin 也发布。更新容易出现的错误是两个版本都存在，而且都不正常。最终不得不卸载了它们并且重新安装最新的一个。然后那个最新的 SDK 工作正常了。建议任何面对 SDK 或者 plugin 升级的人都采用相同的过程。简单的卸载老版本，然后安装新版本。不要升级。

2.26 Q：如何进入 Recovery 模式

A:每部 Android 设备进入 Recovery 模式的方法不同。以 Milestone 为例：若 bootloader 为 90.78，按住键盘的“X”键，再按电源键开机，看到 moto 的经典 logo 即可放开此 2 键。等待出现一个三角形图标，然后按住音量向上键+轻按拍照键，会出现四个选项。放开按键，可以用方向键选择一个选项。作用分别为重启，应用 update.zip，清除所有数据至出厂状态、清除 cache。

2.27 Q：Android bionic 移植需要注意的事项

A：下面通过一个例子来说明移植 binic 需要做的事情（BIONIC 使用的内核头文件从 2.6.29 升级到 2.6.31）：

1）修改之前，先要备份 android/bionic

2）按照下面方法复制 linux kernel v2.6.31 头文件:

复制 include/asm-generic 到 bionic/libc/kernel/original folder

复制 include/linux 到 bionic/libc/kernel/original folder

复制 include/mtd 到 bionic/libc/kernel/original folder

3）运行 tools/update_all.py 脚本，生成新的头文件，方法如下.

$ cd android/bionic/libc/kernel

$ tools/update_all.py

以上的操作将复制新的头文件到 bionic/libc/kernel/common 目录

4）删除 bionic/libc/kernel/original 目录

5）复制平台相关的头文件：

复制 linux v2.6.31 平台相关的 asm 目录到指定的路径，方法如下：

复制 arch/arm/include/asm 到 android/bionic/libc/kernel/arch-arm/

复制 arch/x86/include/asm 到 android/bionic/libc/kernel/arch-x86/

6）从 v2.6.29 bionic/libc/kernel/common/linux 复制一下的头文件到新的 android/bionic/libc/kernel/common/linux 目录：

android-alarm.h

android-pmem.h

android_power.h

ashmem.h

binder.h

msm_adsp.h

msm_audio.h

msm_mdp.h

keychord.h

7）编译

$ cd android

```
$ make -j4
```

2.28 Q：Android 源码下 vendor 目录下文件的作用？

A：指的是某些公司的产品型号，主要是一些针对硬件配置的东西,比如 radio 方面的,若是采用 TI 的,这里面会有 TI 提供的一些 RIL 方面的接口等。

2.29 Q：编译 Android 源码和编译 Android 内核有什么区别？

A:其实没什么区别,就多了几个 android 特有的驱动,最重要的是 binder

2.30 Q：如何获取手机和存储卡上的图片？

A:

```
1.  /**
2.         * 通过 uri 获取文件的绝对路径
3.         * @param uri
4.         * @return
5.         */
6.              protected String getAbsoluteImagePath(Uri uri)
7.        {
8.              // can post image
9.              String [] proj={MediaStore.Images.Media.DATA};
10.             Cursor cursor = managedQuery( uri,
11.                                           proj,
          // Which columns to return
12.                                           null,           // WHERE
    clause; which rows to return (all rows)
13.                                           null,           // WHERE
    clause selection arguments (none)
14.                                           null);
           // Order-by clause (ascending by name)
15.
16.             int column_index =
    cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
```

```
17.             cursor.moveToFirst();
18.
19.             return cursor.getString(column_index);
20.     }
```

2.31 Q：如何使用双缓冲？

A：系统自身已经实现了双缓冲，也就是说为了避免重影，必须连续 onDraw 两次，或者是对上一次 onDraw 里面画脏的部分进行修复。

执行 Bitmap.createBitmap 函数一般要花销 100-300ms 的时间，要想提高它的效率，只能祈祷 google 能把 android 的代码写得更高效一些了。