logo2011 

**“核高基”科技重大专项**

**课题1-4：开源操作系统内核分析和安全性评估**

**详细设计说明书**

**（Android源码编译环境配置）**

**中国科学院软件研究所**

**2013年3月27日**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本编号** | **变更内容** | **变更日期** | **变更人** | **审核日期** | **审核人** |
| V1.0 | 创建文档 | 2013.3.27 | 吴西飞 |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# 初始化编译环境

## 1.1、系统选择

Android源码可以在Linux或Mac OS上编译，大小8.5G左右。源码编译环境在Ubuntu LTS (10.04)经过测试，也可以选择在其它的Linux环境下编译，对于Gingerbread (2.3.x)以上的版本，需要使用64位的编译环境，编译中需要用到的工具有：Python、GNU Make、JDK、Git等。本文选用的系统为Ubuntu LTS (10.04)，以下的环境配置及编译都是在此系统下。

## 1.2、安装JDK

一、安装JDK。首先从SUN网站下载JDK，在shell中执行代码：sudo chmod +x jdk-6u17-linux-i586.bin。修改bin文件权限，使其可执行，然后执行代码：jdk-6u17-linux-i586.bin

将会出现字幕，持续按回车键，直到屏幕出现需要输入yes/no,此时输入yes/y 回车，将会把JDK解压到文件夹，得到jdk1.6.0\_17.此时JDK安装完毕。

二、配置JDK环境。执行代码：

sudo gedit /etc/environment

在 environment中修改如下信息：

PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:（java解压位置）/bin"

export JAVA\_HOME=（java解压位置）

export JRE\_HOME=（java解压位置）/jre

export CLASSPATH=（java解压位置）/lib

export PATH=~/bin:$JAVA\_HOME/bin:$PATH

三、选择后来安装的JDK为默认JDK。ubuntu10.04中有默认jdk还需要执行代码：

sudo update-alternatives --install /usr/bin/java java （java解压位置）/bin/java 300

sudo update-alternatives --install /usr/bin/javac javac （java解压位置）/bin/javac 300

sudo update-alternatives --install /usr/bin/jar jar （java解压位置）/bin/jar 300

通过这一步将我们安装的JDK加入java选单，然后执行代码：

sudo update-alternatives --config java

sudo update-alternatives --config javac

sudo update-alternatives --config jar

通过这一步设置系统默认的JDK

然后在shell中执行代码：

java -version

此时显示的系统中的java就是刚刚安装的 java。安装成功

## 1.3、安装其它用到的工具

执行命令：sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl zlib1g-dev libc6-dev lib32ncurses5-dev ia32-libs x11proto-core-dev libx11-dev lib32readline5-dev lib32z-dev libgl1-mesa-dev g++-multilib mingw32 tofrodos python-markdown libxml2-utils xsltproc

## 1.4、其它相关配置

### 1.4.1、配置USB访问权限

在GNU/linux系统中，普通用户是不能直接访问USB设备的，需要进行相关配置。使用root用户创建文件/etc/udev/rules.d/51-android.rules，并将下面的文字复制进去，<username>需要使用实际的用户名代替。

|  |
| --- |
| # adb protocol on passion (Nexus One)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e12", MODE="0600", OWNER="<username>"  # fastboot protocol on passion (Nexus One)  SUBSYSTEM=="usb", ATTR{idVendor}=="0bb4", ATTR{idProduct}=="0fff", MODE="0600", OWNER="<username>"  # adb protocol on crespo/crespo4g (Nexus S)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e22", MODE="0600", OWNER="<username>"  # fastboot protocol on crespo/crespo4g (Nexus S)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e20", MODE="0600", OWNER="<username>"  # adb protocol on stingray/wingray (Xoom)  SUBSYSTEM=="usb", ATTR{idVendor}=="22b8", ATTR{idProduct}=="70a9", MODE="0600", OWNER="<username>"  # fastboot protocol on stingray/wingray (Xoom)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="708c", MODE="0600", OWNER="<username>"  # adb protocol on maguro/toro (Galaxy Nexus)  SUBSYSTEM=="usb", ATTR{idVendor}=="04e8", ATTR{idProduct}=="6860", MODE="0600", OWNER="<username>"  # fastboot protocol on maguro/toro (Galaxy Nexus)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e30", MODE="0600", OWNER="<username>"  # adb protocol on panda (PandaBoard)  SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d101", MODE="0600", OWNER="<username>"  # fastboot protocol on panda (PandaBoard)  SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d022", MODE="0600", OWNER="<username>"  # usbboot protocol on panda (PandaBoard)  SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d00f", MODE="0600", OWNER="<username>"  # usbboot protocol on panda (PandaBoard ES)  SUBSYSTEM=="usb", ATTR{idVendor}=="0451", ATTR{idProduct}=="d010", MODE="0600", OWNER="<username>"  # adb protocol on grouper/tilapia (Nexus 7)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e42", MODE="0600", OWNER="<username>"  # fastboot protocol on grouper/tilapia (Nexus 7)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4e40", MODE="0600", OWNER="<username>"  # adb protocol on manta (Nexus 10)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee2", MODE="0600", OWNER="<username>"  # fastboot protocol on manta (Nexus 10)  SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee0", MODE="0600", OWNER="<username>" |

其中ATTR{idVendor}是供应商的编号，ATTR{idProduct}是设备编号。如果上面的文字中没有你使用的设备信息，需要将信息按照相应的格式加入文件。

### 1.4.2、配置Ccache

Ccache（“compiler cache”的缩写）是一个编译器缓存，该工具会高速缓存编译生成的信息，并在编译的特定部分使用高速缓存的信息，比如头文件，这样就节省了通常使用cpp解析这些信息所需要的时间。如果您编译清单2中的文件，假定foobar.h中包含对其他头文件的引用，ccache会用那个文件的 cpp-parsed版本来取代include声明。ccache只是将最终的文本拷贝到文件中，使得它可以立即被编译，而不是真正去读取、理解并解释其内容。

这一步是可选的。我在编译的时候没有配置ccache，最后的编译时间用了至少六个小时以上，虽然目前我还没有尝试过配置ccache后的编译速度，这里还是建议读者配置一下，以节省编译时间，不要像我这么悲催。配置方法如下：

打开用户目录下的.bashrc，将下面语句添加到文件后面并保存退出： Export USE\_CCACHE =1

默认的cache会存储在~/.cache下，可通过在.bashrc文件中选择特定的目录：export CCACHE\_DIR=<path-to-your-cache-directory>。

配置cache的大小：prebuilts/misc/linux-x86/ccache/ccache -M 50G，如果android版本为 Ice Cream Sandwich (4.0.x)或之前的，cache是在另外的目录：prebuilt/linux-x86/ccache/ccache -M 50G，这个配置是持久保存在CCACHE\_DIR中。

# 下载Android源码

## 2.1、使用未认证方式进行下载

为了便于使用Git对Android源码进行管理，Google使用python开发了Repo工具。在HOME目录下创建bin文件夹，并加bin文件路径加入到PATH环境变量中。

$ mkdir ~/bin

$ PATH=~/bin:$PATH

将Repo脚本下载到bin文件夹下，并将repo属性设为可执行。

$ curl https://dl-ssl.google.com/dl/googlesource/git-repo/repo > ~/bin/repo

$ chmod a+x ~/bin/repo

初始化repo客户端

创建工作目录，将Android源码下载到工作目录中：

$ mkdir WORKING\_DIRECTORY

$ cd WORKING\_DIRECTORY

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

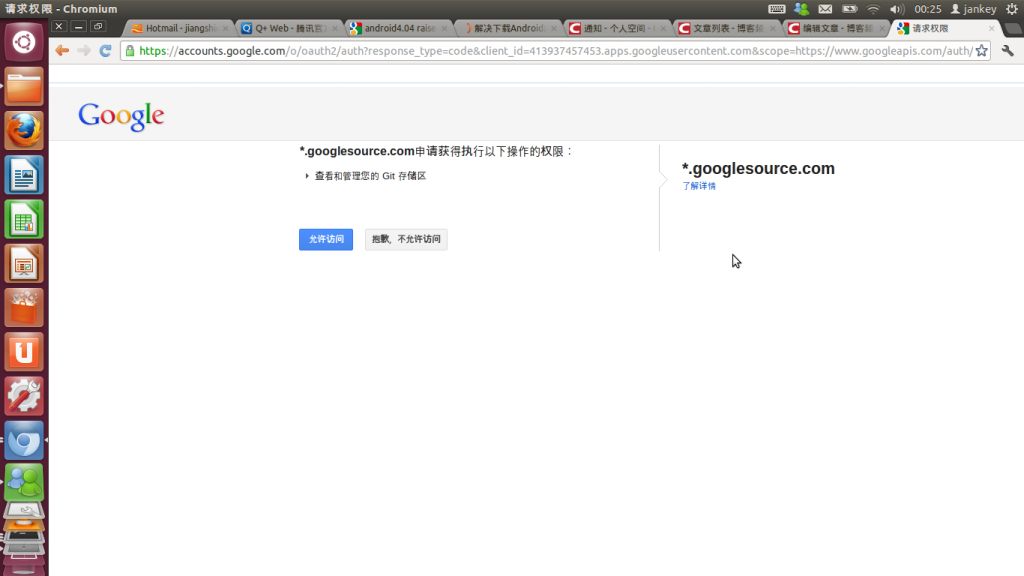
如果要下载一个分支而不是当前最新版本，需要在url后加-b参数：

$ repo init -u [https://android.googlesource.com/platform/manifest -b android-4.0.4\_r1](https://android.googlesource.com/platform/manifest%20-b%20android-4.0.4_r1)

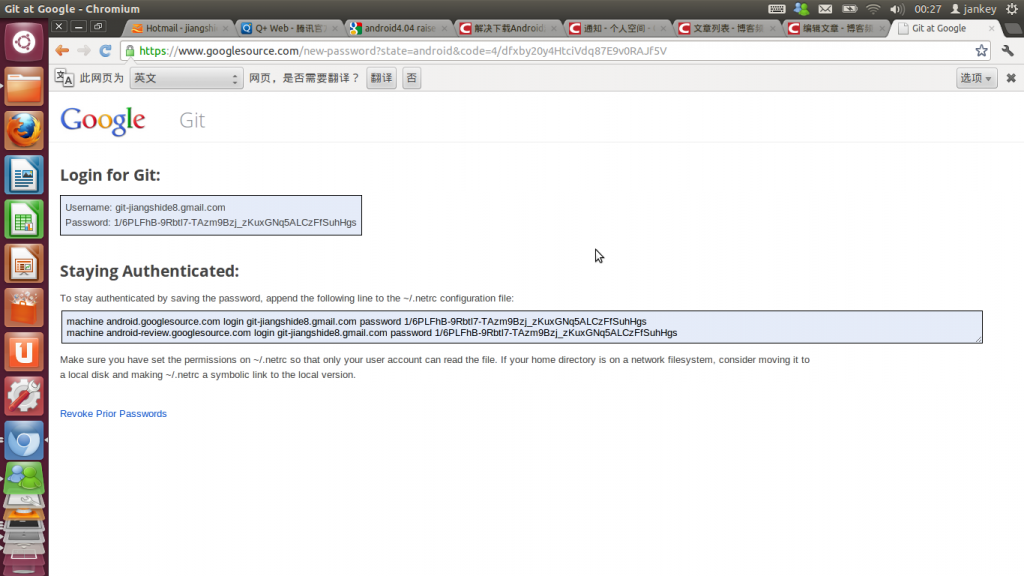
运行repo sync命令，将源码下载到本地：$ repo sync（注：repo sync的过程比较慢，中间还有可能出现断网的现象，断网后可以再次执行repo sync指令，继续上一次才下载。如果想获得更快一些的下载方式，可以参考2.2）

## 2.2、使用认证方式进行下载

通过访问：https://android.googlesource.com/new-password来获得相关认证的用户名与密码：



点击允许，进入页面,要求你输入一个邮箱的用户名与密码，然后得出其以下页面显示：



将Staying Authenticated的内容复制到~/.netrc文件中，初始化repo并执行同步指令：

$ repo init -u https://android.googlesource.com/a/platform/manifest

$ repo sync -j4

## 2.3、添加公钥

将如下公钥加入到GnuPG key数据库中：

$ gpg --import

输入如下公钥，（Ctrl-D表示输入的结束）

|  |
| --- |
| -----BEGIN PGP PUBLIC KEY BLOCK-----  Version: GnuPG v1.4.2.2 (GNU/Linux)  mQGiBEnnWD4RBACt9/h4v9xnnGDou13y3dvOx6/t43LPPIxeJ8eX9WB+8LLuROSV  lFhpHawsVAcFlmi7f7jdSRF+OvtZL9ShPKdLfwBJMNkU66/TZmPewS4m782ndtw7  8tR1cXb197Ob8kOfQB3A9yk2XZ4ei4ZC3i6wVdqHLRxABdncwu5hOF9KXwCgkxMD  u4PVgChaAJzTYJ1EG+UYBIUEAJmfearb0qRAN7dEoff0FeXsEaUA6U90sEoVks0Z  wNj96SA8BL+a1OoEUUfpMhiHyLuQSftxisJxTh+2QclzDviDyaTrkANjdYY7p2cq  /HMdOY7LJlHaqtXmZxXjjtw5Uc2QG8UY8aziU3IE9nTjSwCXeJnuyvoizl9/I1S5  jU5SA/9WwIps4SC84ielIXiGWEqq6i6/sk4I9q1YemZF2XVVKnmI1F4iCMtNKsR4  MGSa1gA8s4iQbsKNWPgp7M3a51JCVCu6l/8zTpA+uUGapw4tWCp4o0dpIvDPBEa9  b/aF/ygcR8mh5hgUfpF9IpXdknOsbKCvM9lSSfRciETykZc4wrRCVGhlIEFuZHJv  aWQgT3BlbiBTb3VyY2UgUHJvamVjdCA8aW5pdGlhbC1jb250cmlidXRpb25AYW5k  cm9pZC5jb20+iGAEExECACAFAknnWD4CGwMGCwkIBwMCBBUCCAMEFgIDAQIeAQIX  gAAKCRDorT+BmrEOeNr+AJ42Xy6tEW7r3KzrJxnRX8mij9z8tgCdFfQYiHpYngkI  2t09Ed+9Bm4gmEO5Ag0ESedYRBAIAKVW1JcMBWvV/0Bo9WiByJ9WJ5swMN36/vAl  QN4mWRhfzDOk/Rosdb0csAO/l8Kz0gKQPOfObtyYjvI8JMC3rmi+LIvSUT9806Up  hisyEmmHv6U8gUb/xHLIanXGxwhYzjgeuAXVCsv+EvoPIHbY4L/KvP5x+oCJIDbk  C2b1TvVk9PryzmE4BPIQL/NtgR1oLWm/uWR9zRUFtBnE411aMAN3qnAHBBMZzKMX  LWBGWE0znfRrnczI5p49i2YZJAjyX1P2WzmScK49CV82dzLo71MnrF6fj+Udtb5+  OgTg7Cow+8PRaTkJEW5Y2JIZpnRUq0CYxAmHYX79EMKHDSThf/8AAwUIAJPWsB/M  pK+KMs/s3r6nJrnYLTfdZhtmQXimpoDMJg1zxmL8UfNUKiQZ6esoAWtDgpqt7Y7s  KZ8laHRARonte394hidZzM5nb6hQvpPjt2OlPRsyqVxw4c/KsjADtAuKW9/d8phb  N8bTyOJo856qg4oOEzKG9eeF7oaZTYBy33BTL0408sEBxiMior6b8LrZrAhkqDjA  vUXRwm/fFKgpsOysxC6xi553CxBUCH2omNV6Ka1LNMwzSp9ILz8jEGqmUtkBszwo  G1S8fXgE0Lq3cdDM/GJ4QXP/p6LiwNF99faDMTV3+2SAOGvytOX6KjKVzKOSsfJQ  hN0DlsIw8hqJc0WISQQYEQIACQUCSedYRAIbDAAKCRDorT+BmrEOeCUOAJ9qmR0l  EXzeoxcdoafxqf6gZlJZlACgkWF7wi2YLW3Oa+jv2QSTlrx4KLM=  =Wi5D  -----END PGP PUBLIC KEY BLOCK----- |

加入公钥后，可以通过$ git tag -v TAG\_NAME确认任何tag。

# Android源码的编译

## 3.1、面向模拟器的源码编译

使用build/envsetup.sh初始化编译环境：$ source build/envsetup.sh或$ . build/envsetup.sh

运行lunch指令，选择编译目标

$ lunch

选择full-eng，将进行面向模拟器的编译。

运行make指令进行源码编译：

$ make -j4 (otapackage 生成ROM包)

-j后面的参数标识编译后的系统面向的线程数目。

使用emulator指令，将在模拟器中显示系统的效果：$ emulator

编译结束后，emulator在~/android/out/host /linux-x86/bin下，ramdisk.img，system.img和userdata.img则在~/android/out /target/product/generic下

$ cd ~/android/out/host/linux-x86/bin

增加环境变量

$ emacs ~/.bashrc

在.bashrc中新增环境变量，如下

#java 程序开发/运行的一些环境变量

export ANDROID\_PRODUCT\_OUT=~/android/out/target/product/generic

ANDROID\_PRODUCT\_OUT\_BIN=~/android/out/host/linux-x86/bin

export PATH=${PATH}:${ANDROID\_PRODUCT\_OUT\_BIN}:${ANDROID\_PRODUCT\_OUT};

最后，同步这些变化：

$ source ~/.bashrc

$ cd ~/android/out/target/product/generic

$ emulator -system system.img -data userdata.img -ramdisk ramdisk.img

最后进入android桌面，就说明成功了。

## 3.2、面向设备的Android源码编译

面向设备的android源码编译，需要在输入lunch指令后选择设备名。本文面向的设备是Google Nexus Prime（Samsung I9250），设备名为maguro，在lunch菜单中选择maguro-eng。

进入[https://developers.google.com/android/nexus/drivers#maguro](http://mail.qq.com/cgi-bin/mail_spam?action=check_link&spam=0&sid=Mn2e73CyzD9oMx1P&url=https%3A%2F%2Fdevelopers.google.com%2Fandroid%2Fnexus%2Fdrivers%23maguro&mailid=ZC4011-byCPPaLKfo3aTGIw56nDJ3a" \t "_blank)，下载脚本文件，安装maguro设备需要的驱动：Wi-Fi, Bluetooth、Graphics 、GSM。

1、source build/envsetup.sh

2、lunch full\_maguro\_userdebug

3、编译

方式一：make -j4 otapackage。可以在/android/out/target/product/maguro生成对应的ROM包和相关镜像，将ROM包加载到SD卡，通过recovery也可以将系统安装到设备上。

方式二：make –j4。会在/android/out/target/product/maguro目录下生成镜像文件：boot.img、system.img、userdata.img、recovery.img。

4、安装

相关工具：adb、fastboot，位于android源码目录的out/host/linux-x86文件夹下。

查看设备连接，启动设备，adb device查看设备是否连接，adb reboot-bootloader进入bootloader。

安装ROM包方式：刷入recovery，将ROM包放入sdcard，使用recovery工具刷机。fastboot flash recovery recovery.img; adb push origin.zip remote.

通过fastboot指令可以将镜像文件写入到设备的对应分区，一般刷入boot、system分区系统就可以正常运行：

fastboot flash boot boot.img

fastboot flash system system.img

fastboot flash userdata userdata.img

fastboot flash recovery recovery.img

## 3.3、maguro设备内核编译

在Android4.0.4源码中kernel是以镜像而不是源码的形式存在的，所以可以自己找到设备对应的源码，编译生成镜像文件，并覆盖Android项目自带的kernel，可以进行kernel的定制。

在Android的编译环境中，可以进行kernel的编译。

1、下载内核源码

git clone https://android.googlesource.com/kernel/omap.git

2、下载交叉编译器

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

export PATH=~/android/kernel/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin:$PATH

$ export ARCH=arm

$ export SUBARCH=arm

$ export CROSS\_COMPILE=arm-eabi-

$ cd omap

$ git checkout -b android-omap-tuna-3.0 origin/ android-omap-tuna-3.0

$ make tuna\_defconfig

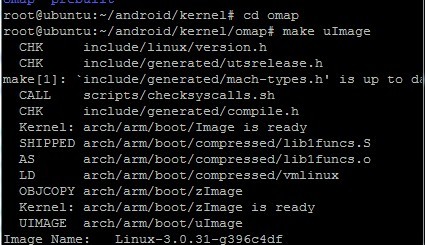
$ make –j4

等待编译，编译过程不长。

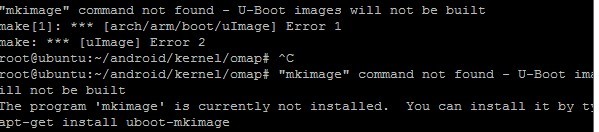


出现如下：即表示编译通过了。

3、如果要使用uboot来启动的话，还需要生成uImage。使用make uImage即可，编译完效果如下：



不过，为了正确得到uImage文件，之前需要安装一下uboot-mkimage工具。使用如下命令进行安装：sudo apt-get install uboot-mkimage。如果不安装进行编译的话，将出现如下错误：



# 发布到真机（maguro设备）

编译完成后执行:

root@ubuntu:~/Android# fastboot -w flashall -p /root/Android/out/target/product/maguro

显示如下：

--------------------------------------------

Bootloader Version...: PRIMEKK15

Baseband Version.....: I9250XXKK1

Serial Number........: 0146AFFC10009006

--------------------------------------------

checking product...

OKAY [ 0.007s]

checking version-bootloader...

OKAY [ 0.010s]

checking version-baseband...

FAILED

Device version-baseband is 'I9250XXKK1'.

Update requires 'I9250XXKK6' or 'I9250XXLA1' or 'I9250XXLA2'.

finished. total time: 0.052s

root@ubuntu:~/Android#

若出现上述错误，说明当前版本的基带不对，根据给出的提示到网上下载对应的基带版本，放到out/target/product/maguro/下，执行如下语句

root@ubuntu:~/Android# fastboot flash radio /root/Android/out/target/product/maguro/radio.img

sending 'radio' (12288 KB)...

OKAY [ 1.451s]

writing 'radio'...

OKAY [ 2.189s]

finished. total time: 3.640s

此时，基带版本已经替换完成。将编译好的img文件，刷入到真机。

root@ubuntu:~/Android# fastboot -w flashall -p /root/Android/out/target/product/maguro

--------------------------------------------

Bootloader Version...: PRIMEKK15

Baseband Version.....: I9250XXKK6

Serial Number........: 0146AFFC10009006

--------------------------------------------

checking product...

OKAY [ 0.008s]

checking version-bootloader...

OKAY [ 0.007s]

checking version-baseband...

OKAY [ 0.008s]

sending 'boot' (4148 KB)...

OKAY [ 0.499s]

writing 'boot'...

OKAY [ 0.723s]

sending 'recovery' (4478 KB)...

OKAY [ 0.539s]

writing 'recovery'...

OKAY [ 0.637s]

sending 'system' (157938 KB)...

OKAY [ 18.525s]

writing 'system'...

OKAY [ 20.959s]

erasing 'userdata'...

OKAY [ 0.550s]

erasing 'cache'...

OKAY [ 0.024s]

rebooting...

finished. total time: 42.514s

# 参考文献

http://source.android.com/source/building.html

http://dev.10086.cn/cmdn/wiki/index.php?doc-view-3882.html#13

http://blog.csdn.net/wait2kyears/article/details/6230212

<http://www.cnblogs.com/froyo/archive/2011/03/20.html>

http://blog.csdn.net/wu070815/article/details/8186547