技术报告---Android系统编译和刷机步骤

# 1．Establish Android Build Environment

    Android应用使用Java语言开发，底层使用C/C++开发，Android运行在Linux内核上（从标准的Linux内核修改移植而来）。因此掌握Java语言和C/C++语言是进行Android系统开发的必要条件；如果想要成为优秀的系统开发者，深入理解Linux内核则非常有必要。

    Android系统开发的目的是制作Android系统软件，如手机或者Pad的ROM，或者增加、修改Android的系统模块。

[**Android Open Source  Project**](https://source.android.com/)

# [​Establishing a Build Environment](https://source.android.com/source/initializing.html)

[Requirements](https://source.android.com/source/requirements.html)

* Host OS : Ubuntu 14.04 64-bit

$ sudo apt-get install git-core gnupg flex bison gperf build-essential \

zip curl zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 \

lib32ncurses5-dev x11proto-core-dev libx11-dev lib32z-dev ccache \

libgl1-mesa-dev libxml2-utils xsltproc unzip

* Java Version

        1. Android Version >= 2.3.x 64-bit Operating System

             >= **5.x**(Lollipop) – **Java7** （支持OpenJDK 1.7）

             2.3.x(Gingerbread) ~ **4.4.x**(Kitkat) – **Java 6  (**for Gingerbread through KitKat)

         2. Android Version < 2.3.x 32-bit OS

             1.5(Cupcake) ~ 2.2.x(Froyo) – Java 5  (for Cupcake through Froyo)

* JDK

        由于Android版本和Java JDK版本更新，编译不同版本的Android系统需要不同的Java版本，其中目前应用最广泛的4.4.x Android版本对应Java 6；最新5.xAndroid系统对应Java 7。

         为了便捷切换Java版本分别编译4.x或5.x的系统，在Ubuntu中使用PPA安装Java：

  sudo add-apt-repository ppa:webupd8team/java  //添加软件仓库地址

sudo apt-get update

sudo apt-get install oracle-java8-installer   //安装Java 8

sudo apt-get install oracle-java7-installer    //安装Java 7

sudo apt-get install oracle-java6-installer    //安装Java 6

javac –version    //查看java版本

sudo apt-get install oracle-java8-set-default  //以Java 8为例设置环境变量

安装多版本Java之后，以后根据需要使用以下命令查看当前工作的Java版本和进行Java版本、环境变量的便捷更换：

javac –version    //查看java版本

sudo update-java-alternatives -s java-7-oracle    //版本切换，为Java 7

参考链接：

[在 Ubuntu 中使用 PPA 安装 Java 8 (支持 Ubuntu 10.04 - Ubuntu 14.04 )](http://www.linuxidc.com/Linux/2014-03/98691.htm)

安装OpenJDK1.7

#For the latest version of Android

sudo apt-get update

sudo apt-get install openjdk-7-jdk

# Optionally, update the default Java version by running:

sudo update-alternatives --config java

sudo update-alternatives --config javac

## 1.2 Configuring USB Access

​    在GNU/Linux中默认普通用户无法直接访问USB设备，如果要访问需要配置（推荐的方法是以root权限创建文件 ‘ /etc/udev/rules.d/51-android.rules ’）。

$ wget -S -O - http://source.android.com/source/51-android.rules | sed "s/<username>/$USER/" | sudo tee >/dev/null /etc/udev/rules.d/51-android.rules; sudo udevadm control --reload-rules

# adb protocol on hammerhead (Nexus 5)

SUBSYSTEM=="usb", ATTR{idVendor}=="18d1", ATTR{idProduct}=="4ee1", MODE="0600", OWNER=""

    If adb was already running and cannot connect to the device after getting those rules set up, it can be killed with **adb kill-server**. That will cause adb to restart with the new configuration.

使用独立或者其他output文件夹，设置**OUT\_DIR\_COMMON\_BASE**变量

export OUT\_DIR\_COMMON\_BASE=<path-to-your-out-directory>

    优化编译，设置ccache；如果经常使用make clean命令，或者频繁切换编译不同的产品，ccache作为C和C++编译器的缓存可以再重复编译时起到加速的作用。

#in .bashrc

export USE\_CCACHE=1 #By default the cache will be stored in ~/.ccache(CCACHE\_DIR)

#The suggested cache size is 50-100GB.

$ prebuilts/misc/linux-x86/ccache/ccache -M 50G

#This setting is stored in the CCACHE\_DIR and is persistent.

或者在源码的根目录运行如下命令：

$ export USE\_CCACHE=1

$ export CCACHE\_DIR=/<path\_of\_your\_choice>/.ccache

$ prebuilts/misc/linux-x86/ccache/ccache -M 50G

# watch the used ccache

$ watch -n1 -d prebuilts/misc/linux-x86/ccache/ccache -s

## 1.3 [Downloading the Source](https://source.android.com/source/downloading.html)

安装Repo

$ mkdir ~/bin

$ PATH=~/bin:$PATH

$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo

$ chmod a+x ~/bin/repo

$ mkdir android-build

$ cd android-build/

$ repo init -u https://android.googlesource.com/platform/manifest # -u 参数初始化软件仓库

$ repo init -u https://android.googlesource.com/platform/manifest -b android-6.0.1\_r9 # -b 参数指定某个分支，默认为master分支

.repo/manifests/: discarding 211 commits

Your identity is: NickHuang <xjhznick@gmail.com>

If you want to change this, please re-run 'repo init' with --config-name

repo has been initialized in /home/nick/android-build

$ repo sync [-j4] # sync命令同步代码，-j参数启动多个线程同时下载

[Source Code Tags and Builds](https://source.android.com/source/build-numbers.html#source-code-tags-and-builds) -- 据此选择设备对应的Android源码branch，例如：

    MMB29S    android-6.0.1\_r9    Marshmallow    Nexus 5, Nexus 6, Nexus 9 (volantis/volantisg)

最新的适用于Nexus5的Android源码编译代号为MMB29S，检出的源码代号为android-6.0.1\_r9。

## Q&A

1.

curl: (22) The requested URL returned error: 404 Not Found

Server does not provide clone.bundle; ignoring.

    Stack Overflow-What to do about curl clone.bundle error on AOSP repo sync

    Repo attempts to download a prepackaged bundle file to bootstrap each git prior to downloading the most recent data via Git's HTTP protocol. The latter is more expensive on the server side and results in worse performance so the bundle file allows the download to cut some corners. If a bundle file isn't available (like in this case), Repo will ignore it and proceed anyway. In other words, don't pay any attention to this.

2. [Troubleshooting Common Build Errors](https://source.android.com/source/building.html#troubleshooting-common-build-errors)

# 2．Build Android System

# 2.1 [Building the System](https://source.android.com/source/building.html)

$ source build/envsetup.sh $ . build/envsetup.sh

$ lunch aosp\_arm-eng

$ make -j4

+ ENABLE\_SPARSE\_IMAGE=

+ '[' out/target/product/generic/system = -s ']'

+ '[' 6 -ne 5 -a 6 -ne 6 ']'

+ SRC\_DIR=out/target/product/generic/system

+ '[' '!' -d out/target/product/generic/system ']'

+ OUTPUT\_FILE=out/target/product/generic/obj/PACKAGING/systemimage\_intermediates/system.img

+ EXT\_VARIANT=ext4

+ MOUNT\_POINT=system

+ SIZE=576716800

+ FC=out/target/product/generic/root/file\_contexts

+ case $EXT\_VARIANT in

+ '[' -z system ']'

+ '[' -z 576716800 ']'

+ '[' -n out/target/product/generic/root/file\_contexts ']'

+ FCOPT='-S out/target/product/generic/root/file\_contexts'

+ MAKE\_EXT4FS\_CMD='make\_ext4fs  -S out/target/product/generic/root/file\_contexts -l 576716800 -a system out/target/product/generic/obj/PACKAGING/systemimage\_intermediates/system.img out/target/product/generic/system'

+ echo make\_ext4fs -S out/target/product/generic/root/file\_contexts -l 576716800 -a system out/target/product/generic/obj/PACKAGING/systemimage\_intermediates/system.img out/target/product/generic/system

make\_ext4fs -S out/target/product/generic/root/file\_contexts -l 576716800 -a system out/target/product/generic/obj/PACKAGING/systemimage\_intermediates/system.img out/target/product/generic/system

+ make\_ext4fs -S out/target/product/generic/root/file\_contexts -l 576716800 -a system out/target/product/generic/obj/PACKAGING/systemimage\_intermediates/system.img out/target/product/generic/system

Creating filesystem with parameters:

   Size: 576716800

   Block size: 4096

   Blocks per group: 32768

   Inodes per group: 7040

   Inode size: 256

   Journal blocks: 2200

   Label:

   Blocks: 140800

   Block groups: 5

   Reserved block group size: 39

Created filesystem with 1277/35200 inodes and 82233/140800 blocks

+ '[' 0 -ne 0 ']'

Install system fs image: out/target/product/generic/system.img

out/target/product/generic/system.img+ maxsize=588791808 blocksize=2112 total=576716800 reserve=5947392

nick@Dell:~/android-source-google-4.4.4$

$ lunch #with no arguments, list the targets.

$ lunch aosp\_arm-eng #a complete build for the emulator, with all debugging enabled.

lunch BUILD-BUILDTYPE

BUILD--指代特定功能组合的codenane

BUILDTYPE

Buildtype Use

user limited access; suited for production

userdebug like "user" but with root access and debuggability; preferred for debugging

eng development configuration with additional debugging tools

## 2.2 运行模拟器(Emulate an Android Device)

    模拟器的位置在编译阶段自动添加到可执行文件的搜索路径中，如果要运行模拟器可以直接输入命令：

$ emulator

# 2.3 [Building Kernels](https://source.android.com/source/building-kernels.html)

    以**panda**kernel为例

1. Figure out the proper kernel

|  |  |  |  |
| --- | --- | --- | --- |
| Device | Binary location | Source location | Build configuration |
| hammerhead | device/lge/hammerhead-kernel | kernel/msm | hammerhead\_defconfig |

    确定设备项目之后，查看git log，Device projects使用如下形式管理 ‘device/<vendor>/<name>’

$ git clone https://android.googlesource.com/device/ti/panda

$ cd panda

$ git log --max-count=1 kernel

2. 确定内核版本

For Nexus 5

$ dd if=zImage-dtb bs=1 skip=$(LC\_ALL=C od -Ad -x -w2 zImage-dtb | grep 8b1f | cut -d ' ' -f1 | head -1) | zgrep -a 'Linux version'

3. 下载源码

# The msm project has the sources for ADP1, ADP2, Nexus One, Nexus 4, Nexus 5, Nexus 6, Nexus 5X, Nexus 6P, and can be used as a starting point for work on Qualcomm MSM chipsets

$ git clone https://android.googlesource.com/kernel/msm.git

# The goldfish project contains the kernel sources for the emulated platforms

$ git clone https://android.googlesource.com/kernel/goldfish.git

4. 下载预编译gcc

$ export PATH=$(pwd)/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin:$PATH

# If Android source tree does not exist, download prebuilt toolchain

$ git clone https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6

5. 编译内核

$ export ARCH=arm

$ export SUBARCH=arm

$ export CROSS\_COMPILE=arm-eabi-

$ cd omap

$ git checkout <commit\_from\_first\_step>

$ make panda\_defconfig

$ make

    编译产生的内核二进制文件位于‘arch/arm/boot/<kernel\_name> ’中，并且可以拷贝至Android源码树只能够用于制作boot镜像。

在运行编译内核命令（如：make bootimage）可以先设置TARGET\_PREBUILT\_KERNEL变量（通过device/common/populate-new-device.sh设置）

$ export TARGET\_PREBUILT\_KERNEL=$your\_kernel\_path/arch/arm/boot/zImage-dtb

    Note: Kernel names differ by device. To locate the correct filename for your kernel, refer to**device/<vendor>/<name>** in the kernel source.

# 3．Flash a Device

## [刷机(Flash a Device)](https://source.android.com/source/building.html#run-it)

1. **fastboot**工具所在路径应该被添加到搜索路径中

2. 设置设备为fastboot模式，手动或者使用命令

$ adb reboot bootloader #place the device in fastboot mode

第一次刷机使用命令‘fastboot flashall -w’很有用，但不是必须的，该命令使用-w参数擦除设备上的/data分区。

$ fastboot flashall -w

## [Running Builds](https://source.android.com/source/running.html)

    Information about building for and running on actual hardware.

### Building fastboot and adb

    fastboot和adb工具可以从常规的编译中编译得到。

$ make fastboot adb

### [Booting into fastboot mode](https://source.android.com/source/running.html#booting-into-fastboot-mode)

     既可以通过组合键手动启动设备进入fastboot模式，也可以使用命令直接重启Android进入bootloader。

|  |  |
| --- | --- |
| hammerhead | Press and hold both Volume Up and Volume Down, then press and hold Power |

adb reboot bootloader

### 解锁(Unlocking the bootloader)

    只有在bootloader允许的情况下才可以刷自定义系统；在fastboot模式下，使用命令解锁bootloader，并需要在用户再屏幕上点击确认，同时删除用户数据。

$ fastboot oem unlock [lock]

### 获取私有二进制文件

    官方二进制文件，用于AOSP分支代码：[Google's Nexus driver page](https://developers.google.com/android/nexus/drivers)

    用于非开源代码编译的硬件专用二进制文件：[Binaries Preview for Nexus Devices](https://developers.google.com/android/nexus/blobs-preview)

    每个二进制文件都是自解压脚本压缩包，在解压后从源码根目录运行自解压的脚本，二进制文件和makefiles文件将安装到源码树的 **vendor/**文件夹中。

    为了确保新安装的二进制包正常工作，需要删除先前编译的输出文件：

$ make clobber

### [为设备选择匹配的编译参数](https://source.android.com/source/running.html#picking-and-building-the-configuration-that-matches-a-device)

|  |  |  |
| --- | --- | --- |
| Device | Code name | Build configuration |
| **Nexus 5 (GSM/LTE)** | **hammerhead** | **aosp\_hammerhead-userdebug** |

**erhead-userdebug**

### 刷机(Flashing a device)

    整个Android系统可以通过一个命令刷机，这个命令在确认被刷入的系统与已经安装的bootloader和radio匹配之后，一起写入boot, recovery和system分区，然后重启系统。 Caution: Flashing a new system image deletes all user data. Be certain to first backup any personal data such as photos.

$ fastboot -w flashall

### 使用Factory image恢复设备至初始出厂状态

Restoring a device to its original factory state

**适用于Nexus设备的官方ROM**

Factory Images for Nexus Devices： <https://developers.google.com/android/nexus/images>

["hammerhead" for Nexus 5 (GSM/LTE)](https://developers.google.com/android/nexus/images#hammerhead)

Binaries for Nexus Devices：<https://developers.google.com/android/nexus/drivers>

[Nexus 5 (GSM/LTE) binaries for Android 6.0.1 (MMB29S)](https://developers.google.com/android/nexus/drivers#hammerhead)

          Broadcom：NFC, Bluetooth, Wi-Fi

          LG： Camera, Sensors, Audio

          Qualcomm：Graphics, GSM, Camera, GPS, Sensors, Media, DSP, USB

    使用system images刷机需要fastboot工具，有两种方法获取该工具，一是从源码编译，二是Android SDK中 **platform-tools/** 目录下存在该工具（使用SDK Manager确保**Android SDK Platform-tools**为最新版本）。

1. 下载镜像文件并解压到安全目录
2. 通过USB连接设备和电脑
3. 设置设备为fastboot模式，使用adb或者按键组合
4. 解锁bootloader
5. 打开一个终端并进入到解压后的系统镜像文件夹
6. 运行**flash-all**脚本，这个脚本安装必要的bootloader, baseband firmware(s), 和 operating system

    脚本执行完毕之后，设备重启，处于安全考虑，最好重新锁定bootloader：

1. 重新进入fastboot模式
2. 运行命令‘fastboot oem lock’