# Android 源码下载和编译

作者:程姚根 联系方式:chengyaogen123@163.com

## 一、开发环境 vmware + ubuntu

1.硬件要求

| | |
|---|---|
| CPU | 64位 |
| 内存 | 至少2G swap分区 6G以上 |
| 硬盘 | 至少60G |

2.虚拟机要求
VMware Workstation 9.0 及 以上版本

3. OS要求
64bit ubuntu 14.04 以及以上版本

**注意：建议实体机直接安装64bit ubuntu系统作为开发环境**

4. android源代码
android 5.0版本

## 二、 安装相关的软件包

1.安装JDK
Android的应用层以及Framework层的一部分都是用Java写的，所以肯定得安装JDK了。在Android 4.4和更老的版本中，使用的是Oracle JDK，而到了Android 5.0以后，Google将其换为OpenJDK（也许是为了专利考虑？），而在Ubuntu中安装OpenJDK是特别容易的：

```
$sudo apt-getupdate
$sudo apt-getinstall openjdk-7-jdk
```

安装后，需要把Java环境设置一下：

```
$sudo update-alternatives--config java
$sudo update-alternatives--config javac
```

如果你想编译老一些的版本，那么就会需要安装更老的OracleJDK了，从Gingerbread （2.3）到Kitkat（4.4）需要的是JDK6，而更老的就要用JDK5了。作为一个只是研究Android源码的人来说，应该不需要考虑这些问题了。而如果是一些因为工作需要而要编译老版本的Android，那么就真的要去找老版本的JDK了。

2.安装android需要的软件包

```
sudo apt-getinstall bisongcc-multilib g++-multilib
sudo apt-get install git gperf libxml2-utils make
sudo apt-get install python-networkx zlib1g-dev:i386 zip gawk
```

这一步在Ubuntu 14.04系统里显得是如此简单，我还很清楚的记得，如果是10.04或者12.04系统，这一步需要安装的包会非常多，大概会有几十个。

## 三、获取android源码

笔者的android 5.0的源码是直接从厂家那边获取的，读者可以自己从谷歌官方网站进行下载，**如下介绍的是如何从谷歌官方网站下载android 4.0源码。**

**1.安装repo**

**首先我们在我们的用户主目录(也称家目录)新建一个子目录，用来存放我们的Android源码**
**mkdir android**

**进入android 目录执行如下命令:**
**curl http://git-repo.googlecode.com/files/repo-1.12 > ./repo**

解释名词:
**(1)curl**
**curl是利用URL语法在命令行方式下工作的开源文件传输工具。**

**(2)repo**
**repo 是一个脚本文件，这个脚本文件主要是用来更方便的通过git下载Android源码。**

**(3)git**
**Git 在Wikipedia上的定义：它是一个免费的、分布式的版本控制工具，或是一个强调了速度快的源代码管理工具。Git最初被Linus Torvalds开发出来用于管理Linux内核的开发。每一个Git的工作目录都是一个完全独立的代码库，并拥有完整的历史记录和版本追踪能力，不依赖 于网络和中心服务器。**

**Git 的出现减轻了许多开发者和开源项目对于管理分支代码的压力，由于对分支的良好控制，更鼓励开发者对自己感兴趣的项目做出贡献。其实许多开源**

项目 包括Linux kernel, Samba, X.org Server, Ruby on Rails，都已经过渡到使用Git作为自己的版本控制工具。对于我们这些喜欢写代码的开发者嘛，有两点最大的好处，我们可以在任何地点(在上班的地铁 上)提交自己的代码和查看代码版本;我们可以开许许多多个分支来实践我们的想法，而合并这些分支的开销几乎可以忽略不计。

2.初始化连接

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

这种方法获取的总是google官方最新的Android源码,如果想过的特定版本的Android源码，可以执行如下命令

**repo init -u https://android.googlesource.com/platform/manifest -b android-4.0.1_r1**

这里的-b 就是用来指定android源码版本的

注意:google的网站不在中国，通过命令行访问的时候会有问题，可以通过如下方法来解决

在/etc/hosts文件中添加如下信息:

**74.125.31.82 www.googlesource.com**

**74.125.31.82 android.googlesource.com**

**203.208.46.172 cache.pack.google.com**

**59.24.3.173 cache.pack.google.com**

这些信息其实就是IP地址和对应的域名,这个IP地址不一定一直是这个，大家可以通过nslookup获取对应域名的IP地址,使用方法如下所示:

```
linux@ubuntu:~$ nslookup www.googlesource.com
Server:         127.0.1.1
Address: 127.0.1.1#53
Non-authoritative answer:
www.googlesource.com    canonical name = googlecode.l.googleusercontent.com.
Name:   googlecode.l.googleusercontent.com
Address: 74.125.31.82
```

3.下载源码

./repo sync

据江湖传言，下载的时候会经常失败，所以大牛们写了一个脚本,内容如下:

load.sh:

```
./repo sync
while [ $? == 1 ]
do
    sleep 3
    ./repo sync
done
```

好了，可以睡觉了，让它慢慢下载吧，Android 4.0源码下载好后差不多10G左右

四、编译源代码

(1)初始化编译环境

$ source build/envsetup.sh

```
linux@ubuntu:~/workdir/android/android-4.0.1$ source build/envsetup.sh
including device/samsung/maguro/vendorsetup.sh
including device/samsung/tuna/vendorsetup.sh
including device/ti/panda/vendorsetup.sh
including sdk/bash_completion/adb.bash
```

(2)选择一个目标平台

```
linux@ubuntu:~/workdir/android/android-4.0.1$ lunch

You're building on Linux

Lunch menu... pick a combo:
     1. full-eng
     2. full_x86-eng
     3. vbox_x86-eng
     4. full_maguro-userdebug
     5. full_tuna-userdebug
     6. full_panda-eng
```

```
Which would you like? [full-eng]
```

在这里选择默认就可以了。接下来输入make 就可以了,如果你的CPU比较多，也可以make -jn。这里的n表示同时

启动n个线程参加编译。我的CPU是双核4线程，我一般使用make -j4。

**注意:编译的过程可能会出现很多错误，大家直接把错误信息进行baidu和google就可以，大牛们已经给出相应的解决**

**l办法了。**

四、通过模拟器运行我们的Android

Android系统编译好后，在out/target/product/generic目录下回看到一些.img文件。

```
linux@ubuntu: ~/workdir/android/android-4.0.1/out/target/product/generic
linux@ubuntu:~/workdir/android/android-4.0.1/out/target/product/generic$ ls
android-info.txt  hardware-qemu.ini         ramdisk.img  system.img
clean_steps.mk    installed-files.txt       root         userdata.img
data              obj                       symbols      userdata-qemu.img
dex_bootjars      previous_build_config.mk  system
```

ramdisk.img :

一个分区影像文件，它会在kernel 启动的时候，以只读的方式被 mount ， 这个文件中只是包含了 /init 以及一些配置文件，

这个ramdisk 被用来调用init，以及把真正的root file system mount 起来。其实ramdisk.**img**的内容就是out/target/product/generic/root

目录的压缩而已。

```
linux@ubuntu:~/workdir/android/android-4.0.1/out/target/product/generic/root$ ls
data          dev   init.goldfish.rc  proc  sys      ueventd.goldfish.rc
default.prop  init  init.rc           sbin  system   ueventd.rc
```

system.**img:**

它包含了整个系统，**android** 的framework，application 等等，会被挂接到 "/" 上，包含了系统中所有的二进制文件system.**img**是out/target.product/generic/system目录的一个映射，类似与根文件系统的映像，放着**android** 的应用程序、配置文件和字体等。

```
linux@ubuntu: ~/workdir/android/android-4.0.1/out/target/product/generic/system
linux@ubuntu:~/workdir/android/android-4.0.1/out/target/product/generic/system$
ls
app  bin  build.prop  etc  fonts  framework  lib  media  tts  usr  vendor  xbin
```

userdata.**img:**

将会被挂接到 /data 下，包含了所有应用相关的配置文件，以及用户相关的数据 。

通过模拟器运行的时候，需要很多参数，为了简便期间，我写了一个shell脚本，大家可以根据需要自己修改:

```
linux@ubuntu: ~/workdir/android/android-4.0.1
#!/bin/bash
PWD_PATH=`pwd`
KERNEL_PATH=$PWD_PATH/prebuilt/android-arm/kernel/kernel-qemu-armv7
EMULATOR_PATH=$PWD_PATH/out/host/linux-x86/bin
export PATH=$EMULATOR_PATH:$PATH
export ANDROID_PRODUCT_OUT=$PWD_PATH/out/target/product/generic
emulator -kernel $KERNEL_PATH -partition-size 1024

```

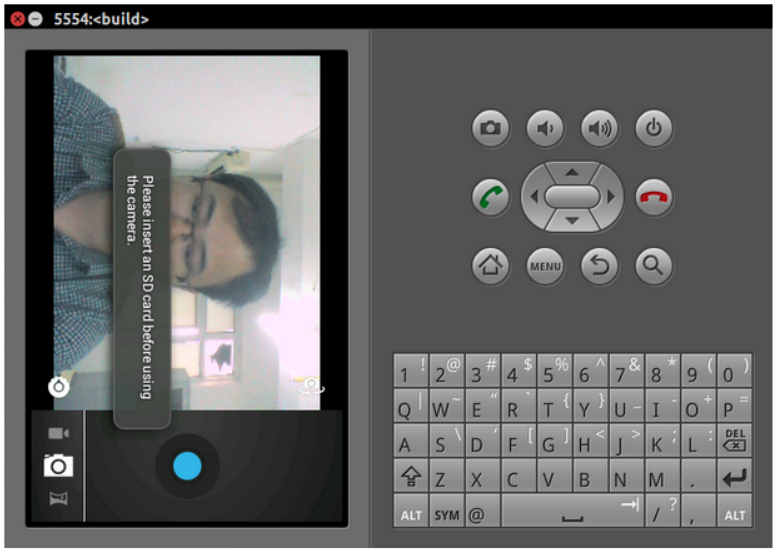
简单解释:

emulator在out/host/linux-x86/bin目录下，所以要把它添加到PATH环境变量中。

-kernel 是指定Android系统的内核，这里我们没有内核，使用Android系统自带的内核kernel-qemu-armv7

ANDROID_PRODUCT_OUT环境变量一定要配置正确，模拟器需要根据它找到那些img文件

好了，直接运行脚本文件，就可以看到我们的Android系统了,效果如下:




五、下载Android的 Linux 内核源码

Google的官方已经给出了一些已经配置好的Android Linux kernel。但是都是针对Google自己的产品,如果想要自己开发产品，要么自己配置(难度较大),要么找对应的芯片厂家索要。这里我们只是在模拟器上运行Android系统，Google的模拟器模拟的是ARM926ej-S的Goldfish处理器，我们只需要下载Goldfish处理器对应的源码即可。

(1)新建一个子目录来存放下载的Linux for Android的源码

mkdir qemu-kernel

(2)进入qemu-kernel目录，用git工具克隆Google官方的goldfish 工程


下载的需要一段时间，请耐心等待，下载好后再你的当前目录会看到一个goldfish子目录,进入这个子目录输入git branch -a 可以看到克隆的Goldfish工程中不同版本的Linux 内核源码。

```
linux@ubuntu:~/workdir/android/qemu-kernel/goldfish$ git branch -a
* (分离自 origin/android-goldfish-2.6.29)
  master
  remotes/origin/HEAD -> origin/master
  remotes/origin/android-3.10
  remotes/origin/android-goldfish-2.6.29
  remotes/origin/android-goldfish-3.10
  remotes/origin/android-goldfish-3.4
  remotes/origin/linux-goldfish-3.0-wip
  remotes/origin/master
```

（3）通过git  checkout 我们想要的分支,输入如下命令

git checkout remotes/origin/android-goldfish-2.6.29

(4)编译Linux kernel  for Android

在编译之前Linux kernel之前，需要指定对应的编译工具链和平台的配置的文件。

A.编译工具链

Android的源代码中已经有了，大家可以在Android源代码的prebuilt/linux-x86/toolchain目录下找他们

```
linux@ubuntu:~/workdir/android/android-4.0.1/prebuilt/linux-x86/toolchain$ ls
arm-eabi-4.2.1  arm-eabi-4.4.3              i686-linux-glibc2.7-4.4.3
arm-eabi-4.3.1  arm-linux-androideabi-4.4.x  i686-unknown-linux-gnu-4.2.1
arm-eabi-4.4.0  i686-android-linux-4.4.3     sh-4.3.3
```

这里我选择的是arm-eabi-4.4.3。

B.配置文件

模拟器模拟的是goldfilsh处理器，所以这里使用 goldfish_armv7_defconfig,大家可以在goldfish/arch/arm/configs找到它

根据Google官方的套路，这里写了一个简单的脚本(build.sh)进行配置和编译

```
linux@ubuntu: ~/workdir/android/qemu-kernel/goldfish
1 export ARCH=arm
2 export SUBARCH=arm
3 export ANDROID_TOOLCHAIN=/home/linux/workdir/android/android-4.0.1/\
4 prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin
5 export PATH=$PATH:$ANDROID_TOOLCHAIN
6 export CROSS_COMPILE=arm-eabi-
7 make goldfish_armv7_defconfig
8 make -j4
```

Android工具链的路径

根据指定的平台，配置内核

成功编译后再arch/arm/boot目录下可以看到编译好的内核。

```
linux@ubuntu:~/workdir/android/qemu-kernel/goldfish$ ls arch/arm/boot/
bootp  compressed  Image  install.sh  Makefile  zImage
```

六、使用编译好的Linux 内核运行Android系统

在前面我们通过模拟器运行Android系统的时候，写过一个脚本(start.sh)，当时我们使用的是Android源码自带的linux kernel,现在用我们自己编译好的Linux kernel。修改配置文件如下:

```
linux@ubuntu: ~/workdir/android/android-4.0.1
1 #!/bin/bash
2 PWD_PATH=`pwd`
3
4 #KERNEL_PATH=$PWD_PATH/prebuild/android-arm/kernel/kernel-qemu-armv7
5 KERNEL_PATH=/home/linux/workdir/android/qemu-kernel/goldfish\
6 /arch/arm/boot/zImage
7
```

Android源码自带的Linux kernel路径

自己编译好的Linux kernel路径

```
 8 EMULATOR_PATH=$PWD_PATH/out/host/linux-x86/bin
 9 export PATH=$EMULATOR_PATH:$PATH
10 export ANDROID_PRODUCT_OUT=$PWD_PATH/out/target/product/generic
11 emulator -kernel $KERNEL_PATH -partition-size 1024&
```

修改完后，通过source start.sh运行我们的脚本，可以看到模拟器成功运行。

怎么知道，模拟器使用的一定是我们编译好的Linux kernel呢? 我们可以通过adb连接模拟器，查看Linux kernel版本


好了，到这里我们已经将Android系统和Linux kernel编译好，在模拟器上运行起来了，接下来我们将会看一些细节东西。