

五 Android 系统开发环境搭建以及编译

本章节中将为您详细介绍 uboot、Linux3.0.15 和 linux-3.5 内核、Android4.0.3 和 Android4.2.2 编译环境的搭建以及编译。

注意:从一开始，我们就是基于 Ubuntu12.04.2 平台做开发，所有的配置和编译脚本也是基于此平台，我们没有在其它平台上测试过。如果你对 Linux 和 Android 开发很熟悉，相信你会根据错误提示逐步找到原因并解决，错误提示一般是选用的平台缺少了某些库文件或者工具等原因造成的；否则，**我们建议初学者使用和我们一致的平台**，即 Ubuntu12.04.2，你可以在我们的网盘下载 Ubuntu12.04.2 的镜像，安装的时候请务必参考我们手册提供的步骤，这是我们经过严格测试的，以免遗漏一些开发时所需要的组件。

Linux 的发行版本众多，我们无法为此一一编写文档，敬请原谅。

Uboot、Kernel 以及 Android 的编译环境看似复杂，用户只要抓住以下几个要点就可以了：

第一：Uboot、Kernel 编译器。编译器在光盘上都有提供，在需要使用的步骤中，会说明编译器在光盘中的位置。

第二：设置环境变量。Uboot、Kernel 编译器的环境变量设置后，编译的时候，系统才能找到编译器。

第三：Android 文件系统的编译器。编译器需要使用 Ubuntu 系统自带的 gcc 编译器，但是版本不对，所以需要降低版本。迅为将这个过程编写了成几个简单的命令，用户只需要挨个执行命令就可以了。

第四：库文件。搭建过程中会给通过执行简单的脚本命令来安装库文件，复杂的步骤变的简单有效。

另外，如果用户想了解编译环境具体是怎么搭建起来的，可以利用我们提供的脚本文件来学习。

5.1 Android4.0.3 编译环境的两种搭建方式

迅为电子给用户两种搭建编译环境的方式，一种方法是用户安装虚拟机，然后安装基础的 Ubuntu12.04.2 系统，利用我们提供工具和详细的使用步骤，搭建编译环境；另外一种方法是用户安装虚拟机，然后直接加载我们“搭建好的 Ubuntu 镜像”，用户只需要修改一下编译器的环境变量，就可以直接用来编译源码。

5.1.1 使用已经搭建好的镜像

使用已经搭建编译环境的镜像，用户只需要安装虚拟机“Vmware_Workstaion_wm”，然后用虚拟机加载搭建好环境的 Ubuntu 镜像。

“搭建好的镜像”提供网址供大家下载，加载方法参考“3.2 安装虚拟机以及 Ubuntu12.04.2 等软件”。

5.1.2 自己搭建环境

另一种方法是自己安装虚拟机，安装 Ubuntu12.04.2 系统，搭建环境。大家可以参考 5.2 小节自己搭建环境，网盘里面提供了所有需要用到的软件。

5.2 搭建环境

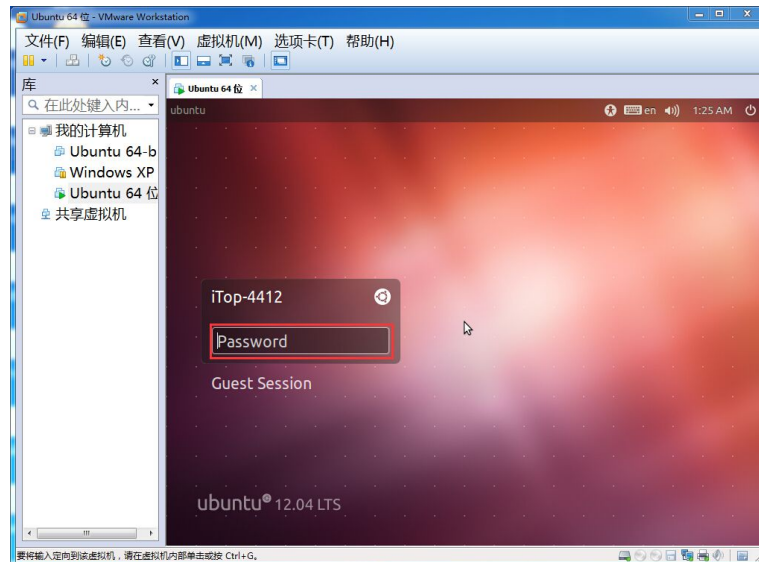
本节的主要内容是，详细讲解如何搭建编译环境。

这里需要注意的是，搭建过程中用到的各类软件，都需要和手册提到的版本保持一致，如果使用的是“搭建好的镜像”，则可以跳过这一节，但是编译的时候要针对性的设置一下环境变量。

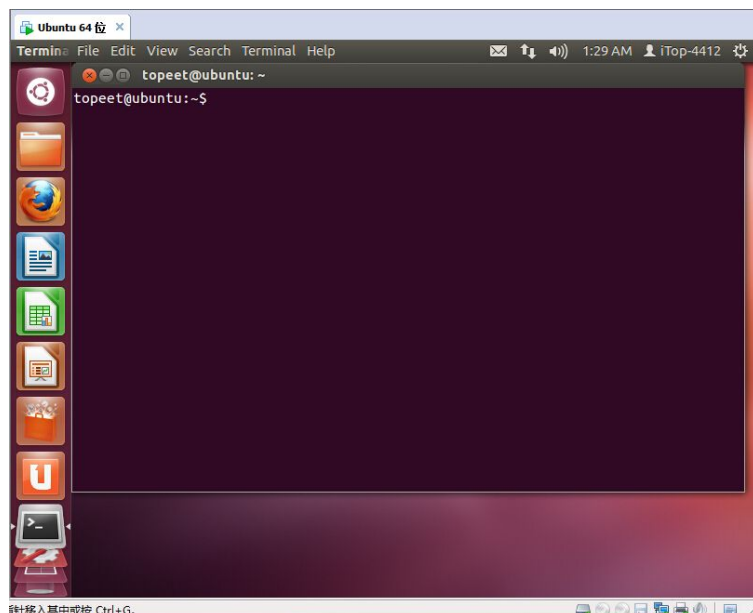
5.2.1 安装基本软件

首先安装虚拟机“Vmware_Workstaion_wm”，然后使用虚拟机安装“Ubuntu12.04.2 初始系统”。

Ubuntu 的安装方法可以参考 3.2 小节来安装 Ubuntu12.04.2 原始系统，如下图所示，Ubuntu 初始系统安装完成。



输入密码“topeet”，登陆 Ubuntu，键盘上按“Ctrl+Alt+t”，弹出 Ubuntu 的控制台。如下图所示。

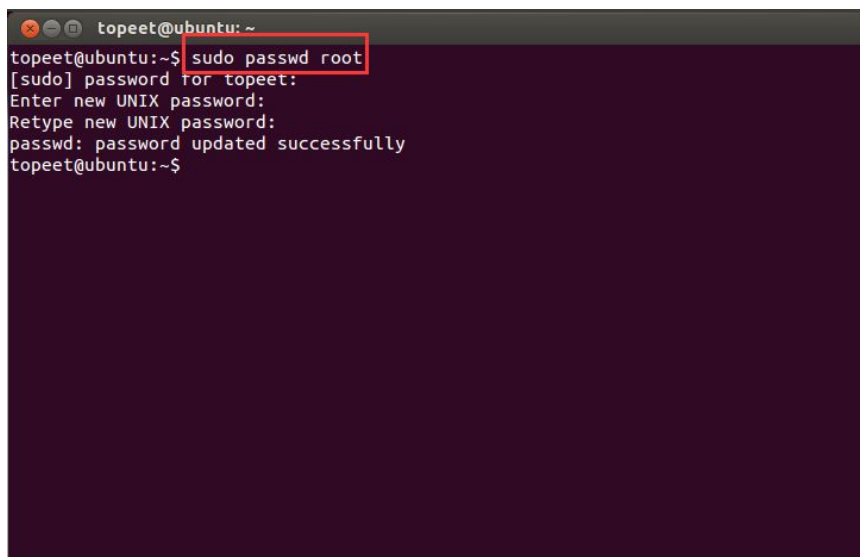


安装完成后进入 Ubuntu 的终端，激活 root 用户，具体操作如下。

在 Ubuntu 命令行中，执行命令 “sudo passwd root”。

接着在 Ubuntu 的终端输入安装时的密码和新密码，Ubuntu 系统中密码默认是隐藏的。

如下图所示。

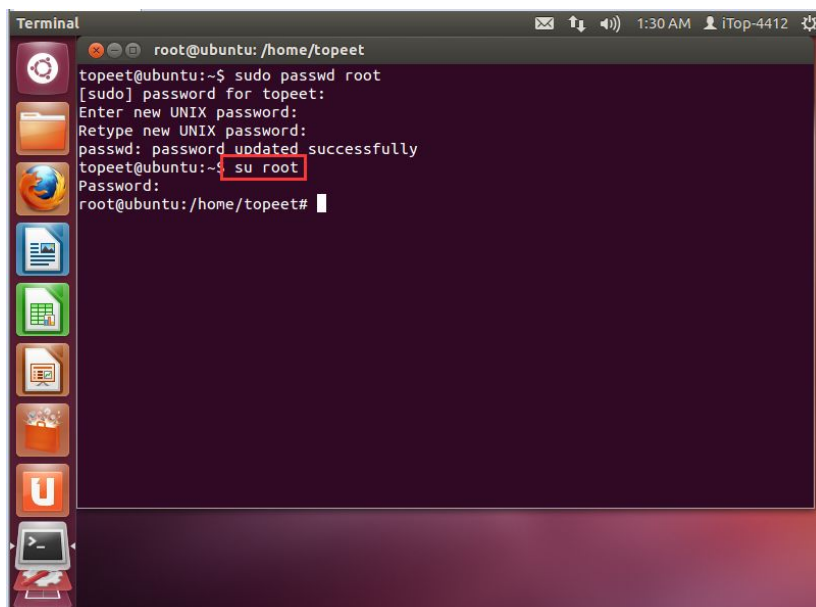


接着登录 root 用户，具体操作如下。

在 Ubuntu 命令行中，执行命令 “su root”。

接着输入密码，就可以登陆 root 用户。

后面所有的操作都需要在 root 用户下面进行操作，如下图所示。



参考 “3.2.4 虚拟机 VMware-workstation8.0.3 联网以及基本设置” 联网。

参考 “3.3.4.2 修改数据源地址” 将 Ubuntu 数据源地址修改为国内地址。

参考 “3.3.4.3 “apt-get update” 命令” 更新数据源。

然后在 Ubuntu 安装软件 vim 和 ssh，在 Ubuntu 命令行中，执行命令 “apt-get install vim” 和 “apt-get install ssh”。

上面安装的 ssh 软件，可以很方便在主机和虚拟机上传递文件，也可以通过远程终端控制 Ubuntu 系统，这里推荐给大家使用。ssh 软件的使用参考 3.3 小节。

5.2.2 安装编译组件

5.2.2.1 交叉编译工具

编译的时候需要用到交叉编译工具，我们提供的交叉编译工具是用户光盘 “02_编译器以及烧写工具” → “arm 交叉编译器” 文件夹中的压缩包 “arm-2009q3.tar.bz2”。

名称	修改日期	类型	大小
 arm-2009q3.tar.bz2	2015/7/4 ...	360压缩	82,40...

使用 SSH 工具将交叉编译工具拷贝到 Ubuntu12.04.2 系统的文件夹 “usr” --

> “local” --> “arm” 中，local 下默认没有 arm 文件夹，可以新建一个。如下图所示。

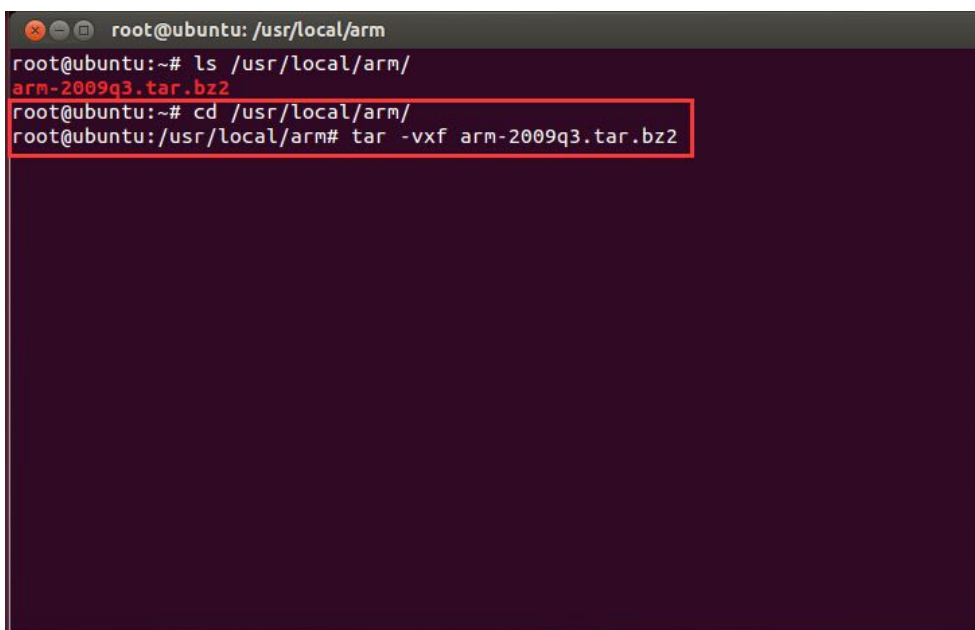
```
root@ubuntu: ~  
root@ubuntu:~# mkdir /usr/local/arm  
root@ubuntu:~# ls /usr/local/  
arm bin etc games include lib man sbin share src  
root@ubuntu:~#
```

拷贝编译器之后如下图所示。

```
root@ubuntu: ~  
root@ubuntu:~# mkdir /usr/local/arm  
root@ubuntu:~# ls /usr/local/  
arm bin etc games include lib man sbin share src  
root@ubuntu:~# ls /usr/local/arm/  
arm-2009q3.tar.bz2  
root@ubuntu:~#
```

然后在 Ubuntu 系统中将压缩包解压到当前目录下。

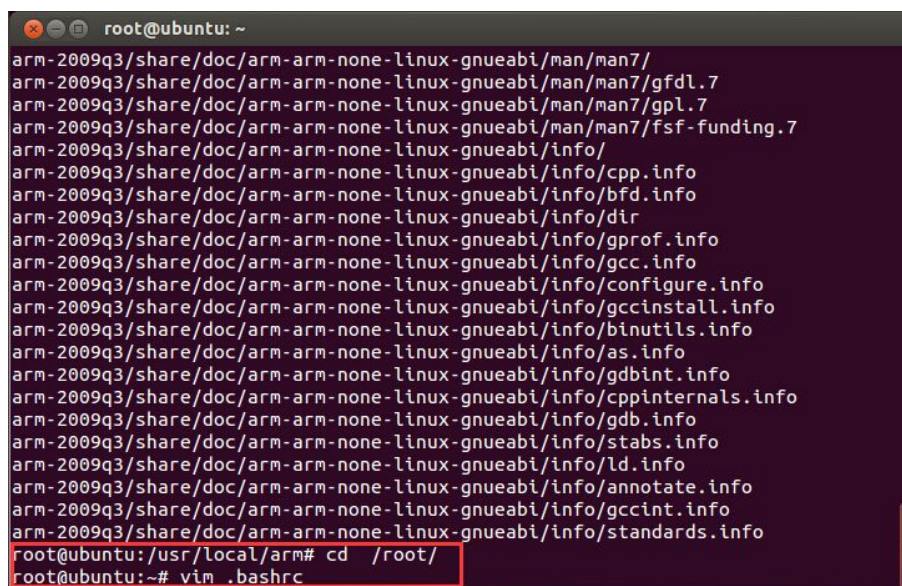
使用命令 “cd /usr/local/arm/” 进入/usr/local/arm 文件夹，然后使用解压命令 “tar -vxf arm-2009q3.tar.bz2” 解压压缩包,如下图所示。



```
root@ubuntu: /usr/local/arm
root@ubuntu:~# ls /usr/local/arm/
arm-2009q3.tar.bz2
root@ubuntu:~# cd /usr/local/arm/
root@ubuntu:/usr/local/arm# tar -vxf arm-2009q3.tar.bz2
```

5.2.2.2 修改交叉编译工具的路径（修改环境变量）

修改交叉编译工具路径，需要修改环境变量。在 Ubuntu 命令行中，执行命令 “cd /root” 和 “vim .bashrc”，打开环境变量文件 “.bashrc”，如下图所示。

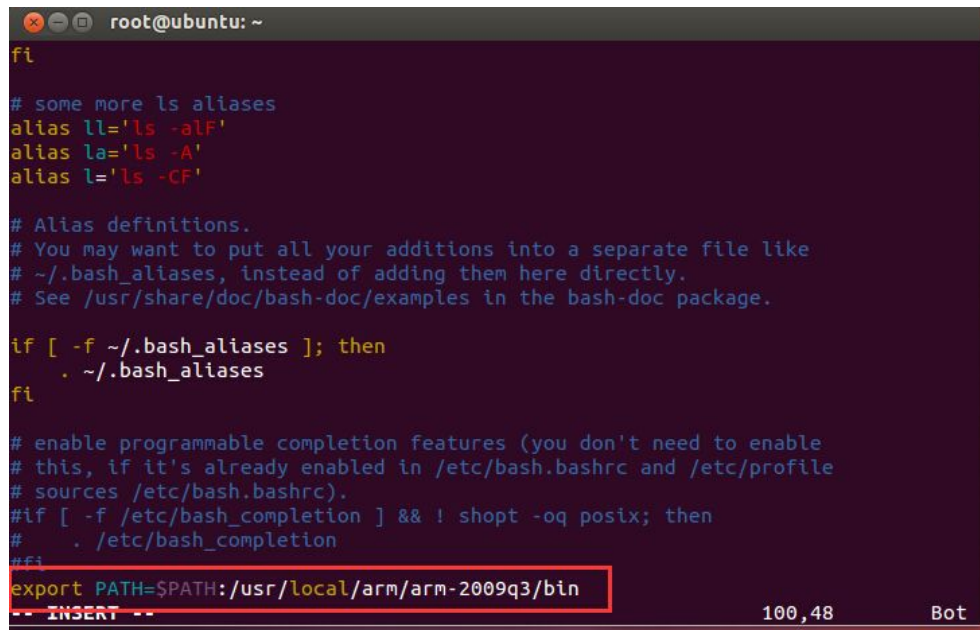


```
root@ubuntu: ~
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/gfdl.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/gpl.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/man/man7/fsf-funding.7
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/cpp.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/bfd.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/dir
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gprof.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gcc.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/configure.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccinstall.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/binutils.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/as.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdbint.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/cppinternals.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdb.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/stabs.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/ld.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/annotate.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccint.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info
root@ubuntu:/usr/local/arm# cd /root/
root@ubuntu:~# vim .bashrc
```

然后在 “.bashrc” 文件中的最后一行添加如下信息：

“export PATH=\$PATH:/usr/local/arm/arm-2009q3/bin”

如下图所示。



```
root@ubuntu: ~
fi
# some more ls aliases
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'

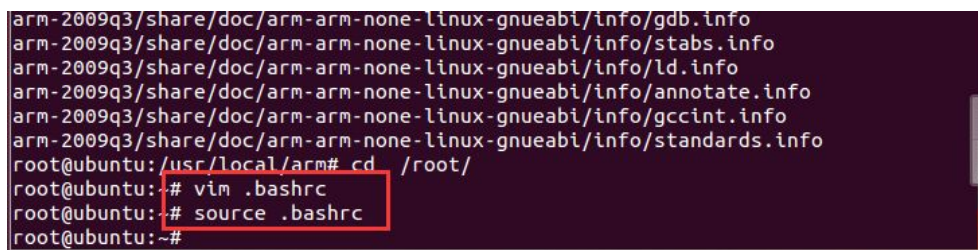
# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi
export PATH=$PATH:/usr/local/arm/arm-2009q3/bin
-- INSERT --
```

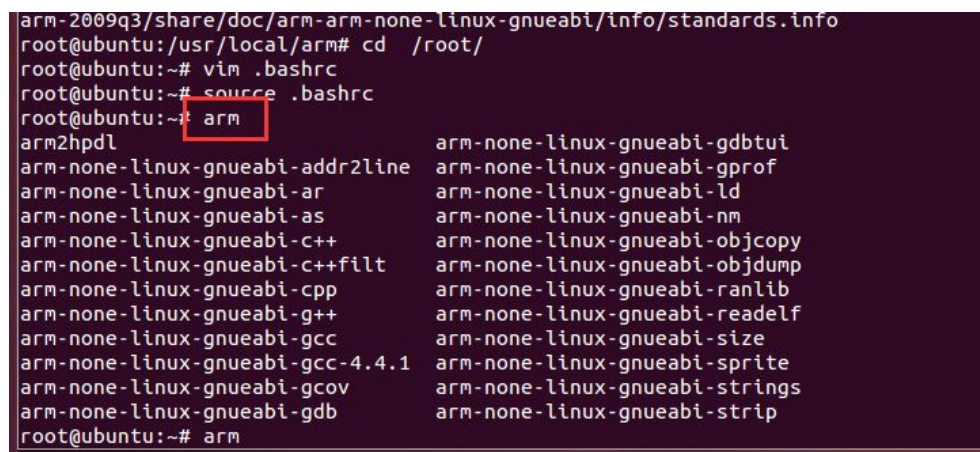
修改完成后保存退出。

执行下列命令，更新环境变量“source .bashrc”的命令，如下图所示。



```
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gdb.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/stabs.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/ld.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/annotate.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/gccint.info
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info
root@ubuntu:/usr/local/arm# cd /root/
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~#
```

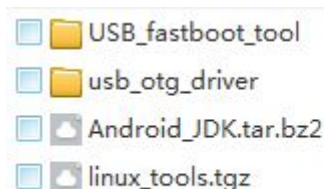
最后，在 Ubuntu 命令行中输入命令“arm”，然后按 TAB 键，如果在命令行中能够看到 arm 编译器的信息，就表明交叉编译工具安装成功。如下图所示。



```
arm-2009q3/share/doc/arm-arm-none-linux-gnueabi/info/standards.info
root@ubuntu:/usr/local/arm# cd /root/
root@ubuntu:~# vim .bashrc
root@ubuntu:~# source .bashrc
root@ubuntu:~# arm
arm2hpd1
arm-none-linux-gnueabi-addr2line
arm-none-linux-gnueabi-ar
arm-none-linux-gnueabi-as
arm-none-linux-gnueabi-c++
arm-none-linux-gnueabi-c++filt
arm-none-linux-gnueabi-cpp
arm-none-linux-gnueabi-g++
arm-none-linux-gnueabi-gcc
arm-none-linux-gnueabi-gcc-4.4.1
arm-none-linux-gnueabi-gcov
arm-none-linux-gnueabi-gdb
arm-none-linux-gnueabi-gdbtui
arm-none-linux-gnueabi-gprof
arm-none-linux-gnueabi-ld
arm-none-linux-gnueabi-nm
arm-none-linux-gnueabi-objcopy
arm-none-linux-gnueabi-objdump
arm-none-linux-gnueabi-ranlib
arm-none-linux-gnueabi-readelf
arm-none-linux-gnueabi-size
arm-none-linux-gnueabi-sprite
arm-none-linux-gnueabi-strings
arm-none-linux-gnueabi-strip
root@ubuntu:~# arm
```


5.2.3 安装库文件、JDK 以及降低 GCC 版本

为了方便用户，将库文件和 JDK 的安装命令制作成了脚本文件，用户只要执行两个脚本就可以安装库文件和 JDK。这两个脚本在用户光盘“02_编译器以及烧写工具”→“tools”文件夹下的压缩包“Android_JDK.tar.bz2”中，如下图所示。



用户将压缩包拷贝到 Ubuntu 系统中，解压压缩包会生成文件夹“Android_JDK”，如下图所示。

```
root@ubuntu:~# ls /home/topeet/
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
Desktop              Downloads  Music             Public   Videos
root@ubuntu:~# cd /home/topeet/
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet#
root@ubuntu:/home/topeet# ls
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
Desktop              Downloads  Music             Public   Videos
root@ubuntu:/home/topeet# tar -vxf Android_JDK.tar.bz2
Android_JDK/
Android_JDK/.DS_Store
Android_JDK/apt-source/
Android_JDK/apt-source/sources.list.163
Android_JDK/apt-source/sources.list.cn
Android_JDK/install-devel-packages.sh
Android_JDK/jdk6/
Android_JDK/jdk6/install-sun-java6.sh
Android_JDK/jdk6/jdk-6u43-linux-x64.bin
Android_JDK/update_gcc.txt
root@ubuntu:/home/topeet# ls
Android_JDK  Desktop  Downloads  Music  Public  Videos
Android_JDK.tar.bz2  Documents  examples.desktop  Pictures  Templates
root@ubuntu:/home/topeet#
```

5.2.3.1 安装库文件和 JDK

使用 cd 命令，进入解压出来的“Android_JDK” --> “jdk6” 文件夹，运行脚本文件“install-sun-java6.sh”。

需要注意的是，这条命令执行完毕可能会耗时 15 分钟以上。

如下图所示，执行 “./install-sun-java6.sh” 脚本。

```
Android_JDK/apr-source/sources.lst.cn
Android_JDK/install-devel-packages.sh
Android_JDK/jdk6/
Android_JDK/jdk6/install-sun-java6.sh
Android_JDK/jdk6/jdk-6u43-linux-x64.bin
Android_JDK/update_gcc.txt
root@ubuntu:/home/topeet# ls
Android_JDK Desktop Downloads Music Public Videos
tar.bz2 Documents examples.desktop Pictures Templates
root@ubuntu:/home/topeet# cd Android_JDK
root@ubuntu:/home/topeet/Android_JDK# cd jdk6/
root@ubuntu:/home/topeet/Android_JDK/jdk6# ./install-sun-java6.sh
```

执行上面的命令的时候，根据提示输入 “回车” 命令。升级完成之后如下图所示。

```
root@ubuntu:/home/topeet/Android_JDK/jdk6
There is only one alternative in link group java: /usr/lib/jvm/jdk1.6.0_43/bin/j
ava
Nothing to configure.
There is only one alternative in link group javac: /usr/lib/jvm/jdk1.6.0_43/bin/
javac
Nothing to configure.
There is only one alternative in link group javadoc: /usr/lib/jvm/jdk1.6.0_43/bi
n/javadoc
Nothing to configure.
There is only one alternative in link group mozilla-javaplugin.so: /usr/lib/jvm/
jdk1.6.0_43/jre/lib/amd64/libnpjp2.so
Nothing to configure.
There is only one alternative in link group javaws: /usr/lib/jvm/jdk1.6.0_43/bin
/javaws
Nothing to configure.
There is only one alternative in link group jar: /usr/lib/jvm/jdk1.6.0_43/bin/ja
r
Nothing to configure.
java version "1.6.0_43"
Java(TM) SE Runtime Environment (build 1.6.0_43-b01)
Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)
javac 1.6.0_43
root@ubuntu:/home/topeet/Android_JDK/jdk6#
root@ubuntu:/home/topeet/Android_JDK/jdk6#
```

进入解压出来的文件夹 “Android_JDK” 中运行脚本 “install-devel-packages.sh” ，安
装库文件。在 Ubuntu 命令行中，执行命令 “./install-devel-packages.sh” ，需要注意的
是，这条命令可能会耗时 40 分钟以上，如下图所示。

```
There is only one alternative in link group jar: /usr/lib/jvm/jdk1.6.0_43/bin/ja
r
Nothing to configure.
java version "1.6.0_43"
Java(TM) SE Runtime Environment (build 1.6.0_43-b01)
Java HotSpot(TM) 64-Bit Server VM (build 20.14-b01, mixed mode)
javac 1.6.0_43
root@ubuntu:/home/topeet/Android_JDK/jdk6#
root@ubuntu:/home/topeet/Android_JDK/jdk6# cd ../
root@ubuntu:/home/topeet/Android_JDK# ./install-devel-packages.sh
```

上面命令执行的时候，需要根据提示输入 “Y” 。

```
x11proto-damage-dev x11proto-gl-dev x11proto-glx-dev x11proto-gt-dev
x11proto-input-dev x11proto-kb-dev x11proto-xext-dev
x11proto-xf86vidmode-dev xorg-sgml-doctools xsltproc xtrans-dev zlib1g:i386
zlib1g-dev:i386
The following packages will be upgraded:
  gnupg libc-bin libc-dev-bin libc6 libc6-dev libdrm-intel1 libdrm-nouveau1a
  libdrm-nouveau2 libdrm-radeon1 libdrm2 libgl1-mesa-dri-lts-quantal
  libgl1-mesa-glx-lts-quantal libglapi-mesa-lts-quantal libllvm3.1
  libpciaccess0 libx11-6 libx11-xcb1 libxcb-glx0 libxcb1 libxext6 libxf86vm1
  linux-libc-dev
23 upgraded, 111 newly installed, 0 to remove and 505 not upgraded.
Need to get 115 MB of archives.
After this operation, 380 MB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

然后安装过程中，还会提示输入“y”，如下图所示。

```
==> Executing: 'apt-get install vim dos2unix minicom gawk'
Reading package lists... Done
Building dependency tree
Reading state information... Done
vim is already the newest version.
The following extra packages will be installed:
  libsigsegv2 lrzsz
The following NEW packages will be installed:
  dos2unix gawk libsigsegv2 lrzsz minicom
0 upgraded, 5 newly installed, 0 to remove and 505 not upgraded.
Need to get 947 kB of archives.
After this operation, 3,170 kB of additional disk space will be used.
Do you want to continue [Y/n]? █
```

如下图所示，安装完毕。

```
Unpacking lrzsz (from .../lrzsz_0.12.21-5_amd64.deb) ...
Selecting previously unselected package minicom.
Unpacking minicom (from .../minicom_2.5-2_amd64.deb) ...
Selecting previously unselected package dos2unix.
Unpacking dos2unix (from .../dos2unix_5.3.1-1_amd64.deb) ...
Processing triggers for man-db ...
Setting up gawk (1:3.1.8+dfsg-0.1ubuntu1) ...
Setting up lrzsz (0.12.21-5) ...
Setting up minicom (2.5-2) ...
Setting up dos2unix (5.3.1-1) ...
root@ubuntu:/home/topeet/Android_JDK# █
```

这里需要注意的是，上面这个脚本执行完毕的时候，注意一下有些库文件是不是提示没有安装。如果发现有库文件没有安装，有可能是网络不好或者下载源丢失。这个时候用户使用一下更新下载源的命令“apt-get update”，然后再执行一下上面的两个脚本。

例如，如下图所示，再次运行“./install-devel-packages.sh”之后，提示没有无法安装的库和软件，那么表明已经安装完全了。

```
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.
==> Executing: 'apt-get install vim dos2unix minicom gawk'
Reading package lists... Done
Building dependency tree
Reading state information... Done
gawk is already the newest version.
dos2unix is already the newest version.
minicom is already the newest version.
vim is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.
root@ubuntu:/home/topeet/Android_JDK# █
```


5.2.3.2 降低 GCC 版本

使用 Ubuntu 编译 Android 的时候需要用到 Ubuntu 系统自带的 GCC4.4.7 编译器，但是安装的 Ubuntu12.04.2 版本，它的 GCC 版本过高，所以这里需将要 GCC 编译的版本降低到 4.4.7。

进入前面解压的文件夹 “Android_JDK” 中，会看到一个文本 “update_gcc.txt”，打开文本 “update_gcc.txt” 后会看到里面有 8 条命令，这 8 条命令需要在 Ubuntu 命令行中依次执行。如下图所示，使用命令打开 “update_gcc.txt” 文件。

```
vim is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.  
root@ubuntu:/home/topeet/Android_JDK# ls  
apt-source install-devel-packages.sh make update_gcc.txt  
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
```

如下图所示，有 8 条命令。

```
root@ubuntu:/home/topeet/Android_JDK  
1.apt-get install gcc-4.4 g++-4.4 g++-4.4-multilib gcc-4.4-multilib  
2.update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.4 100  
3.update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-4.6 50  
4.update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.4 100  
5.update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-4.6 50  
6.update-alternatives --install /usr/bin/cpp cpp-bin /usr/bin/cpp-4.4 100  
7.update-alternatives --install /usr/bin/cpp cpp-bin /usr/bin/cpp-4.6 50  
8.gcc -v  
~  
~  
"update_gcc.txt" [noeol][dos] 15L, 546C 1,1 All
```

例如执行第一条命令。

```
gcc is already the newest version.  
dos2unix is already the newest version.  
minicon is already the newest version.  
vim is already the newest version.  
0 upgraded, 0 newly installed, 0 to remove and 505 not upgraded.  
root@ubuntu:/home/topeet/Android_JDK# ls  
apt-source install-devel-packages.sh make update_gcc.txt  
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt  
root@ubuntu:/home/topeet/Android_JDK# apt-get install gcc-4.4 g++-4.4 g++-4.4-m  
ltilib gcc-4.4-multilib
```

根据提示输入 “y”，如下图所示。

```
cpp-4.4 gcc-4.4-base libstdc++6-4.4-dev
Suggested packages:
gcc-4.4-locales gcc-4.4-doc libstdc++6-4.4-dbg lib32stdc++6-4.4-dbg
libmudflap0-4.4-dev libgcc1-dbg libgomp1-dbg libmudflap0-dbg libclog-ppl0
libppl-c2 libppl7 lib32mudflap0 libstdc++6-4.4-doc
The following NEW packages will be installed:
cpp-4.4 g++-4.4 g++-4.4-multilib gcc-4.4 gcc-4.4-base gcc-4.4-multilib
libstdc++6-4.4-dev
0 upgraded, 7 newly installed, 0 to remove and 505 not upgraded.
Need to get 17.4 MB of archives.
After this operation, 44.7 MB of additional disk space will be used.
Do you want to continue [Y/n]?
```

其余几条命令，如下图所示，执行起来很快。

```
root@ubuntu: /home/topeet/Android_JDK
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/g++
g++ /usr/bin/g++-4.4 100
update-alternatives: using /usr/bin/g++-4.4 to provide /usr/bin/g++ (g++) in aut
o mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/g++
g++ /usr/bin/g++-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/gcc
gcc /usr/bin/gcc-4.4 100
update-alternatives: using /usr/bin/gcc-4.4 to provide /usr/bin/gcc (gcc) in aut
o mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/gcc
gcc /usr/bin/gcc-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
LibreOffice Impress topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.4 100
update-alternatives: using /usr/bin/cpp-4.4 to provide /usr/bin/cpp (cpp-bin) in
auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.6 50
root@ubuntu:/home/topeet/Android_JDK#
```

在执行了这 8 条命令之后，Ubuntu 系统就将 gcc 的版本降低到 4.4.7。

如下图所示，使用命令 “gcc -v”，可以看到 gcc 的版本为 4.4.7 了。

```
root@ubuntu: /home/topeet/Android_JDK
cpp-bin /usr/bin/cpp-4.4 100
update-alternatives: using /usr/bin/cpp-4.4 to provide /usr/bin/cpp (cpp-bin) in
auto mode.
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK# update-alternatives --install /usr/bin/cpp
cpp-bin /usr/bin/cpp-4.6 50
root@ubuntu:/home/topeet/Android_JDK# vim update_gcc.txt
root@ubuntu:/home/topeet/Android_JDK#
root@ubuntu:/home/topeet/Android_JDK# gcc -v
Using built-in specs.
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.4.7-1ubu
ntu2' --with-bugurl=file:///usr/share/doc/gcc-4.4/README.Bugs --enable-languages
=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.4 --enable-shared
--enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-incl
uded-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.4
--libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-li
bstdc++-debug --enable-objc-gc --disable-werror --with-arch-32=i686 --with-tune=
generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-g
nu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.4.7 (Ubuntu/Linaro 4.4.7-1ubuntu2)
```

需要注意的是，在执行这 8 条命令时，只有第一条命令会耗时 10 分钟左右，其它的都会很快完成，而且命令一定要依次执行，不能有遗漏。

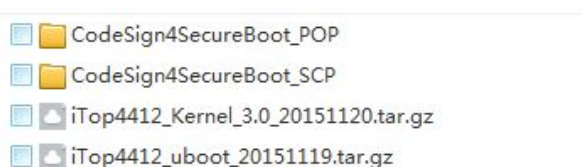
5.3 Android4.0.3 镜像的编译

无论什么文件系统都需要和 linux 内核以及 uboot 对应，所以在本章节先介绍 Android4.0.3 文件系统对应的 uboot 以及 kernel 编译，再介绍 Android4.0.3 文件系统的编译。

5.3.1 uboot 的编译

5.3.1.1 源码目录

Uboot 源码在光盘 “06_源码_uboot 和 kernel” 目录下，如下图所示。



5.3.1.2 编译器

如下图所示，编译器是使用的的光盘目录下，“02_编译器以及烧写工具” → “arm 交叉编译器” 下的 “arm-2009q3.tar.bz2”。如果使用的是搭建好的环境，确保编译器环境变量，如下图所示。


```
root@ubuntu: ~  
# Alias definitions.  
# You may want to put all your additions into a separate file like  
# ~/.bash_aliases, instead of adding them here directly.  
# See /usr/share/doc/bash-doc/examples in the bash-doc package.  
  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# enable programmable completion features (you don't need to enable  
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile  
# sources /etc/bash.bashrc).  
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then  
#    . /etc/bash_completion  
#fi  
  
export PATH=$PATH:/usr/local/ndk/android-ndk-r8b  
export PATH=$PATH:/usr/local/arm/arm-2009q3/bin  
#export PATH=$PATH:/usr/local/arm/4.4.1/bin  
#export PATH=$PATH:/usr/local/arm/4.3.2/libexec/gcc
```

5.3.1.3 参数配置

编译 uboot 的脚本是源码文件夹中的“build_uboot.sh”，在编译的时候需要向脚本传参数，根据核心板的不同，脚本执行参数如下表所示。

硬件分类	脚本执行参数
核心板 SCP 1G 内存	SCP_1GDDR
核心板 SCP 2G 内存	SCP_2GDDR
核心板 POP 1G 内存	POP_1GDDR
核心板 POP 2G 内存	POP_2GDDR

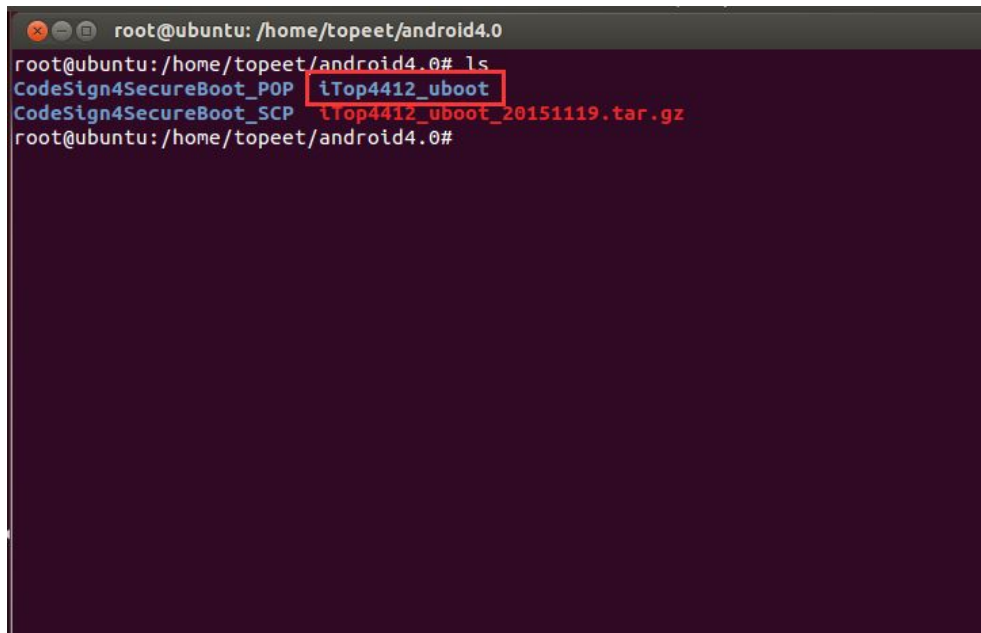
5.3.1.4 编译生成 uboot 镜像举例

这里以 SCP 1G 核心板为例编译 uboot 镜像。

将光盘“06_源码_uboot 和 kernel”目录下“CodeSign4SecureBoot_POP”、

“CodeSign4SecureBoot_SCP”以及“iTop4412_uboot_xxx.tar.gz”拷贝到 Ubuntu 系统

下，然后将 “iTop4412_uboot_xxx.tar.gz” 解压，得到 “iTop4412_uboot” 文件夹，如下图所示。

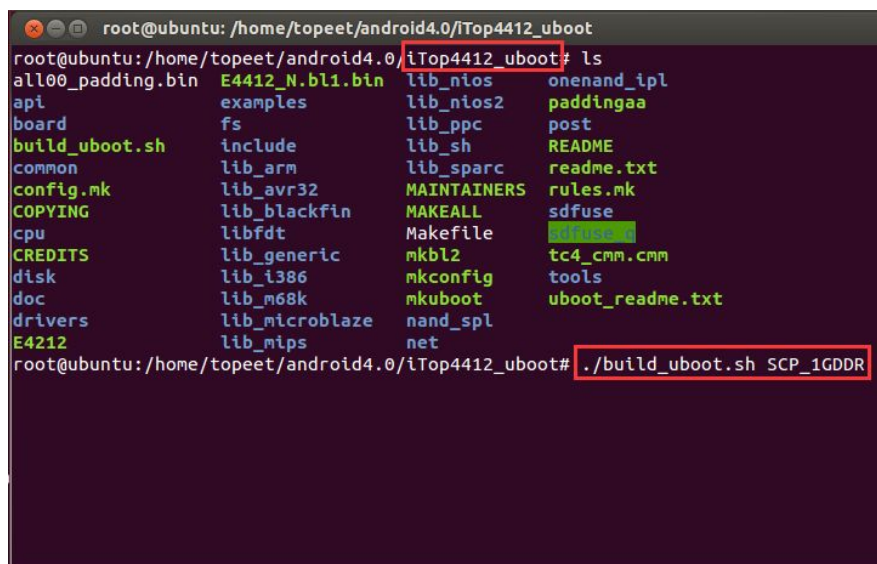


```
root@ubuntu: /home/topeet/android4.0
root@ubuntu: /home/topeet/android4.0# ls
CodeSign4SecureBoot_P0P iTop4412_uboot
CodeSign4SecureBoot_SCP iTop4412_uboot_20151119.tar.gz
root@ubuntu: /home/topeet/android4.0#
```

进入 “iTop4412_uboot” 文件夹，使用编译脚本 “build_uboot.sh” 编译 uboot，这里需要编译的是 “SCP 1G 核心板” 的 uboot 镜像，那么编译命令是

“./build_uboot.sh SCP_1GDDR”

输入编译命令，如下图所示。这里一定先确定核心板是哪种类型，然后将对应的参数传到脚本。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_uboot
root@ubuntu: /home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  E4412_N.bl1.bin  lib_nios  onenand_ipl
api                examples         lib_nios2 paddingaa
board              fs               lib_ppc   post
build_uboot.sh     include          lib_sh    README
common             lib_arm          lib_sparc readme.txt
config.mk           lib_avr32        MAINTAINERS rules.mk
COPYING             lib_blackfin     MAKEALL   sdfuse
cpu                 libfdt           Makefile  sdfuse
CREDITS             lib_generic      mkbl2     tc4_cmm.cmm
disk                lib_i386         mkconfig  tools
doc                 lib_m68k         mkuboot   uboot_readme.txt
drivers             lib_microblaze   nand_spl
E4212              lib_mips         net
root@ubuntu: /home/topeet/android4.0/iTop4412_uboot# ./build_uboot.sh SCP_1GDDR
```

如下图所示，编译中。

```
enand.a drivers/mtd/ubi/libubi.a drivers/mtd/spi/libspi_flash.a drivers/net/libnet.a drivers/net/phy/libphy.a drivers/pct/libpci.a drivers/pcmcia/libpcmcia.a drivers/power/libpower.a drivers/spi/libspi.a drivers/rtc/librtc.a drivers/serial/libserial.a drivers/twserial/libtwserial.a drivers/usb/gadget/libusb_gadget.a drivers/usb/host/libusb_host.a drivers/usb/musb/libusb_musb.a drivers/usb/phy/libusb_phy.a drivers/video/libvideo.a drivers/watchdog/libwatchdog.a common/libcommon.a libbfdt/libbfdt.a api/libapi.a post/libpost.a board/samsung/smdkc210/libsmkc210.a
--end-group /home/topeet/android4.0/iTop4412_uboot/lib_arm/eabi_compat.o -L /usr/local/arm/arm-2009q3/bin/./lib/gcc/arm-none-linux-gnueabi/4.4.1 -lgcc -Map u-boot.map -o u-boot
/usr/local/arm/arm-2009q3/bin/arm-none-linux-gnueabi-objcopy -O srec u-boot u-boot.srec
/usr/local/arm/arm-2009q3/bin/arm-none-linux-gnueabi-objcopy --gap-fill=0xff -O binary u-boot u-boot.bin
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_uboot/sdfuse_q'
gcc -o checksum checksum.c
gcc -o add_sign add_sign.c
gcc -o add_padding add_padding.c
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_uboot/sdfuse_q'
bl2aa file size= 14336B
before padding u-boot.bin file size= 249988B
85884 B written
```

如下图所示，脚本执行完成，在“iTop4412_uboot”文件夹下生成了“u-boot-iTOP-4412.bin”文件。生成的文件“u-boot-iTOP-4412.bin”文件就是 SCP 1G 内存核心板对应的 uboot 镜像文件。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot# ls
all00_padding.bin  examples  lib_sh  rules.mk
api                fs        lib_sparc  sdfuse
board             include   MAINTAINERS  sdfuse
build_uboot.sh    lib_arm  MAKEALL  System.map
checksum_bl2_14k.bin  lib_avr32  Makefile  tc4_cmm.cmm
common           lib_blackfin  mkbl2  tools
config.mk        libbfdt  mkconfig  u-boot
COPYING          lib_generic  mkuboot  u-boot.bin
cpu              lib_i386  nand_spl  u-boot-iTOP-4412.bin
CREDITS          lib_m68k  net      u-boot.lds
disk             lib_microblaze  onenand_ipl  u-boot.map
doc              lib_mips  paddingaa  uboot_readme.txt
drivers          lib_nios  post      u-boot.srec
E4212           lib_nios2  README
E4412_N.bl1.bin  lib_ppc  readme.txt
root@ubuntu:/home/topeet/android4.0/iTop4412_uboot#
```

5.3.2 Linux 内核的编译

5.3.2.1 源码目录

Linux 内核源码在光盘“06_源码_uboot 和 kernel”目录下，如下图所示。

```
CodeSign4SecureBoot_POP
CodeSign4SecureBoot_SCP
iTop4412_Kernel_3.0_20151120.tar.gz
iTop4412_uboot_20151119.tar.gz
```

5.3.2.2 编译器

内核的编译器和 uboot 的编译器一样，参考“5.3.1.2 编译器”。

5.3.2.3 参数配置

内核的编译是组合式配置文件，基本的配置文件名是“config_for_android_YY_elite”，YY 表示用下表所示的参数替代。

硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存	config_for_android_scp_elite
核心板 POP 1G 内存	config_for_android_pop_elite
核心板 POP 2G 内存	config_for_android_pop2G_elite

5.3.2.4 编译生成内核镜像举例

这里以 SCP 1G 核心板为例编译 zImage 内核镜像,那么配置文件为

“config_for_android_scp_elite”。

将光盘“06_源码_uboot 和 kernel”目录下的压缩包

“iTop4412_Kernel_3.0_xxx.tar.gz”拷贝到 Ubuntu，然后解压，得到文件夹

“iTop4412_Kernel_3.0 ”，如下图所示。

```
root@ubuntu: /home/topeet/android4.0
root@ubuntu: /home/topeet/android4.0# ls
CodeSign4SecureBoot_P0P  iTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iTop4412_uboot
iTop4412_Kernel_3.0      iTop4412_uboot_20151119.tar.gz
root@ubuntu: /home/topeet/android4.0#
```

进入文件夹 “iTop4412_Kernel_3.0 ” ，使用命令

“cp config_for_android_scp_elite .config” 覆盖自带的配置文件，如下图所示。

```
root@ubuntu: /home/topeet/android4.0# ls
CodeSign4SecureBoot_P0P  iTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iTop4412_uboot
iTop4412_Kernel_3.0      iTop4412_uboot_20151119.tar.gz
root@ubuntu: /home/topeet/android4.0# cd iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_android_s
cp_elite .config
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0#
```

然后使用编译命令 “make zImage” ，如下图所示。

```
root@ubuntu: /home/topeet/android4.0# ls
CodeSign4SecureBoot_P0P  iTop4412_Kernel_3.0_20151120.tar.gz
CodeSign4SecureBoot_SCP  iTop4412_uboot
iTop4412_Kernel_3.0      iTop4412_uboot_20151119.tar.gz
root@ubuntu: /home/topeet/android4.0# cd iTop4412_Kernel_3.0
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_android_s
cp_elite .config
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
```


编译中，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# cp config_for_android_3.0
cp_elite .config
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# make zImage
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/lex.zconf.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
scripts/kconfig/conf --silentoldconfig Kconfig
CHK include/linux/version.h
UPD include/linux/version.h
CHK include/generated/utsrelease.h
UPD include/generated/utsrelease.h
Generating include/generated/mach-types.h
CC kernel/bounds.s
GEN include/generated/bounds.h
CC arch/arm/kernel/asm-offsets.s
GEN include/generated/asm-offsets.h
CALL scripts/checksyscalls.sh
CC scripts/mod/empty.o
HOSTCC scripts/mod/mk_elfconfig
```

编译完成，如下图所示。

```
CC init/version.o
LD init/built-in.o
LD .tmp_vmlinux1
KSYM .tmp_kallsyms1.S
AS .tmp_kallsyms1.o
LD .tmp_vmlinux2
KSYM .tmp_kallsyms2.S
AS .tmp_kallsyms2.o
LD vmlinux
SYSMAP System.map
SYSMAP .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS arch/arm/boot/compressed/head.o
GZIP arch/arm/boot/compressed/piggy.gzip
AS arch/arm/boot/compressed/piggy.gzip.o
CC arch/arm/boot/compressed/misc.o
CC arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS arch/arm/boot/compressed/lib1funcs.o
LD arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

文件夹 “iTop4412_Kernel_3.0 ” 下的 “arch” --> “arm” --> “boot” 会生成镜像文件 “zImage” ，这个 zImage 镜像可以给 **SCP 1G** 和 **SCP 2G** 的核心板使用，如下图所示。


```
LD      .tmp_vmlinux1
KSYM    .tmp_kallsyms1.S
AS      .tmp_kallsyms1.o
LD      .tmp_vmlinux2
KSYM    .tmp_kallsyms2.S
AS      .tmp_kallsyms2.o
LD      vmlinux
SYSMAP  System.map
SYSMAP  .tmp_System.map
OBJCOPY arch/arm/boot/Image
Kernel: arch/arm/boot/Image is ready
AS      arch/arm/boot/compressed/head.o
GZIP    arch/arm/boot/compressed/piggy.gzip
AS      arch/arm/boot/compressed/piggy.gzip.o
CC      arch/arm/boot/compressed/misc.o
CC      arch/arm/boot/compressed/decompress.o
SHIPPED arch/arm/boot/compressed/lib1funcs.S
AS      arch/arm/boot/compressed/lib1funcs.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls arch/arm/boot/
bootp compressed Image install.sh Makefile zImage
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

由于 Android 代码中需要内核中 wifi 驱动，在编译好内核之后，还需要在内核目录中，执行编译模块的命令“make modules”，再执行编译后面 Android4.0.3 文件系统的脚本，Android4.0.3 源码才能通过。

5.3.3 Android4.0.3 的编译

5.3.3.1 源码目录

Android4.0.3 文件系统的源码在光盘“07_源码_Android4.0.3 文件系统”目录下，如下图所示。

文件名	大小
iTop4412_ICS_git_20151120.tar.gz	1.88GB

5.3.3.2 编译器

Android4.0.3 的编译器是 4.4.7 版本（包括其他所有版本的 Android 编译器都是相同的），如下图所示，在控制台使用命令“gcc -v”，可以查看到 gcc 的版本。

```
root@ubuntu:/home/topeet# gcc -v
Using built-in specs.
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu/Linaro 4.4.7-1ubuntu2' --with-bugurl=file:///usr/share/doc/gcc-4.4/README.Bugs --enable-languages=c,c++,fortran,objc,obj-c++ --prefix=/usr --program-suffix=-4.4 --enable-shared --enable-linker-build-id --with-system-zlib --libexecdir=/usr/lib --without-included-gettext --enable-threads=posix --with-gxx-include-dir=/usr/include/c++/4.4 --libdir=/usr/lib --enable-nls --with-sysroot=/ --enable-clocale=gnu --enable-libstdcxx-debug --enable-objc-gc --disable-werror --with-arch-32=i686 --with-tune=generic --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu
Thread model: posix
gcc version 4.4.7 (Ubuntu/Linaro 4.4.7-1ubuntu2)
root@ubuntu:/home/topeet#
```

5.3.3.3 参数配置

无参数配置。所有种类核心板对应的 Android4.0.3 都使用同一套源码，同一种编译方法。

编译脚本是 “build_android.sh” 。

5.3.3.4 编译生成 Android4.0.3 镜像

将光盘 “07_源码_Android4.0.3 文件系统” 目录下压缩包

“iTop4412_ICS_git_xxx.tar.gz” 拷贝到 Ubuntu 系统中，解压压缩包，得到文件夹

“iTop4412_ICS_git” 。这里需要注意的是，Android 源码文件夹 “iTop4412_ICS_git ” 和内核源码文件夹 “iTop4412_Kernel_3.0” 需要放到同一目录下，如下图所示。

```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP      iTop4412_Kernel_3.0
CodeSign4SecureBoot_SCP      iTop4412_Kernel_3.0_20151120.tar.gz
iTop4412_ICs_git              iTop4412_uboot
iTop4412_ICs_glt_20151120.tar.gz iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0#
```

进入 “iTop4412_ICs_git” 目录，使用命令 “./build_android.sh” 运行编译脚本,编译 Android4.0.3，如下图所示。注意：编译 Android4.0.3 必须保证给 Ubuntu 系统提供 2G 以上内存。这里“提供 2G 以上内存”的意思，不仅仅是指在 VMware Workstations 虚拟机中设置分配 2G 内存。例如，用户在编译的时候，PC 机的内存一共是 4G，在虚拟机中设置分配 3G 内存，但是在 Windows 系统下，开着 QQ，杀毒软件，音乐播放器，浏览器等等，这样在 Windows 下占用的内存就快 2G 了，那么虚拟机会自动调整到只占用 2G，甚至不到 2G 的内存，这样就有可能没法编译通过。

正确的分配内存方法是，首先给虚拟机分配 2G 以上内存，然后在 Windows 下关掉尽量多的不必要的应用，关掉尽量多应用的后台程序。

当然，如果用户的 PC 是 8G 或者 16G 内存，随意就成。

```
root@ubuntu:/home/topeet/android4.0# ls
CodeSign4SecureBoot_POP      iTop4412_Kernel_3.0
CodeSign4SecureBoot_SCP      iTop4412_Kernel_3.0_20151120.tar.gz
iTop4412_ICS_git              iTop4412_uboot
iTop4412_ICS_git_20151120.tar.gz iTop4412_uboot_20151119.tar.gz
root@ubuntu:/home/topeet/android4.0# cd iTop4412_ICS_git
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git# ./build_android.sh
```

开始编译，如下图所示。

```
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git# ./build_android.sh

Build android for smdk4x12

[[[[[[ Build android platform ]]]]]]

make -j4 PRODUCT=full_smdk4x12-eng

=====
PLATFORM_VERSION_CODENAME=REL
PLATFORM_VERSION=4.0.3
TARGET_PRODUCT=full_smdk4x12
TARGET_BUILD_VARIANT=eng
TARGET_BUILD_TYPE=release
TARGET_BUILD_APPS=
TARGET_ARCH=arm
TARGET_ARCH_VARIANT=armv7-a-neon
HOST_ARCH=x86
HOST_OS=linux
HOST_BUILD_TYPE=release
BUILD_ID=IML74K
=====
```

编译比较耗费时间，第一次编译会花费 60 分钟以上。

```
out/target/product/smdk4x12/system.img+ total size is 253251420
Total compile time is 4327 seconds

[[[[[[ Make ramdisk image for u-boot ]]]]]]

Image Name: ramdisk
Created: Fri Nov 27 07:34:58 2015
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 922118 Bytes = 900.51 kB = 0.88 MB
Load Address: 40800000
Entry Point: 40800000

[[[[[[ Make additional images for fastboot ]]]]]]

boot.img -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
update.zip -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
    adding: android-info.txt (stored 0%)
    adding: boot.img (deflated 1%)
    adding: system.img (deflated 35%)

ok success !!!
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git#
```

编译完成后在文件夹 “iTop4412_ICS” --> “out” --> “target” --> “product” --> “smdk4x12” 中生成 Android4.0.3 镜像文件 “ramdisk-uboot.img” 和 “system.img” , 如下图所示。

```
Created:      Fri Nov 27 07:34:58 2015
Image Type:   ARM Linux RAMDisk Image (uncompressed)
Data Size:    922118 Bytes = 900.51 kB = 0.88 MB
Load Address: 40800000
Entry Point:  40800000

[[[[[[ Make additional images for fastboot ]]]]]]

boot.img -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
update.zip -> /home/topeet/android4.0/iTop4412_ICS_git/out/target/product/smdk4x12
  adding: android-info.txt (stored 0%)
  adding: boot.img (deflated 1%)
  adding: system.img (deflated 35%)

ok success !!!
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git# ls out/target/product/smdk4x12/
android-info.txt  installed-files.txt  root          update.zip
boot.img          obj                  symbols       userdata.img
clean_steps.mk   previous_build_config.mk  system       zImage
data             ramdisk-uboot.img      system.img
root@ubuntu:/home/topeet/android4.0/iTop4412_ICS_git#
```

如果编译报错，请注意：在编译好内核之后，还需要在内核目录中，执行编译模块的命令 “make modules” ，再执行编译 Android4.0.3 文件系统的脚本才不会报错，因为 Android4.0.3 源码中会用到内核中的 wifi 驱动，wifi 模块的驱动必须要使用命令编译一下才行。

5.4 Android4.4 的编译

Android4.4 的内核以及文件系统的源码在网盘 “iTOP4412 开发板资料汇总（不含光盘内容）\iTOP-4412 开发板系统源码及镜像（其他）\android_4.4.4 源码以及对应 Kernel 源码” 目录下。

需要注意的是，其中 “20170803” 以及后续新增的目录下的源码编译方法参考 5.4.1 小节，在 “20170803” 之前的源码编译参考 5.4.2 小节。

另外，由于 eMMC 升级和屏幕升级，如果用户是 2018 年或者之后购买的开发板，或者新购买了金属 7 寸屏/10.1 寸屏，请直接下载“2018xxx”目录下上传的内核代码和镜像测试，Android 源码和 20170803 通用，编译方法参考 5.4.1 小节。

5.4.1 Android4.4 源码编译

注意：本小节，介绍的源码指的是“20170803”以及后续新增的源码。

5.4.1.1 uboot 的编译

Android4.4.4 对应 uboot 的源码，编译器，参数配置，编译脚本以及编译参数和 Android4.0.3 的 uboot 全部一模一样。

5.4.1.2 Linux 内核的编译

源码目录

网盘下载 Android4.4.4 对应的源码。在网盘“iTOP-4412 开发板系统源码及镜像（其他）”→“android_4.4.4 源码及镜像”目录下，在最新日期的文件夹下的“iTop4412-android4.4-kernel_xxxx.tar.gz”压缩包，xxxx 表示日期。

编译器

Android4.4.4 对应内核的编译器和 Android4.0.3 的内核编译器一模一样。

参数配置

内核的编译是组合式配置文件，基本的配置文件名是 “config_for_android_XX_YY” ，XX,YY 表示用下表所示的参数替代。POP 和 SCP 分别对应核心板的 POP 和 SCP；elite 和 super 分别对应精英版和全能版；no_wifi 参数仅用于精英版，表示不支持 WiFi 模块（如果没有 WiFi 模块，那么就需要配置为 no_wifi）。

硬件分类	配置文件
精英版 Android4.4POP 核心板	config_for_android_pop_elite
精英版 Android4.4POP 核心板不支持 WIFI	config_for_android_pop_no_wifi
全能版 Android4.4POP 核心板	config_for_android_pop_super
精英版 Android4.4SCP 核心板	config_for_android_scp_elite
精英版 Android4.4SCP 核心板不支持 WIFI	config_for_android_scp_no_wifi
全能版 Android4.4SCP 核心板	config_for_android_scp_super

如上表所示，如果需要 Android4.4 支持 WiFi，则需要配置对应的参数，不支持 WiFi 也需要进行对应配置。

编译生成内核镜像举例

和 Android4.0.3 内核一样，如果需要编译对应核心板的内核，首先使用 cp 命令将对应的配置文件覆盖掉 “.config”，然后在执行编译命令 “make zImage”。

生成内核镜像的目录也是 “arch” --> “arm” --> “boot”。

例如：开发板是精英版，POP 核心板，需要支持 WiFi 模块，那么缺省文件就需要配置为 “config_for_android_pop_elite”。

开发板是精英版，POP 核心板，不支持 WiFi 模块，那么缺省文件就需要配置为 “config_for_android_pop_no_wifi”。

5.4.1.3 Android4.4.4 的编译

更新环境 javap

编译 Android4.4.4，还缺少一个“javap”命令，使用命令：

```
“update-alternatives --install "/usr/bin/javap" "javap"
```

```
"/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1”
```

更新一下，如下图所示。

```
root@ubuntu:/home/iTOP-4412-Android4.4# update-alternatives --install "/usr/bin/javap"
" "javap" "/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1
```

内核以及 Android 源码路径

编译 Android4.4 源码的时候，一定要注意内核源码的目录。在和 Android 源码目录

“iTop4412_KK4.4_git”的同一级目录下，将内核源码解压。

例如，如下图所示，作者将内核源码和 Android4.4 源码压缩包放到“/home/iTOP-4412-Android4.4”目录下，解压“iTop4412_KK4.4_git_xxx.tar.gz”和“iTop4412-android4.4-kernel_xxx.tar.gz”，Android 源码解压之后得到“iTop4412_KK4.4_git”和“kernel”目录。“iTop4412_KK4.4_git”目录下就是 Android 源码，“kernel/iTop4412_Kernel_3.0”目录下就是内核源码。

```
root@ubuntu:/home/iTOP-4412-Android4.4# ls
iTop4412-android4.4-kernel_20170802.tar.gz  iTop4412_KK4.4_git_20170802.tar.gz
iTop4412_KK4.4_git                          kernel
```

另外在编译 Android4.4 之前，**必须成功编译 Android4.4 对应的内核。**

然后进入 Android4.4.4 源码解压后得到文件夹“iTop4412_KK4.4_git”中，使用命令

“./build_android.sh”，运行一键编译脚本，开始编译 Android4.4.4。**注意：编译**

Android4.4.4 必须保证给 Ubuntu 系统提供 3G 以上内存。这里“提供 3G 以上内存”的意

思，不仅仅是指在 VMware Workstations 虚拟机中设置分配 3G 内存。例如，用户在编译的时候，PC 机的内存一共是 4G，在虚拟机中设置分配 3G 内存，但是在 Windows 系统下，开着 QQ，杀毒软件，音乐播放器，浏览器等等，这样在 Windows 下占用的内存就快 2G 了，那么虚拟机会自动调整到只占用 2G，甚至不到 2G 的内存，这样是没法编译通过的。

正确的分配内存方法是，首先给虚拟机分配 3G 以上内存，然后在 Windows 下关掉尽量多的不必要的应用，关掉尽量多应用的后台程序。

当然，如果用户的 PC 是 8G 或者 16G 内存，随意就成。

编译完成之后，在文件夹 “iTop4412_KK4.4_git/out/target/product/smdk4x12” 中，生成镜像 “system.img” 和 “ramdisk-uboot.img”。

这里还需要注意的是，Android4.4.4 源码需要占用较大的空间，用户需要确认有足够的空间才能够成功编译。Android4.0.3 编译完成后总共大约占用 18G 的空间，Android4.4.4 编译完成后总共占用大约 36G 的空间。

用户完全按照步骤编译错误，可以使用 “df -l” 查看一下盘符剩余空间还剩下多少，如果是已使用 100%，则是空间不足。

5.4.2 Android4.4 旧源码编译

注意：本小节，介绍的旧源码指的是 “20170803” 之前的源码。

在网盘目录中 “iTOP4412 开发板资料汇总（不含光盘内容）\iTOP-4412 开发板系统源码及镜像（其他）\android_4.4.4 源码以及对应 Kernel 源码” 中下载 Android4.4.4 的文件系统，并通过 github 下载 uboot（Android4.4 的 uboot 源码和 Android4.0.3 完全一样）和 kernel 的源码（参考附录六）。

5.4.2.1 uboot 的编译

Android4.4.4 对应 uboot 的源码，编译器，参数配置，编译脚本以及编译参数和 Android4.0.3 的 uboot 全部一模一样。

5.4.2.2 Linux 内核的编译

源码目录

网盘下载 Android4.4.4 对应的源码。在网盘 “iTOP-4412 开发板系统源码及镜像（其他）” → “android_4.4.4 源码及镜像” 目录下。

编译器

Android4.4.4 对应内核的编译器和 Android4.0.3 的内核编译器一模一样。

参数配置

内核的编译是组合式配置文件，基本的配置文件名是 “config_for_android_YY”，YY 表示用下表所示的参数替代。

硬件分类	配置文件
核心板 SCP 1G 或者 2G 内存	config_for_android_scp
WiFi 支持配置：	config_for_android_scp_wifi&Bluetooth
核心板 POP 1G 或者 2G 内存	config_for_android_pop
WiFi 支持配置：	config_for_android_pop_wifi&Bluetooth

如上表所示，如果需要 Android4.4.4 支持 WiFi，则需要配置对应的参数。

编译生成内核镜像举例

和 Android4.0.3 内核一样，如果需要编译对应核心板的内核，首先使用 cp 命令将对应的配置文件覆盖掉“.config”，然后在执行编译命令“make zImage”。

生成内核镜像的目录也是“arch”-->“arm”-->“boot”。

5.4.2.3 Android4.4.4 的编译

Android4.4.4 源码在网盘“iTOP-4412 开发板系统源码及镜像（其他）\android_4.4.4 源码以及对应 Kernel 源码”目录中下载，编译器和参数配置和 Android4.0.3 一模一样。

更新环境 javap

编译 Android4.4.4，还缺少一个“javap”命令，使用命令：

“update-alternatives --install "/usr/bin/javap" "javap"

"/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1”

更新一下，如下图所示。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# update-alternatives --install  
l "/usr/bin/javap" "javap" "/usr/lib/jvm/jdk1.6.0_43/bin/javap" 1
```

源码下载

内核的源码也可以在 github 上下载，具体参考使用手册附录 6.2，附录中有迅为 Android4.4 对应内核源码 github 下载地址。

使用手册附录中有介绍 repo 下载 Android4.4 源码的方法，但是从今年（2016）开始，repo 经常性的无法下载和使用。所以最好到迅为百度网盘中下载源码（目录参考使用手册编译章节），选取日期最近的目录下载即可。

网盘中源码包一般是 gz 后缀、7z 或者 rar 后缀。gz 后缀可以直接在 Ubuntu 下解压；7z 和 rar 后缀都是需要先在 windows 下解压出 gz 后缀压缩包，然后再在 Ubuntu 下解压。

内核以及 Android 源码路径

编译 Android4.4 源码的时候，一定要注意内核源码的目录。在和 Android 源码目录 “iTop4412_KK4.4_git” 的同一级目录下，新建一个 kernel 目录，然后将内核源码放到 kernel 目录下，要使用默认的文件名 “iTop4412_Kernel_3.0”。

例如：在 Android4.4 的源码目录下，使用命令

“ls ../kernel/iTop4412_Kernel_3.0/”，如下图所示，内核的目录才是正确的。

```
root@ubuntu:/home/exynos4412/iTop4412_KK4.4_git# ls ../kernel/iTop4412_Kernel_3.0/
arch          Documentation  lib           scripts
binary        drivers       MAINTAINERS  security
block         firmware     Makefile     sound
config_for_android_pop  fs           mm           System.map
config_for_android_pop_wifi&Bluetooth  include     modem.patch  tools
config_for_android_scp  init        Module.symvers  usr
config_for_android_scp_wifi&Bluetooth  ipc        net          virt
config_for_linux        Kbuild     pull_log.bat  vmlinux
COPYING           Kconfig    README        vmlinux.o
CREDITS           kernel     REPORTING-BUGS
crypto            kernel_readme.txt  samples
```

另外在编译 Android4.4 之前，必须成功编译 Android4.4 对应的内核。

然后进入 Android4.4.4 源码解压后得到文件夹 “iTop4412_KK4.4” 中，使用命令

“./build_android.sh”，运行一键编译脚本，开始编译 Android4.4.4。注意：编译

Android4.4.4 必须保证给 Ubuntu 系统提供 3G 以上内存。这里“提供 3G 以上内存”的意思，不仅仅是指在 VMware Workstations 虚拟机中设置分配 3G 内存。例如，用户在编译的时候，PC 机的内存一共是 4G，在虚拟机中设置分配 3G 内存，但是在 Windows 系统下，开着 QQ，杀毒软件，音乐播放器，浏览器等等，这样在 Windows 下占用的内存就快 2G 了，那么虚拟机会自动调整到只占用 2G，甚至不到 2G 的内存，这样是没法编译通过的。

正确的分配内存方法是，首先给虚拟机分配 3G 以上内存，然后在 Windows 下关掉尽量多的不必要的应用，关掉尽量多应用的后台程序。

当然，如果用户的 PC 是 8G 或者 16G 内存，随意就成。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# ./build_android.sh

Build android for smdk4x12

including device/samsung/manta/vendorsetup.sh
including device/samsung/smdk4x12/vendorsetup.sh
including device/asus/grouper/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
including device/asus/tilapia/vendorsetup.sh
```

如下图所示，编译完成。在文件夹

“iTop4412_KK4.4/out/target/product/smdk4x12” 中，生成镜像 “system.img” 和 “ramdisk-uboot.img”。

```
out/target/product/smdk4x12/system.img+ maxsize=4110/9680 blocksize=4224 total=2
91223112 reserve=4156416
Total compile time is 5000 seconds

[[[[[[[ Make ramdisk image for u-boot ]]]]]]]

Image Name: ramdisk
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 325500 Bytes = 317.87 kB = 0.31 MB
Load Address: 40800000
Entry Point: 40800000
/home/topeet/Android4.4/iTop4412_KK4.4/out/target/product/smdk4x12/ramdisk-uboot
.img
OK!
ok success !!!
```

这里还需要注意的是，Android4.4.4 源码需要占用较大的空间，用户需要确认有足够的空间才能够成功编译。Android4.0.3 编译完成后总共大约占用 18G 的空间，Android4.4.4 编译完成后总共占用大约 36G 的空间，如下图。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_KK4.4# du -sh
36G
```

用户完全按照步骤编译错误，可以使用 “df -l” 查看一下盘符剩余空间还剩下多少，如下图所示，如果是已使用 100%，则是空间不足。

```
root@ubuntu:/home/topeet/Android4.4/iTop4412_Kernel_3.0# df -l
Filesystem      1K-blocks    Used Available Use% Mounted on
/dev/sda1        60894268 45378900 12422120  79% /
udev            2013748      4    2013744    1% /dev
tmpfs            809208     812    808396    1% /run
none              5120        0      5120     0% /run/lock
none            2023016     200    2022816    1% /run/shm
root@ubuntu:/home/topeet/Android4.4/iTop4412_Kernel_3.0#
```