

[如何下载并编译Android4.0 内核源码goldfish\(图文\)](#)

第一步:下载 goldfish 源码

在 Android 源码根目录下新建 kernel 文件夹

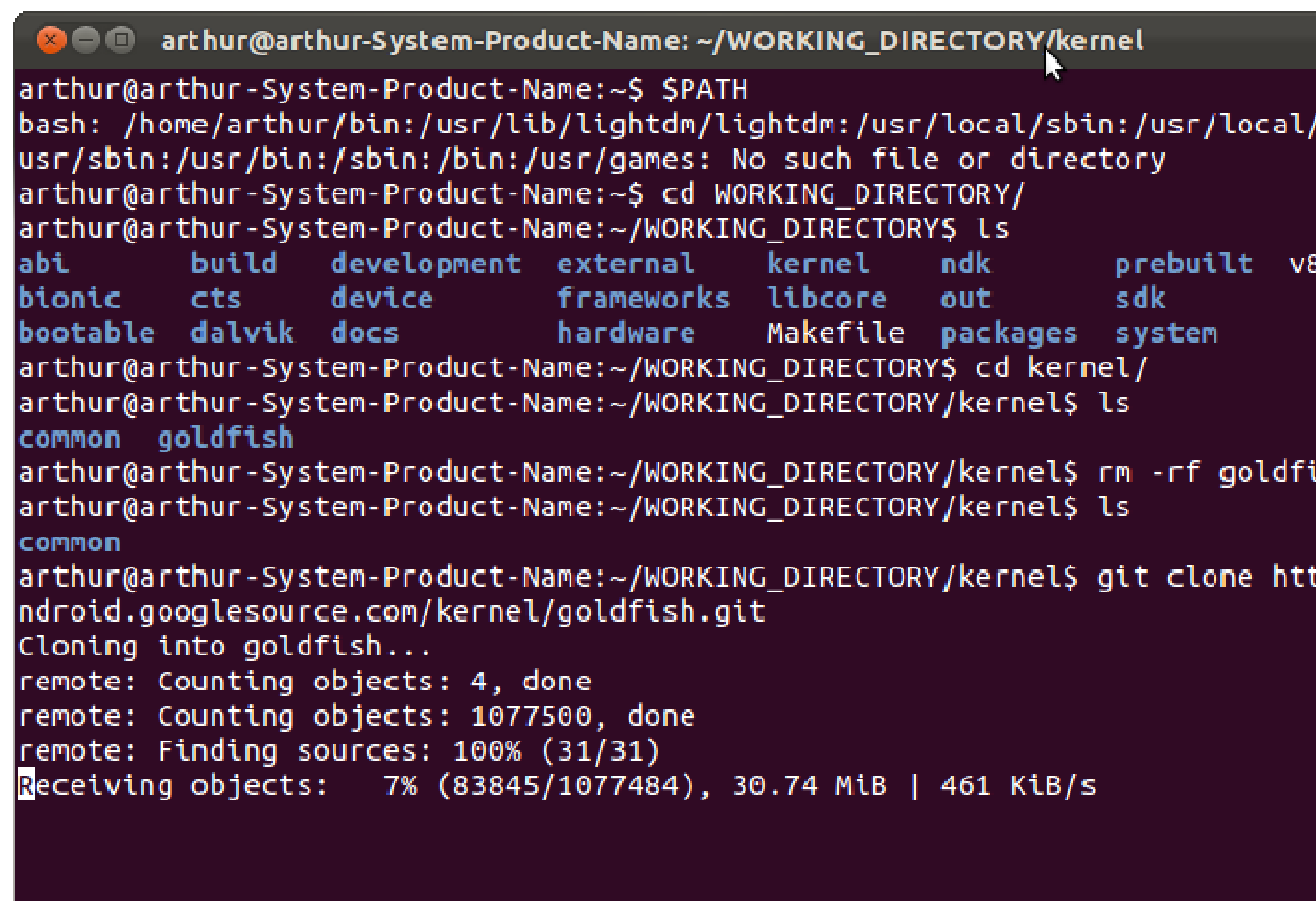
[html] [view plaincopy](#)

1. \$mkdir kernel
2. \$cd kernel

下载源码:

[html] [view plaincopy](#)

1. \$git clone http://android.googlesource.com/kernel/goldfish.git

A terminal window screenshot showing the steps to clone the goldfish kernel source code. The user is in the directory ~/WORKING_DIRECTORY/kernel. They run 'ls' and see a list of directories including 'goldfish'. They then run 'rm -rf goldfish' to remove it. Finally, they run 'git clone http://android.googlesource.com/kernel/goldfish.git', which successfully clones the repository into the 'goldfish' directory. The terminal output shows the progress of cloning, including counting objects and finding sources.

```
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel
arthur@arthur-System-Product-Name:~$ $PATH
bash: /home/arthur/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/
usr/sbin:/usr/bin:/sbin:/bin:/usr/games: No such file or directory
arthur@arthur-System-Product-Name:~$ cd WORKING_DIRECTORY/
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY$ ls
abi      build   development  external  kernel  ndk      prebuilt  v8
bionic   cts     device       frameworks libcore  out       sdk
bootable dalvik  docs         hardware  Makefile packages system
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY$ cd kernel/
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ ls
common  goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ rm -rf goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ ls
common
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ git clone http://
android.googlesource.com/kernel/goldfish.git
Cloning into goldfish...
remote: Counting objects: 4, done
remote: Counting objects: 1077500, done
remote: Finding sources: 100% (31/31)
Receiving objects: 7% (83845/1077484), 30.74 MiB | 461 KiB/s
```

下载完毕如下图:

```

x - □ arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel
arthur@arthur-System-Product-Name:~$ $PATH
bash: /home/arthur/bin:/usr/lib/lightdm/lightdm:/usr/local/sbin:/usr/local/
usr/sbin:/usr/bin:/sbin:/bin:/usr/games: No such file or directory
arthur@arthur-System-Product-Name:~$ cd WORKING_DIRECTORY/
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY$ ls
abi          build  development  external  kernel  ndk      prebuilt  v8
bionic       cts    device       frameworks  libcore  out      sdk
bootable     dalvik docs        hardware   Makefile packages system
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY$ cd kernel/
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ ls
common  goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ rm -rf goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ ls
common
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$ git clone https://
ndroid.googlesource.com/kernel/goldfish.git
Cloning into goldfish...
remote: Counting objects: 4, done
remote: Counting objects: 1077500, done
remote: Finding sources: 100% (31/31)
remote: Total 1077484 (delta 899689), reused 1077484 (delta 899689)
Receiving objects: 100% (1077484/1077484), 227.29 MiB | 458 KiB/s, done.
Resolving deltas: 100% (900010/900010), done.
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel$
```

此时在 kernel 目录下会生成一个 goldfish 文件夹。进入此目录：

[html] [view plaincopy](#)

1. \$cd goldfish

此目录下有一个隐藏的目录.git, 通过

[html] [view plaincopy](#)

1. \$ls -al

可看到此目录：

```
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish$ ls -la
total 12
drwxr-xr-x 3 arthur arthur 4096 2011-12-14 14:38 .
drwxrwxr-x 4 arthur arthur 4096 2011-12-14 14:38 ..
drwxrwxr-x 8 arthur arthur 4096 2011-12-14 14:47 .git
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish$
```

查看所有分支：

[html] [view plaincopy](#)

1. git branch -a

如下图：

```
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ ls -l
total 12
drwxr-xr-x 3 arthur arthur 4096 2011-12-14 14:38 .
drwxrwxr-x 4 arthur arthur 4096 2011-12-14 14:38 ..
drwxrwxr-x 8 arthur arthur 4096 2011-12-14 14:47 .git
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ git
h -a
* master
  remotes/origin/HEAD -> origin/master
  remotes/origin/android-goldfish-2.6.29
  remotes/origin/master
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$
```

check out:

[html] [view plaincopy](#)

1. \$git checkout remotes/origin/android-goldfish-2.6.29

```
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ git
out remotes/origin/android-goldfish-2.6.29
Checking out files: 100% (26801/26801), done.
Note: checking out 'remotes/origin/android-goldfish-2.6.29'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 46b05b2... goldfish: Enable CONFIG_TUN
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$
```

此时你会看到 goldfish 目录下会出现很多文件:

[html] [view plaincopy](#)

1. \$ls

```

    arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ git
out remotes/origin/android-goldfish-2.6.29
Checking out files: 100% (26801/26801), done.
Note: checking out 'remotes/origin/android-goldfish-2.6.29'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

  git checkout -b new_branch_name

HEAD is now at 46b05b2... goldfish: Enable CONFIG_TUN
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ ls
arch      crypto      fs          Kbuild      Makefile     REPORTING-BUGS  sou
block     Documentation include     kernel       mm           samples        usn
COPYING   drivers     init        lib          net          scripts        vin
CREDITS   firmware    ipc         MAINTAINERS  README      security
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$
```

这个时候 goldfish 源码就已经下下来了，接下来的事情就是编译了。

第二步:编译 goldfish

导出交叉编译工具目录到\$PATH 环境变量中去。

[html] [view plaincopy](#)

1. export PATH=\$PATH:~/WORKING_DIRECTORY/prebuilt/linux-x86/toolchain/arm-eabi-4.4.3/bin

我们将使用上述这个目录下的交叉编译器 **arm-eabi-gcc**

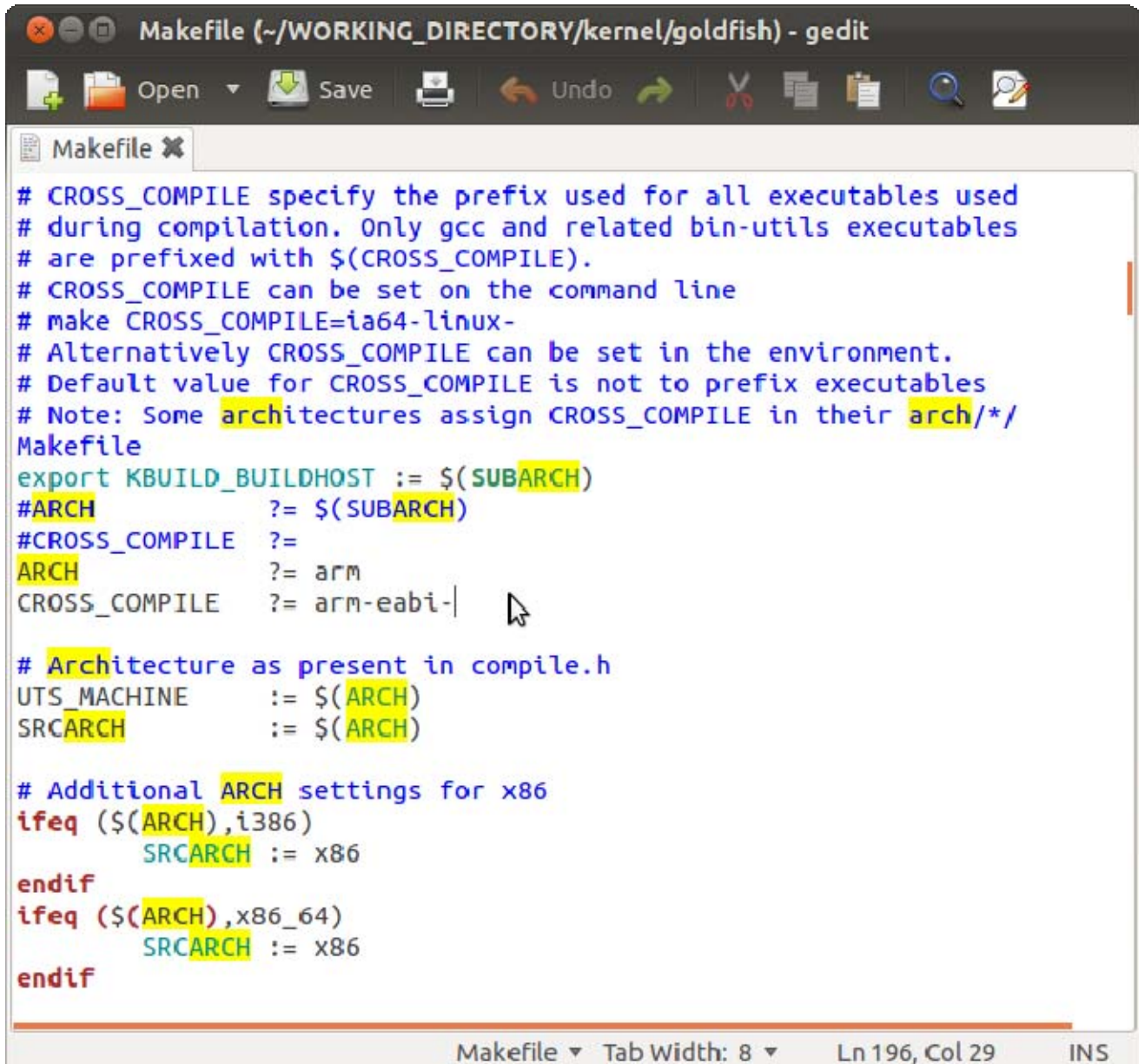
然后在 glodfish 目录下用 gedit 打开 Makefile 文件，找到这两行文字：

```
# ARCH
```

```
?= (SUBARCH)
```

```
# CROSS_COMPILE?=
修改为:
ARCH ?= arm
CROSS_COMPILE      ?= arm-eabi-
[html] view plaincopy
```

1. \$gedit Makefile



```
Makefile (~/WORKING_DIRECTORY/kernel/goldfish) - gedit

# CROSS_COMPILE specify the prefix used for all executables used
# during compilation. Only gcc and related bin-utils executables
# are prefixed with $(CROSS_COMPILE).
# CROSS_COMPILE can be set on the command line
# make CROSS_COMPILE=ia64-linux-
# Alternatively CROSS_COMPILE can be set in the environment.
# Default value for CROSS_COMPILE is not to prefix executables
# Note: Some architectures assign CROSS_COMPILE in their arch/*/
Makefile
export KBUILD_BUILDHOST := $(SUBARCH)
#ARCH ?= $(SUBARCH)
#CROSS_COMPILE ?=
ARCH ?= arm
CROSS_COMPILE ?= arm-eabi-

# Architecture as present in compile.h
UTS_MACHINE := $(ARCH)
SRCARCH := $(ARCH)

# Additional ARCH settings for x86
ifeq ($(ARCH),i386)
    SRCARCH := x86
endif
ifeq ($(ARCH),x86_64)
    SRCARCH := x86
endif
```

注意:ARCH ?=arm, 的 arm 后边不要有空格, 不然就会出现如下类似错误:
Make:...../kernel/goldfish/arch/arm: Is a directory. Stop.
害我白白浪费了几个小时.

上述操作的目的是为了指定目标设备的体系架构和交叉编译器, 其实也可以通过以下指令来完成:

[plain] [view plaincopy](#)

1. \$ export ARCH=arm
2. \$ export SUBARCH=arm
3. \$ export CROSS_COMPILE=arm-eabi-

为了确保环境参数正确，接下来执行下面两条指令，否则有可能不能正常启动模拟器。（注：以下两条指令是在 Android 源码根目录下执行）

[cpp] [view plaincopy](#)

1. \$ source build/envsetup.sh //缺少这条指令，可能会导致无法编译通过
2. \$ lunch full-eng //缺少这条指令，可能会导致无法启动模拟器，系统报无法找到 AVD，并要求你创建 AVD。

关闭 gedit, 接下来就开始 make 了, 执行如下指令:

[html] [view plaincopy](#)

1. \$ make goldfish_armv7_defconfig
2. \$ make

注：用\$make goldfish_defconfig 这样配置也可以编译通过，模拟器也可以启动，但是 Android 的开机画机就显示不了，\$adb shell 也死活连不上，原因就是在这个 goldfish_defconfig 这个配置文件问题。

提示：

\$make goldfish_armv7_defconfig 指令的意思是将目录 WORKING_DIRECTORY/kernel/goldfish/arch/arm/configs/下的 goldfish_armv7_defconfig 文件内的 Kconfig 配置内容复制到 WORKING_DIRECTORY/kernel/goldfish/目录下的.config 文件中，.config 文件是一个隐藏目录，保存着各个目录下 Kconfig 文件的配置。最终结果如下图所示：


```

x _ □ arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
UPD      include/linux/compile.h
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.gz
AS        arch/arm/boot/compressed/piggy.o
CC        arch/arm/boot/compressed/misc.o
LD        arch/arm/boot/compressed/vmlinux
OBJCOPY   arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
arthur@arthur-System-Product-Name:~/WORKING_DIRECTORY/kernel/goldfish$ ls
```

这样就表示编译成功了.

[html] [view plaincopy](#)

1. \$ ls arch/arm/boot/

可以看到 zImage 文件.

```

arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
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.gz
AS      arch/arm/boot/compressed/piggy.o
CC      arch/arm/boot/compressed/misc.o
LD      arch/arm/boot/compressed/vmlinux
OBJCOPY arch/arm/boot/zImage
Kernel: arch/arm/boot/zImage is ready
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish$ ls a
rm/boot/
bootp  compressed  Image  install.sh  Makefile  zImage
arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish$
```

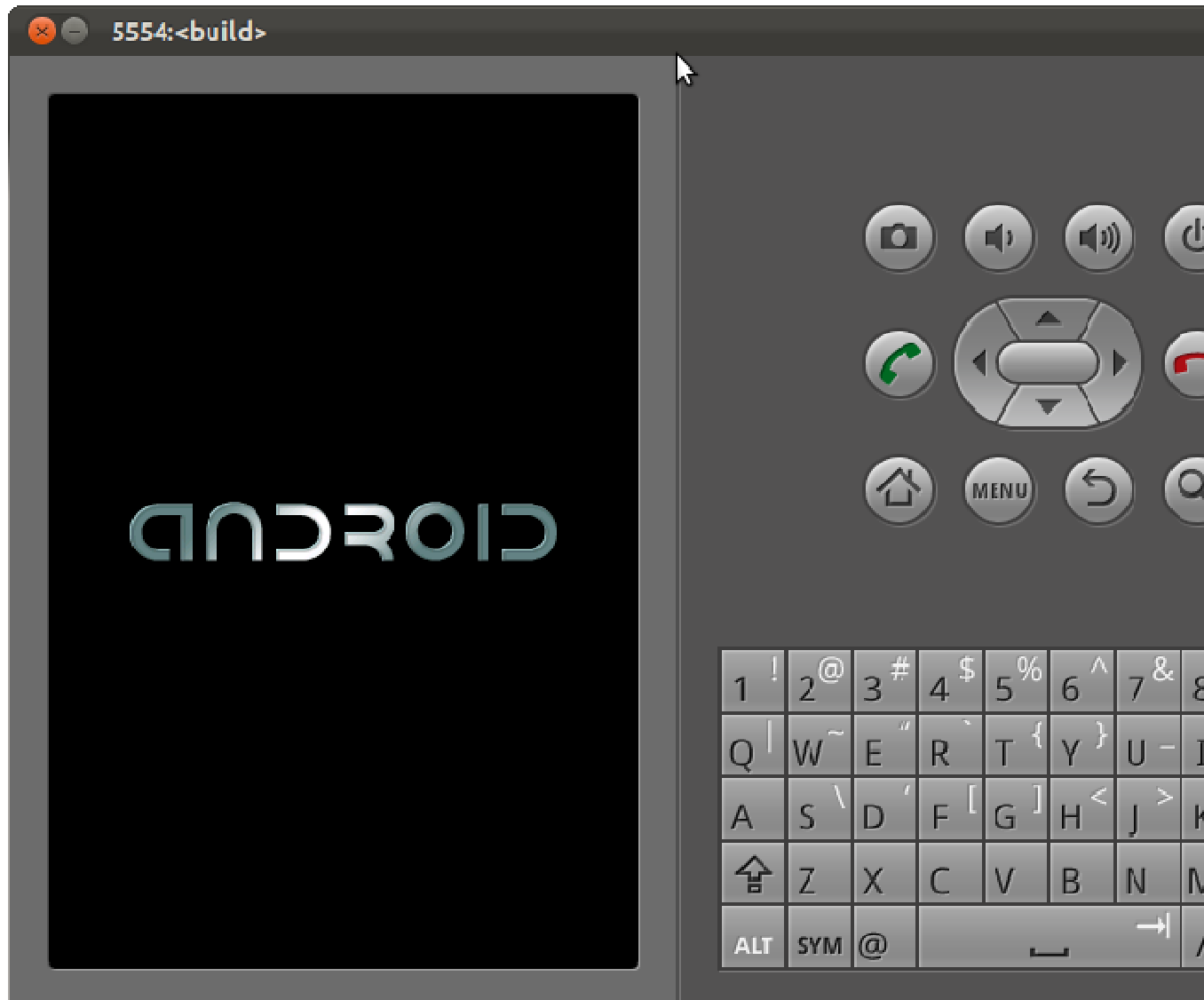
第三步:在模拟器中启动编译好的内核

按下来再运行其下指令:

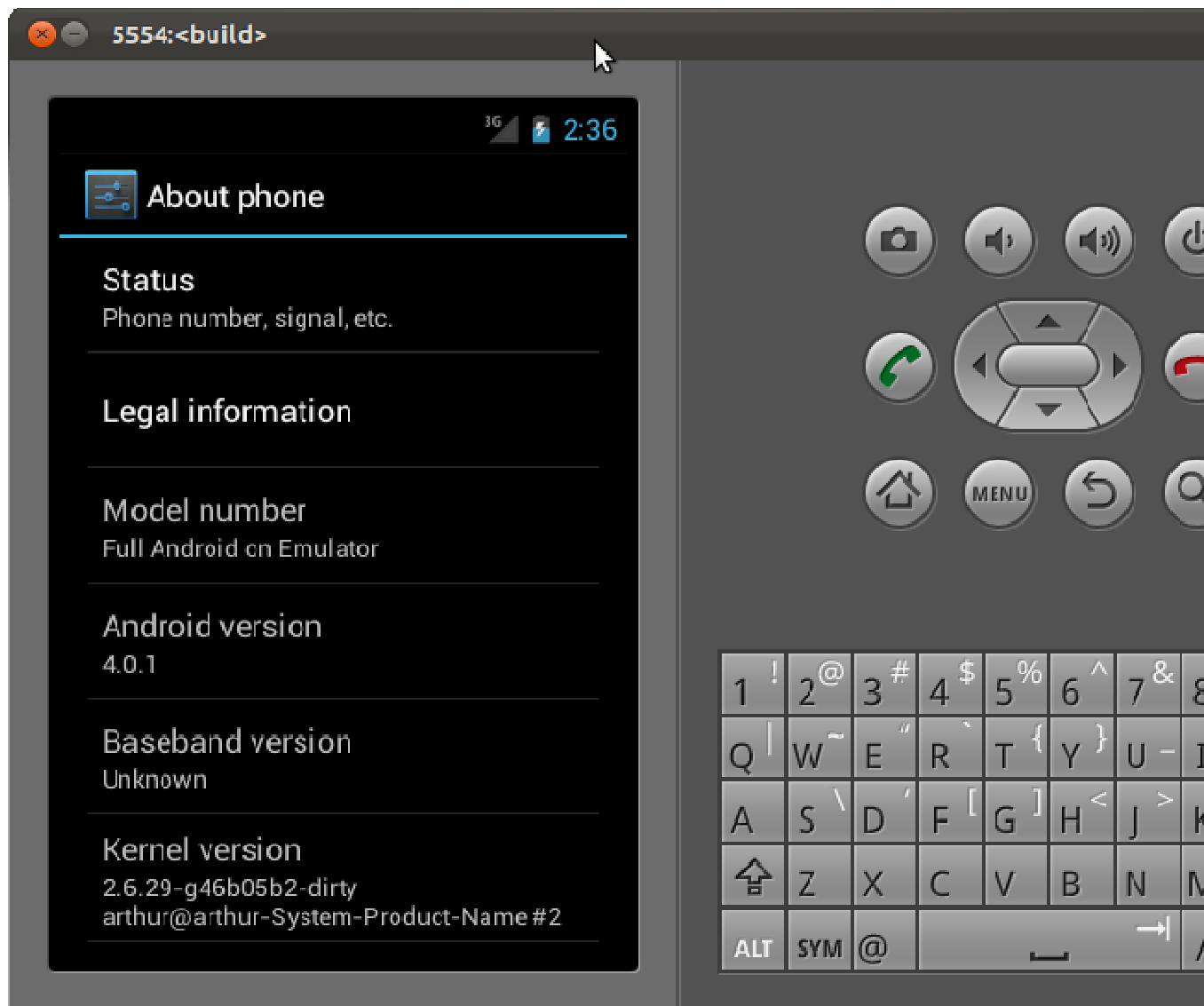
[html] [view plaincopy](#)

1. \$ export PATH=\$PATH:~/WORKING_DIRECTORY/out/host/linux-x86/bin
2. \$ export ANDROID_PRODUCT_OUT=~/WORKING_DIRECTORY/out/target/product/generic
3. \$ emulator -
kernel ~/WORKING_DIRECTORY/kernel/goldfish/arch/arm/boot/zImage
&

模拟器启动界面：



进入模拟器从设置里看版本信息：



从上图可以看出当前 Android 版本是 4.0.1, 内核版本是 2.6.29, 说明成功了.

同样也可以通过 adb shell 来查看内核版本信息, 如下图:

[html] [view plaincopy](#)

1. \$adb shell
2. #cd proc
3. #cat version

```

x _ arthur@arthur-System-Product-Name: ~/WORKING_DIRECTORY/kernel/goldfish
config
d
data
default.prop
dev
etc
init
init.goldfish.rc
init.rc
mnt
proc
root
sbin
sdcard
sys
system
ueventd.goldfish.rc
ueventd.rc
vendor
# cd proc
# cat version
Linux version 2.6.29-g46b05b2-dirty (arthur@arthur-System-Product-Name) (gcc
sion 4.4.3 (GCC) ) #2 Thu Dec 15 10:20:00 CST 2011
#
```