**Before You Start**

**Install Dev Packages**

Install dev packages:

sudo apt-get install gcc-arm-linux-gnueabihf build-essential lzop libncurses5-dev libssl-dev

# If using 64 bits Ubuntu, you need also:

sudo apt-get install libc6:i386

**Install mkbootimg utility**

git clone <https://github.com/neo-technologies/rockchip-mkbootimg.git>

cd rockchip-mkbootimg

make && sudo make install

**Get Kernel Source and Cross Compiling Toolchain**

If you have already downloaded the Firefly-RK3288 Android SDK, the kernel source and cross compiling toolchain are in SDK/kernel and SDK/prebuilts respectively. Please skip to next step.

If not, please download the kernel source code and Android's arm-eabi-4.6 cross compiling toolchain.

Download the kernel source code：

git clone <https://bitbucket.org/T-Firefly/firefly-rk3288-kernel.git>

If you are using Firefly-RK3288-Reload board, please download the "lollipop" branch, and then checkout to "lollipop"

git fetch origin lollipop:lollipop

git checkout lollipop

Note: This is Linux kernel source code in separated source repository, which is extracted from Firefly-RK3288 Android SDK, for convenience of users who only want the kernel source code instead of the huge sdk.

For Android's arm-eabi-4.6 cross compiling toolchain, you may look around to see if any local Android SDK has prebuilts/gcc/linux-x86/arm/arm-eabi-4.6. If there is, you can reuse it; otherwise download from [here](https://android.googlesource.com/platform/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/+/refs/tags/android-4.4.2_r1/) and extract it.

**Compile Kernel**

**Compile Image**

If you are not compiling kernel in the SDK, you need to specify ARCH and CROSS\_COMPILE first:

export ARCH=arm

export CROSS\_COMPILE=/path/to/prebuilts/gcc/linux-x86/arm/arm-eabi-4.6/bin/arm-eabi-

In the kernel source directory, execute:

make firefly-rk3288-linux\_defconfig

make -j8 firefly-rk3288-linux.img

If you are using Firefly-RK3288-Reload board, executeː

make firefly-rk3288-reload-linux\_defconfig

make -j8 firefly-rk3288-reload-linux.img

Note: if you are using the beta development board, please replace "firefly-rk3288.img" above with "firefly-rk3288\_beta.img".

**Compile Modules**

Run:

make modules

mkdir modules\_install

make INSTALL\_MOD\_PATH=./modules\_install modules\_insatll

Copy the kernel module files to the root file system:

rsync -av ./modules\_install/ /path/to/your/rfs/

You can also copy to the development board, using ssh remote connection:

rsync -av ./modules\_install/ root@<Board IP>:/

Clean up the module install directory (it contains links which impact compilation of Android SDK):

rm -rf ./modules\_install

**Create boot.img**

**Create Initramfs**

The kernel will load the initramfs as the first root file system. It is in charge of finding the actual root device, load it then finally switch to it. The development board use eMMC flash. There is no need of special kernel module file. Actually you can skip this step. But initramfs offers flexibility like multiple OS boot.

git clone <https://github.com/TeeFirefly/initrd.git>

make -C initrd

**Package Kernel and Initramfs**

Package kernel and initramfs into boot.img:

mkbootimg --kernel arch/arm/boot/zImage --ramdisk initrd.img -o boot.img

**Modify Parameter File**

The root file system can reside on different partitions in different storage devices (eMMC, TF card or USB disk, etc). It must be specified in the kernel's command line. There is a CMDLINE in the paramter file:

CMDLINE:console=tty0 ... mtdparts=rk29xxnand:0x00002000@0x00002000(uboot),...,-@0x00394000(user)

Add one of below according to your need: (# and contents after are comments, no need to enter):

root=/dev/block/mtd/by-name/linuxroot # flash partition named "linuxroot"

root=/dev/mmcblk0p1 # First partition of TF card

root=/dev/sda1 # First partition of USB disk

root=LABEL=linuxroot # Partition whose label is "linuxroot"

**Flash to Device**

Please reference [Flash image](http://wiki.t-firefly.com/index.php/Firefly-RK3288/Flash_image/en), choose the newly created boot.img and modified parameter files, flash to "boot" and "parameter" partitions respectively. Then the kernel update is done.

If the root filesystem image is not flashed yet, you can download the prebuilt image, or customize your own one, and flash it to the partition specified in parameter file.