

# Linux内核移植

---

## 前言

---

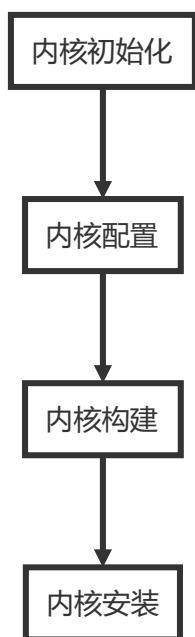
使用的开发板：正点原子开发板

使用的linux内核源码：linux-imx-rel\_imx\_4.1.15\_2.1.0\_ga.tar.bz2

## 简述

---

linux内核移植流程图：



## 内核简介

- Linux内核是由许多逻辑构成要素组成的综合软件
- 由必要的调度程序、文件系统、内存管理器、网络系统等许多子系统组成。
- 通过构建过程生成二进制映像文件zImage--烧写到存储芯片中--通过uboot启动zImage

## 内核初始化

- 为了构建内核，用户必须先完成内核初始化过程
- 通过kernel.org等网站获得tar.gz格式的内核初始源代码
- 内核源代码解压后会内核初始状态的内核源代码树
- make mrproper命令回退，make distclean命令初始化
- 不能立即对内核进行编译，否则会引起内核严重错误。

## 内核配置

- 内核配置过程也是适当选择与自身系统相吻合的各种内核要素的过程。如我们使用的正点原子开发板与NXP官方开发板存在差别，需要进行一些配置。
- 使用menuconfig对象进行配置。make menuconfig提供一种基于文本的图形界面，如果使用该对象，需要安装两个软件包build-essential和libncurses5-dev。

- 配置后会生成.config文件，文件内形式为CONFIG\_XXX (y, n或m)，也称为kconfig

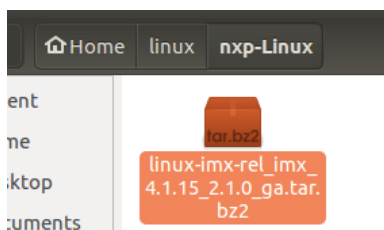
## 内核构建与编译

- 利用kconfig完成内核配置，并准备好具有自身内核配置的.config文件后，即可构建内核（编译内核并链接二进制文件，由此生成一个二进制文件zImage的一系列过程）。
- 通过顶层的Makefile文件进行编译
- 在Ubuntu环境下，利用make -kpkg命令完成内核的编译

## 步骤

### 1.内核初始化

1. 在windows中准备好NXP（恩智浦）的linux内核源码。使用FileZilla传输到 linux 系统中。这里，我们将新建了一个名为npx-Linux的文件夹，将内核放到了这个文件夹下。



2. 打开终端，进入/linux/nxp-Linux目录。

```
flower@flower-VirtualBox:~$ cd linux
flower@flower-VirtualBox:~/linux$ cd nxp-Linux
flower@flower-VirtualBox:~/linux/nxp-Linux$ ls
linux-imx-rel_imx_4.1.15_2.1.0_ga.tar.bz2
flower@flower-VirtualBox:~/linux/nxp-Linux$
```

将源码进行

解压操作。输入 `tar -jxvf linux-imx-rel_imx_4.1.15_2.1.0_ga.tar.bz2` 命令进行解压。

```
flower@flower-VirtualBox:~/linux/nxp-Linux$ tar -jxvf linux-imx-rel_imx_4.1.15_2.1.0_ga.tar.bz2
```

```
flower@flower-VirtualBox:~/linux/nxp-Linux$ ls
linux-imx-rel_imx_4.1.15_2.1.0_ga  linux-imx-rel_imx_4.1.15_2.1.0_ga.tar.bz2
```

3. 安装相关包。输入 `sudo apt install lzop`，安装这个包后才能成功编译linux内核。

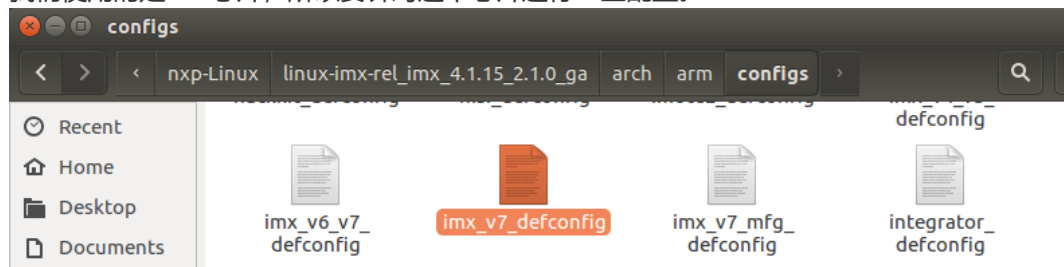
```
flower@flower-VirtualBox:~/linux/nxp-Linux$ sudo apt install lzop
[sudo] password for flower:
```

4. 输入 `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean` 命令进行初始化操作

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga$ make
make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- distclean
CLEAN    scripts/basic
CLEAN    scripts/kconfig
CLEAN    include/config include/generated
```

### 2.内核配置

1. 我们使用的是imx芯片，所以要针对这个芯片进行一些配置。



输入 `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- imx_v7_defconfig` 命令进行配置

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga$ make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- imx_v7_defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
SHIPPED scripts/kconfig/zconf.tab.c
SHIPPED scripts/kconfig/zconf.lex.c
SHIPPED scripts/kconfig/zconf.hash.c
HOSTCC scripts/kconfig/zconf.tab.o
HOSTLD scripts/kconfig/conf
#
# configuration written to .config
#
```

2. NPX Linux内核源码的结构如下图：

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga$ ls
arch      Documentation  init          lib           README        sound
block     drivers        ipc           MAINTAINERS  REPORTING-BUGS tools
COPYING   firmware      Kbuild       Makefile     samples       usr
CREDITS   fs             Kconfig      mm           scripts       virt
crypto    include        kernel        net          security
```

各文件夹及其功能内容如下表：

文件夹名称	功能				
arch	CPU相关信息	drivers	内置驱动	include	头文件
block	块设备相关	firmware	硬件管理	init	内核启动处理
Documentation	文本文档	fs	文件系统	ipc	进程间通信相关
mm	内存管理	net	网络相关	security	内核安全
sound	声音相关	tools	内核交互工具	samples	模块代码

3. 使用的开发板为**正点原子**开发板，需要修改一些NXP官方linux源码配置

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga$ cd arch/arm/configs
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga/arch/arm/configs$
```

输入 `cp imx_v7_mfg_defconfig imx_alientek_emmc_defconfig` 命令，拷贝 `imx_v7` 的配置文件并命名为 `imx_alientek_emmc_defconfig`，以后 `imx_alientek_emmc_defconfig` 就是正点原子的EMMC版开发板默认配置文件了。

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga/arch/arm/configs$ cp imx_v7_mfg_defconfig imx_alientek_emmc_defconfig
```

4. 添加开发板对应的设备树。跳转到 `boot/dts`，输入 `cp imx6ull-14x14-evk.dts imx6ull-alientek-emmc.dts` 命令，拷贝 `imx6ull-14x14` 的设备树并命名为 `imx6ull-alientek-emmc.dts`

```
flower@flower-VirtualBox:~/linux/nxp-Linux/linux-imx-rel_imx_4.1.15_2.1.0_ga/arch/arm/boot/dts$ cp imx6ull-14x14-evk.dts imx6ull-alientek-emmc.dts
```

5. 使用VS Code打开解压后的 `linux-imx-rel_imx_4.1.15_2.1.0_ga` 文件夹。由于之前新添加了 `imx_alientek_emmc_defconfig` 配置文件和对应的设备树需要修改设备树有关的 `Makefile`。`.dts`是设备树源码文件，编译Linux的时候会将其编译为`.dtb`文件。在 `arch/arm/boot/dtc`中找到`Makefile`文件并在VS Code中打开。添加以下代码 `imx6ull-alientek-emmc.dtb`（`.dts`是设备树源码文件，编译Linux的时候会将其编译为`.dtb`文件。）添加

完毕后，linux内核在编译的时候才能从imx6ull-alientek-emmc.dts编译出imx6ull-alientek-

emmc.dtb文件。

```
Makefile
arch > arm > boot > dts > Makefile
415     imx6ull-14x14-ddr3-arm2-usb.dtb \
416     imx6ull-14x14-ddr3-arm2-wm8958.dtb \
417     imx6ull-14x14-evk.dtb \
418     imx6ull-14x14-evk-btwifi.dtb \
419     imx6ull-14x14-evk-emmc.dtb \
420     imx6ull-14x14-evk-gpmi-weim.dtb \
421     imx6ull-14x14-evk-usb-certi.dtb \
422     imx6ull-alientek-emmc.dtb \
423     imx6ull-9x9-evk.dtb \
424     imx6ull-9x9-evk-btwifi.dtb \
425     imx6ull-9x9-evk-ldo.dtb
```

至此，设备树相关工作配置完成。

6. 我们的开发板eMMC作为flash闪存，而nxp官方开发板使用4根线连接CPU与闪存芯片，正点原子采用8根线，传送速度更快，所以需要更改闪存相关驱动配置。

打开我们刚才创建的设备树文件 imx6ull-alientek-emmc.dts，添加如下代码

```
734 &usdhc2 {
735     pinctrl-names = "default", "state_100mhz", "state_200mhz";
736     pinctrl-0 = <&pinctrl_usdhc2_8bit>;
737     pinctrl-1 = <&pinctrl_usdhc2_8bit_100mhz>;
738     pinctrl-2 = <&pinctrl_usdhc2_8bit_200mhz>;
739     bus-width = <8>;
740     non-removable;
741     status = "okay";
742 };
```

总线宽度设置为8。

7. 修改网络驱动。正点原子开发板与NXP官方的物理芯片不一样，复位信号也不一样。

其中引脚7和引脚8是不同的，需要进行修改。

- 首先查看两个引脚有没有被其他的地方所用过，发现在 spi4grp 中用到了这样两个引脚，将这两行代码删除掉。

```
584     pinctrl_spi4: spi4grp {
585         fsl,pins = <
586             MX6ULL_PAD_BOOT_MODE0_GPIO5_I010    0x70a1
587             MX6ULL_PAD_BOOT_MODE1_GPIO5_I011    0x70a1
588             //MX6ULL_PAD_SNVS_TAMPER7_GPIO5_I007    0x70a1
589             //MX6ULL_PAD_SNVS_TAMPER8_GPIO5_I008    0x80000000
590         >;
591     };
```

- 找到 spi4 中关于两个引脚的代码，删除

```
spi4 {
    compatible = "spi-gpio";
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_spi4>;
    pinctrl-assert-gpios = <&gpio5 8 GPIO_ACTIVE_LOW>;
    status = "okay";
    gpio-sck = <&gpio5 11 0>;
    gpio-mosi = <&gpio5 10 0>;
    cs-gpios = <&gpio5 7 0>;
    num-chipselects = <1>;
    #address-cells = <1>;
    #size-cells = <0>;
}
```

- 修改enet1和enet2初始化的情况，添加一些复位信息

```
pinctrl_enet1:enet1grp {
    fsl,pins = <
        MX6UL_PAD_ENET1_RX_EN_ENET1_RX_EN 0x1b0b0
        MX6UL_PAD_ENET1_RX_ER_ENET1_RX_ER 0x1b0b0
        MX6UL_PAD_ENET1_RX_DATA0_ENET1_RDATA00 0x1b0b0
        MX6UL_PAD_ENET1_RX_DATA1_ENET1_RDATA01 0x1b0b0
        MX6UL_PAD_ENET1_TX_EN_ENET1_TX_EN 0x1b0b0
        MX6UL_PAD_ENET1_TX_DATA0_ENET1_TDATA00 0x1b0b0
        MX6UL_PAD_ENET1_TX_DATA1_ENET1_TDATA01 0x1b0b0
        MX6UL_PAD_ENET1_TX_CLK_ENET1_REF_CLK1 0x4001b031
        MX6ULL_PAD_SNVS_TAMPER7_GPI05_I007 0x10B0
    >;
};

pinctrl_enet2:enet2grp {
    fsl,pins = <
        MX6UL_PAD_GPI01_I007_ENET2_MDC 0x1b0b0
        MX6UL_PAD_GPI01_I006_ENET2_MDIO 0x1b0b0
        MX6UL_PAD_ENET2_RX_EN_ENET2_RX_EN 0x1b0b0
        MX6UL_PAD_ENET2_RX_ER_ENET2_RX_ER 0x1b0b0
        MX6UL_PAD_ENET2_RX_DATA0_ENET2_RDATA00 0x1b0b0
        MX6UL_PAD_ENET2_RX_DATA1_ENET2_RDATA01 0x1b0b0
        MX6UL_PAD_ENET2_TX_EN_ENET2_TX_EN 0x1b0b0
        MX6UL_PAD_ENET2_TX_DATA0_ENET2_TDATA00 0x1b0b0
        MX6UL_PAD_ENET2_TX_DATA1_ENET2_TDATA01 0x1b0b0
        MX6UL_PAD_ENET2_TX_CLK_ENET2_REF_CLK2 0x4001b031
        MX6ULL_PAD_SNVS_TAMPER8_GPI05_I008 0x10B0
    >;
};
```

- 修改物理芯片信息。

```
&fec1 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_enet1>;
    phy-mode = "rmii";
    phy-handle = <&ethphy0>;
    phy-reset-gpios = <&gpio5 7 GPIO_ACTIVE_LOW>;
    phy-reset-duration = <200>;
    status = "okay";
};

&fec2 {
    pinctrl-names = "default";
    pinctrl-0 = <&pinctrl_enet2>;
    phy-mode = "rmii";
    phy-handle = <&ethphy1>;
    status = "okay";
    phy-reset-gpios = <&gpio5 8 GPIO_ACTIVE_LOW>;
    phy-reset-duration = <200>;
};
```

添加复位引脚信息，设置低电平有效，持续时间200ms

- 在下面的mido中添加物理地址信息

```
ethphy0: ethernet-phy@2 {
    compatible = "ethernet-phy-ieee802.3-c22";
    smsc, disable-energy-detect;
    reg = <0>;
};

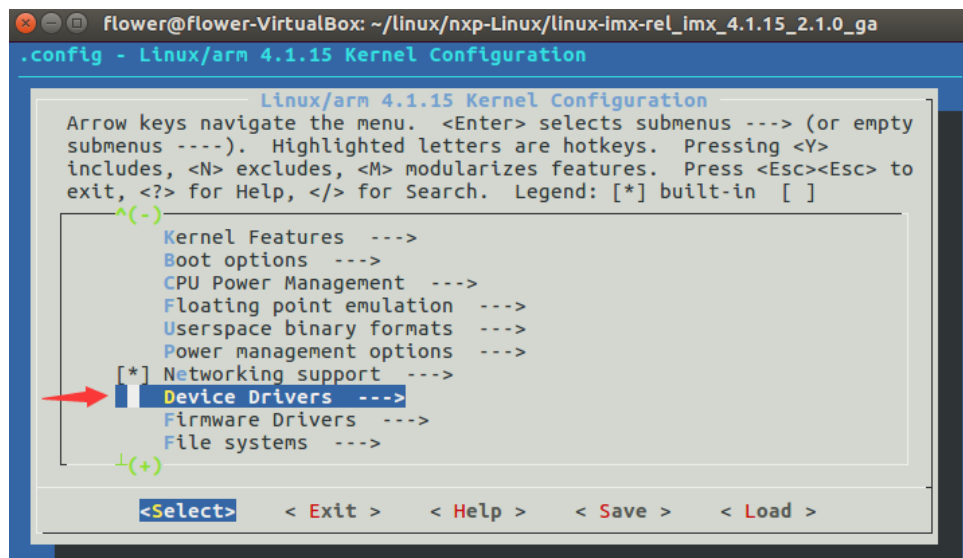
ethphy1: ethernet-phy@1 {
    compatible = "ethernet-phy-ieee802.3-c22";
    smsc, disable-energy-detect;
    reg = <1>;
};
```

- 引脚信息修改完之后，修改驱动信息。打

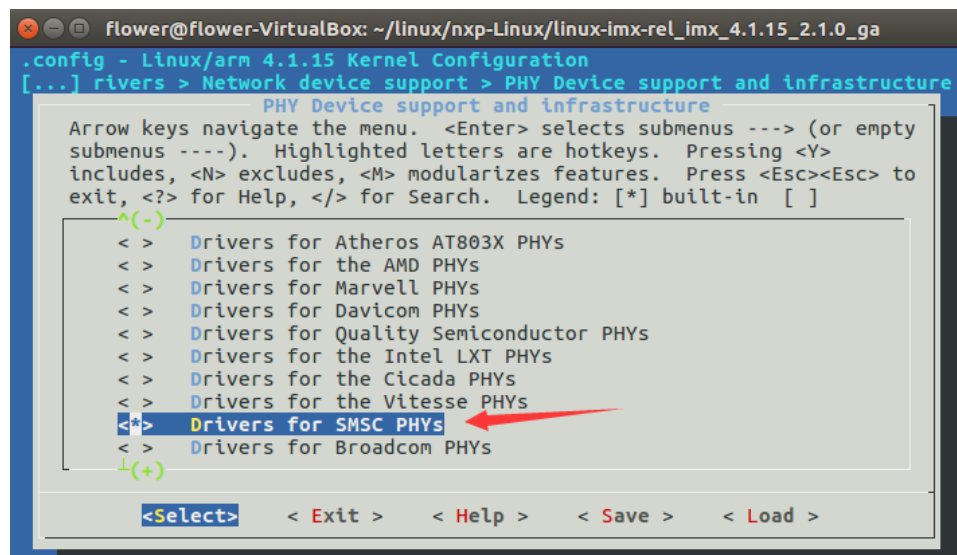
开 `/drivers/net/ethernet/freescale/fec_main.c`，找到 `fec_probe` 函数（3439行），添加软复位代码。

```
3452 void __iomem *IMX6UL_ENET1_TX_CLK;
3453 void __iomem *IMX6UL_ENET2_TX_CLK;
3454
3455 IMX6UL_ENET1_TX_CLK = ioremap(0x020EB0DC, 4);
3456 write(0x14, IMX6UL_ENET1_TX_CLK);
3457
3458 IMX6UL_ENET2_TX_CLK = ioremap(0x020EB0FC, 4);
3459 write(0x14, IMX6UL_ENET2_TX_CLK);
3460
```

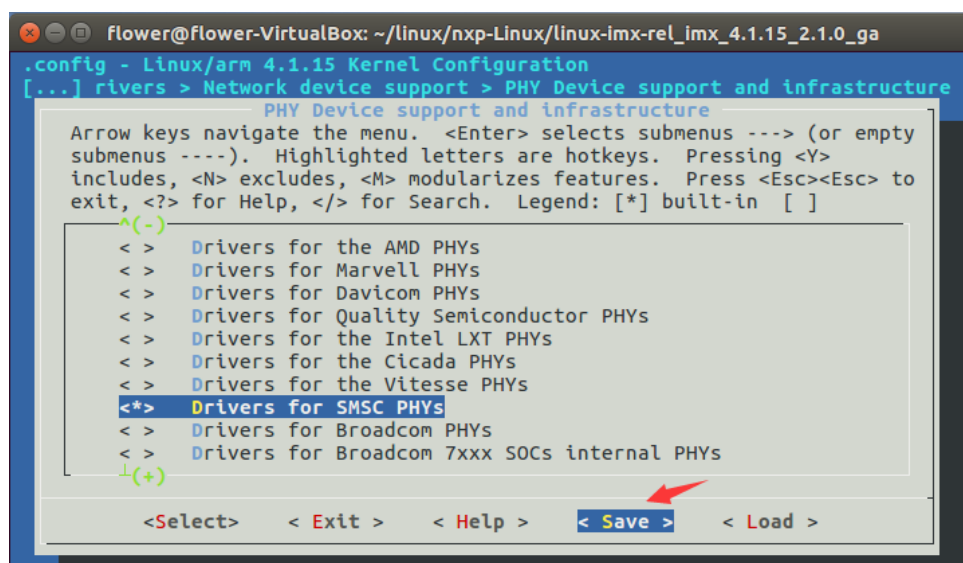
- 代码部分配置好后，需要在config中配置驱动。输入 `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig` 命令，打开界面配置。通过上下箭头键找到 `Device Drivers`，按回车进入。

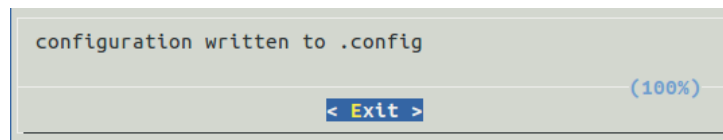


找到 `Network device support> PHY Device support and infrastructure> Drivers for SMSC PHYs` 按y选中



之后按一下回车，在下方使用左右箭头移动到 `save`，回车，保存并退出





这样网络驱动就配置成功了。

### 3.内核编译

输入 `make ARCH=arm CROSS_COMPILE=arm-linux-gnueabi- all -j16` 命令进行编译。等待一段时间后，编译完成。

linux内核移植过程结束。