



嵌入式系统 课程实验

第二、三篇实验



1 开发板启动方式

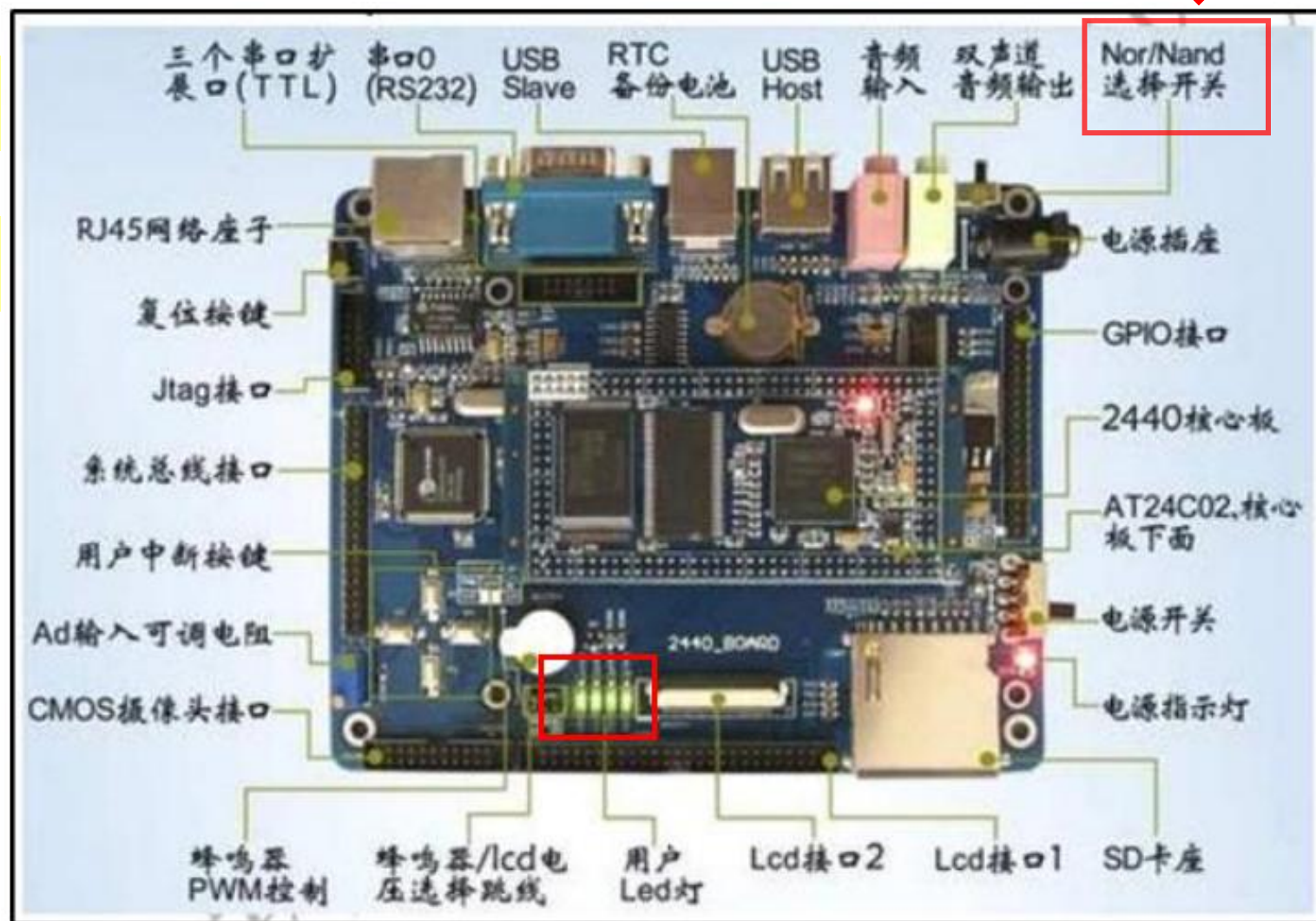
2 第三次实验

3 第四次实验

4 实验报告及验收标准

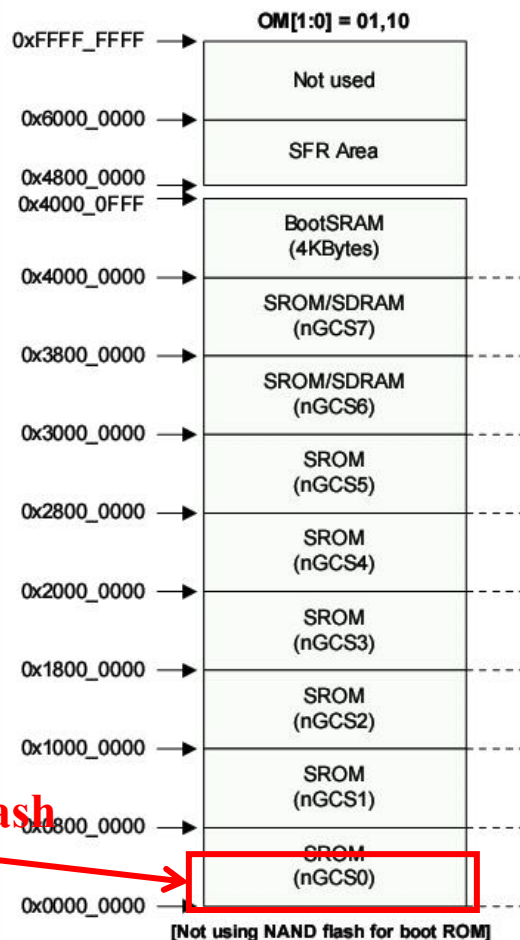
1. 开发板启动方式

- 从 NOR FLASH 启动
- 从 NAND FLASH 启动



1.1 从 NOR FLASH 启动

- 启动 ROM 就要定位于内存的起始地址空间(nGCS0) 0x00000000, 处理器直接在 ROM 上运行启动程序。



```
##### Boot for Nor Flash Main Menu #####
##### EmbedSky USB download mode #####

[1] Download u-boot or STEPLDR.nb1 or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program (eg: uCOS-II or TQ2440_Test)
[b] Download LOGO Picture (.bin) to Nand Flash
[l] Set LCD Parameters
[n] Enter TFTP download mode menu
[o] Download u-boot to Nor Flash
[r] Reboot u-boot
[t] Test Linux Image (zImage)
[q] quit from menu
Enter your selection:
```



```
#####      Boot for Nor Flash Main Menu      #####
#####      EmbedSky USB download mode        #####

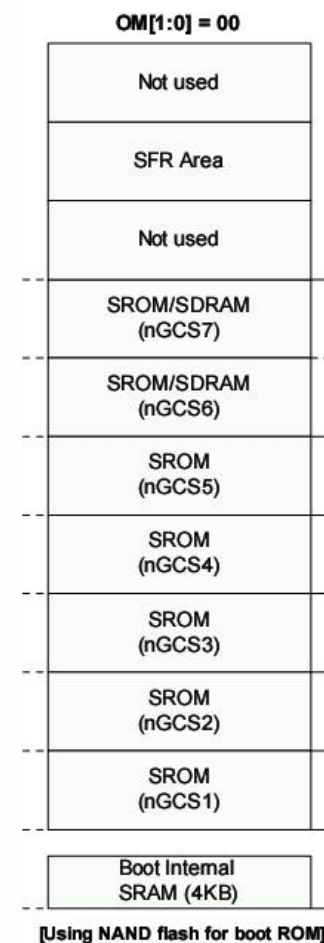
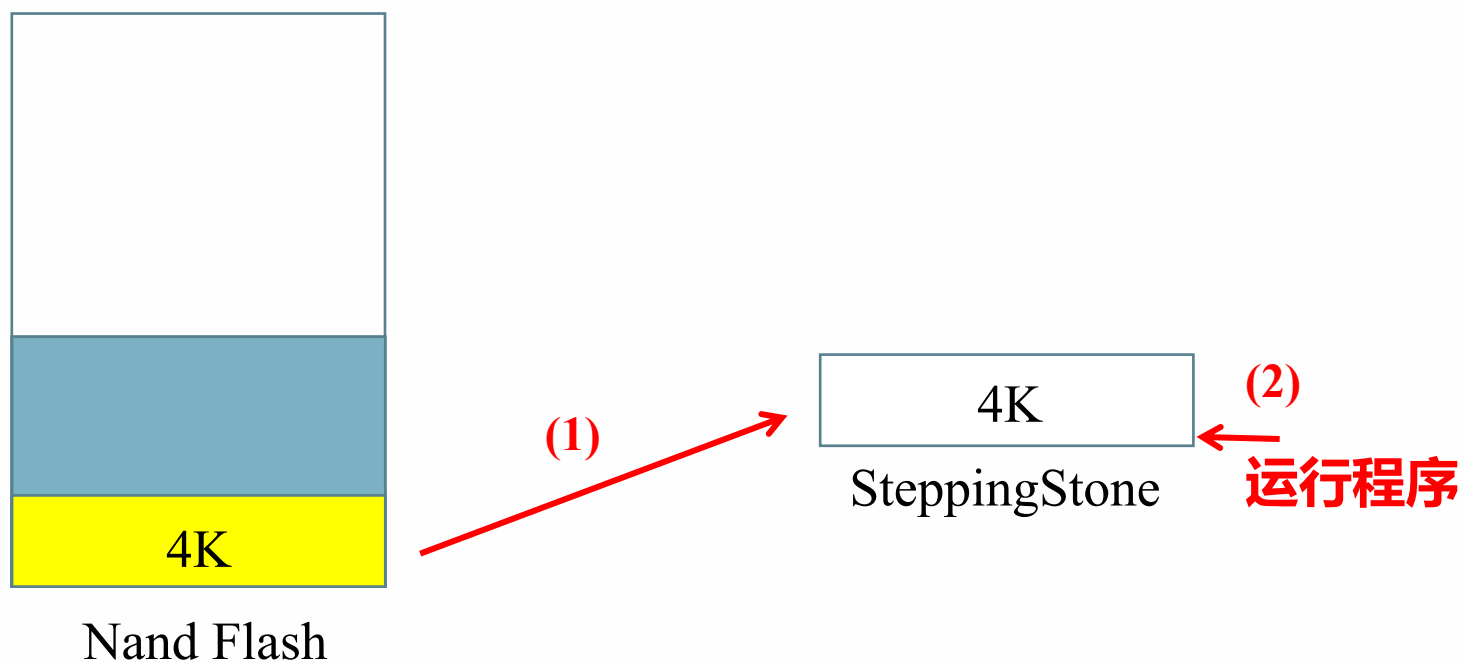
[1] Download u-boot or STEPLDR.nb1 or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
[a] Download User Program (eg: uCOS-II or TQ2440_Test)
[b] Download LOGO Picture (.bin) to Nand Flash
[l] Set LCD Parameters
[n] Enter TFTP download mode menu
[o] Download u-boot to Nor Flash
[r] Reboot u-boot
[t] Test Linux Image (zImage)
[q] quit from menu
Enter your selection:
```

实验一~三代码下载到哪里？
从哪里开始运行？

实验四代码下载到哪里？
从哪里开始运行？

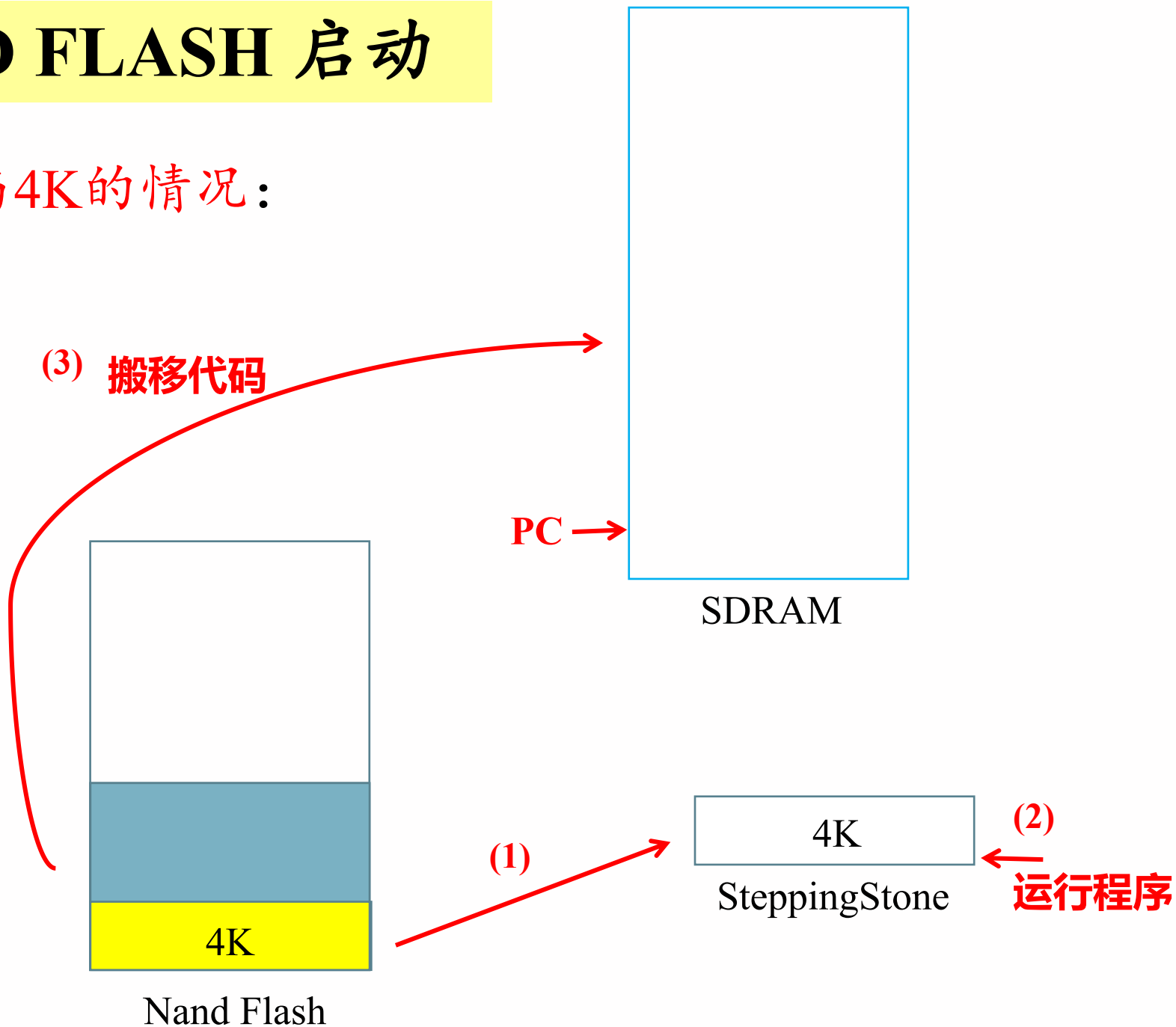
1.2 从 NAND FLASH 启动

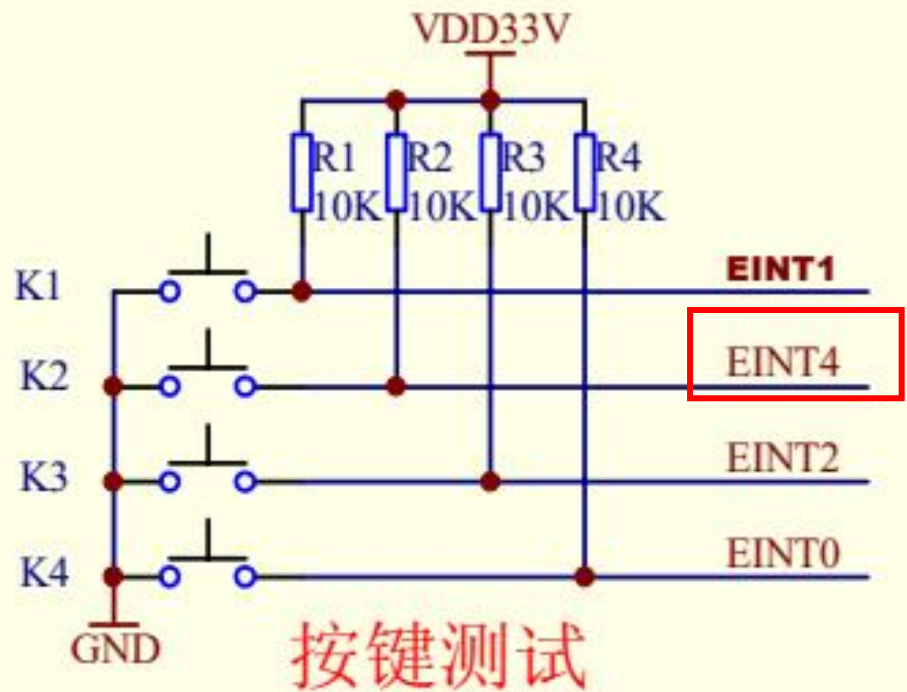
- S3C2410/2440配备了称为Steppingstone的内置SRAM，Steppingstone映射到地址空间(nGCS0)0x00000000，大小为4K。上电后NAND FLASH的前4K代码被拷贝至Steppingstone中，并开始运行。
- 代码小与4K的情况：



从 NAND FLASH 启动

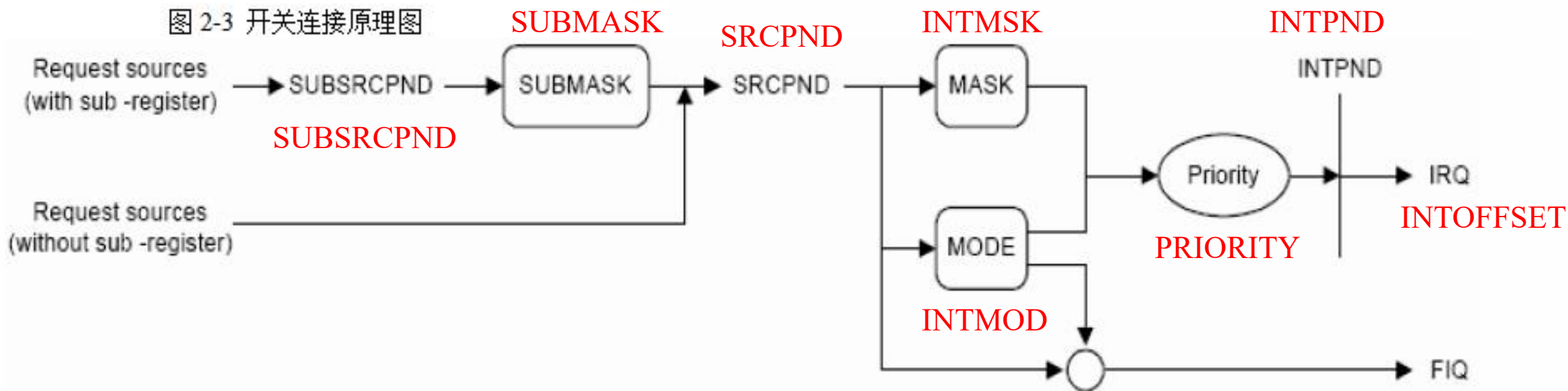
- 代码大与4K的情况:





EINT8_23	External interrupt 8 – 23	ARB1
EINT4_7	External interrupt 4 – 7	ARB1
EINT3	External interrupt 3	ARB0
EINT2	External interrupt 2	ARB0
EINT1	External interrupt 1	ARB0
EINT0	External interrupt 0	ARB0

图 2-3 开关连接原理图



2. 第三次实验

实验指导书中第二篇实验五~实验十

第二篇 Linux 环境下裸机开发实验.....	21
实验五 熟悉 Linux 操作系统.....	21
实验六 Linux 常用命令的熟悉.....	25
实验七 文本编辑器 Vi 的熟悉.....	26
实验八 编译工具 GCC 的使用.....	31
实验九 交叉编译工具的使用.....	34
实验十 Linux 环境下裸机程序设计及编译.....	36

2.1 Linux常用命令

- Linux命令行的一般格式为：

命令名 [选择项] [参数]

注意：Linux终端下的命令区分大小写，而windows系统命令行不区分大小写。

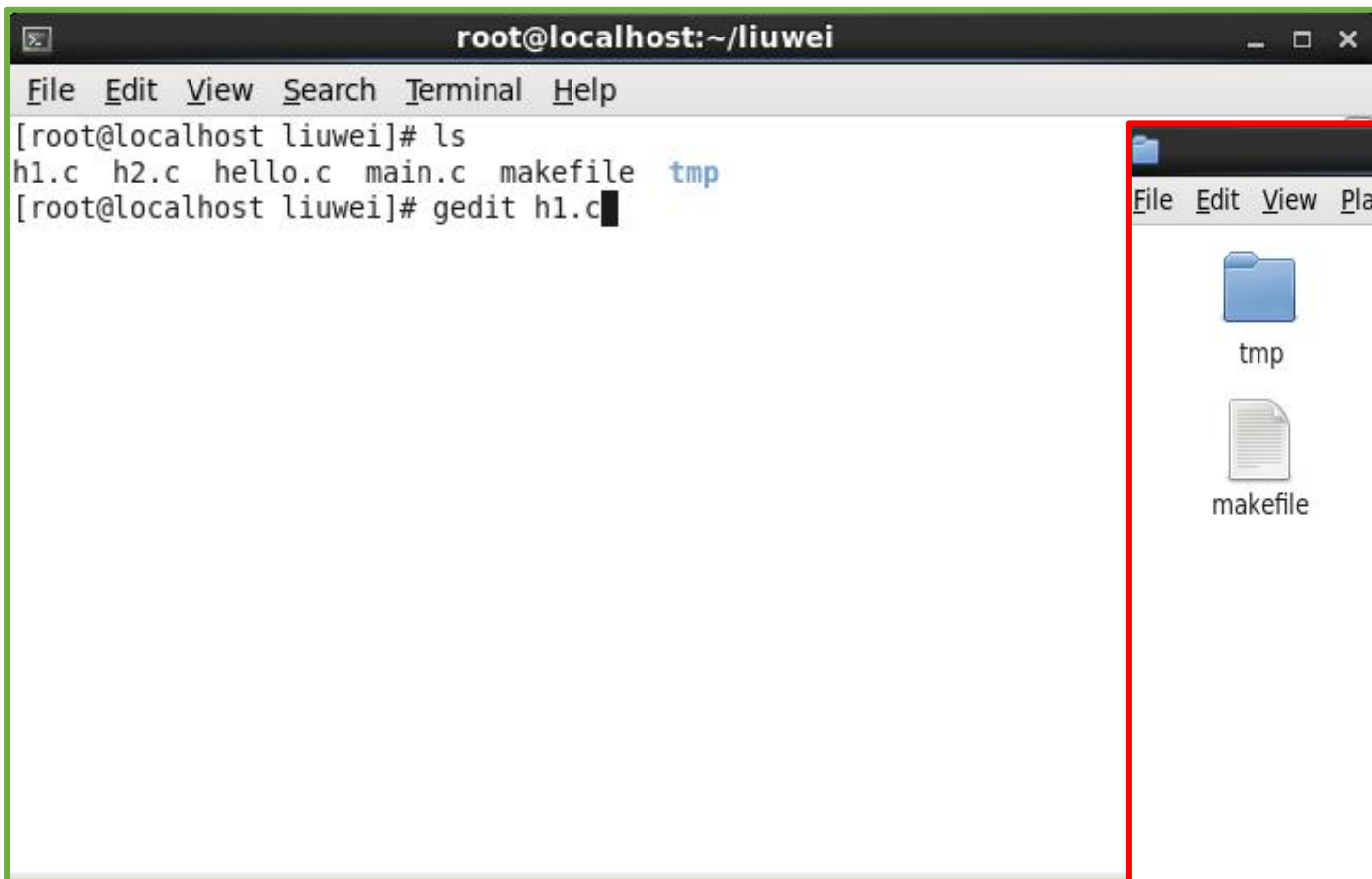
- 在终端下练习以下命令（参考Linux命令全集.chm）：

- ✓ **pwd**: 显示工作目录
- ✓ **cd**: 切换工作目录
- ✓ **ls**: 列出目录内容
- ✓ **mkdir**: 创建新目录
- ✓ **cp**: 复制文件或目录
- ✓ **mv**: 移动或更名现有的文件或目录
- ✓ **rm**: 删除文件或目录

2.2 文本编辑器 gedit 和 Vi 的使用

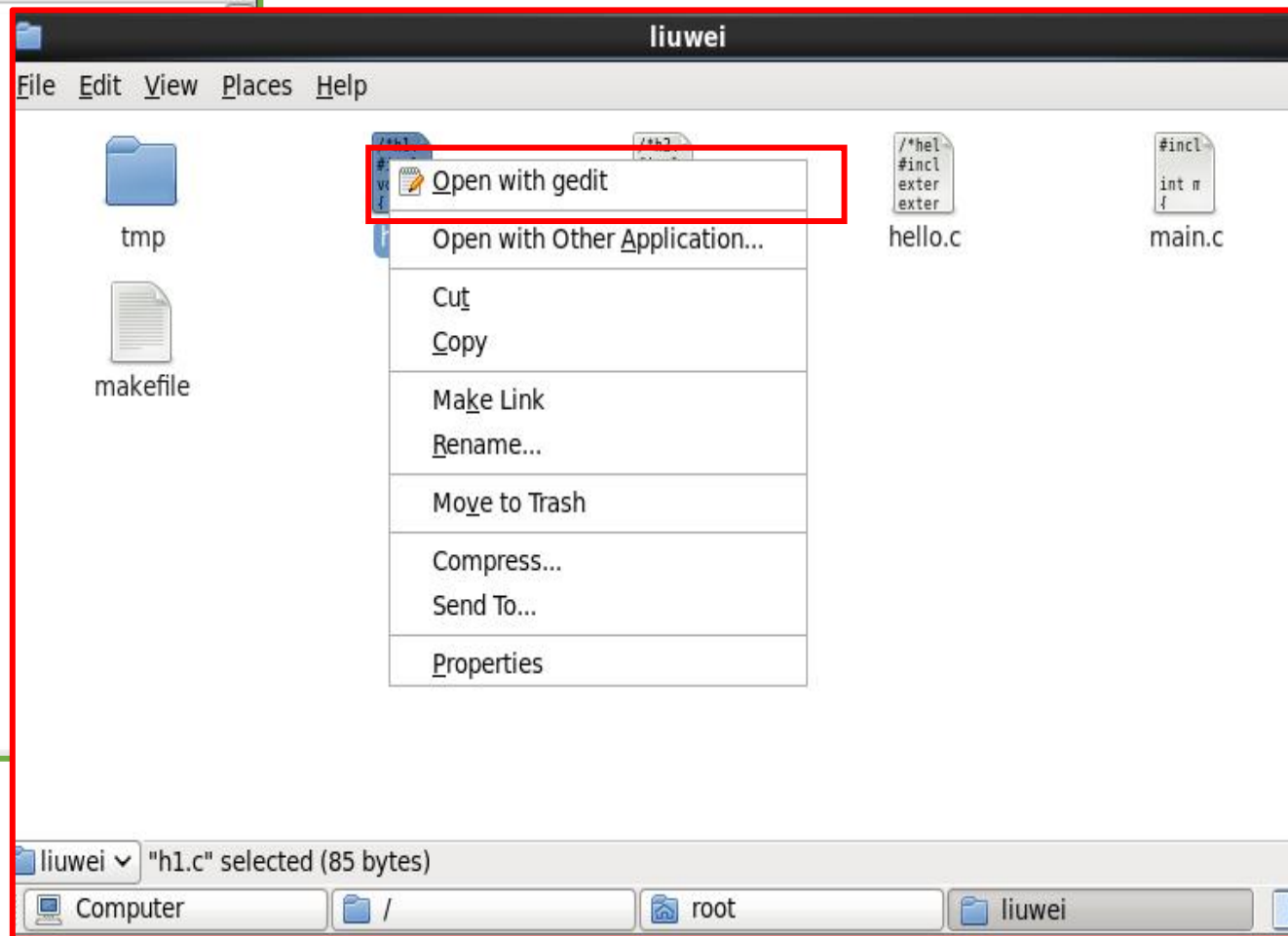
- 文本编辑器 gedit

桌面打开



A terminal window titled 'root@localhost:~/liuwei'. The command prompt shows the user has run 'ls' and 'gedit h1.c'. The file 'h1.c' is highlighted in blue in the 'ls' output. The 'gedit h1.c' command is entered at the prompt.

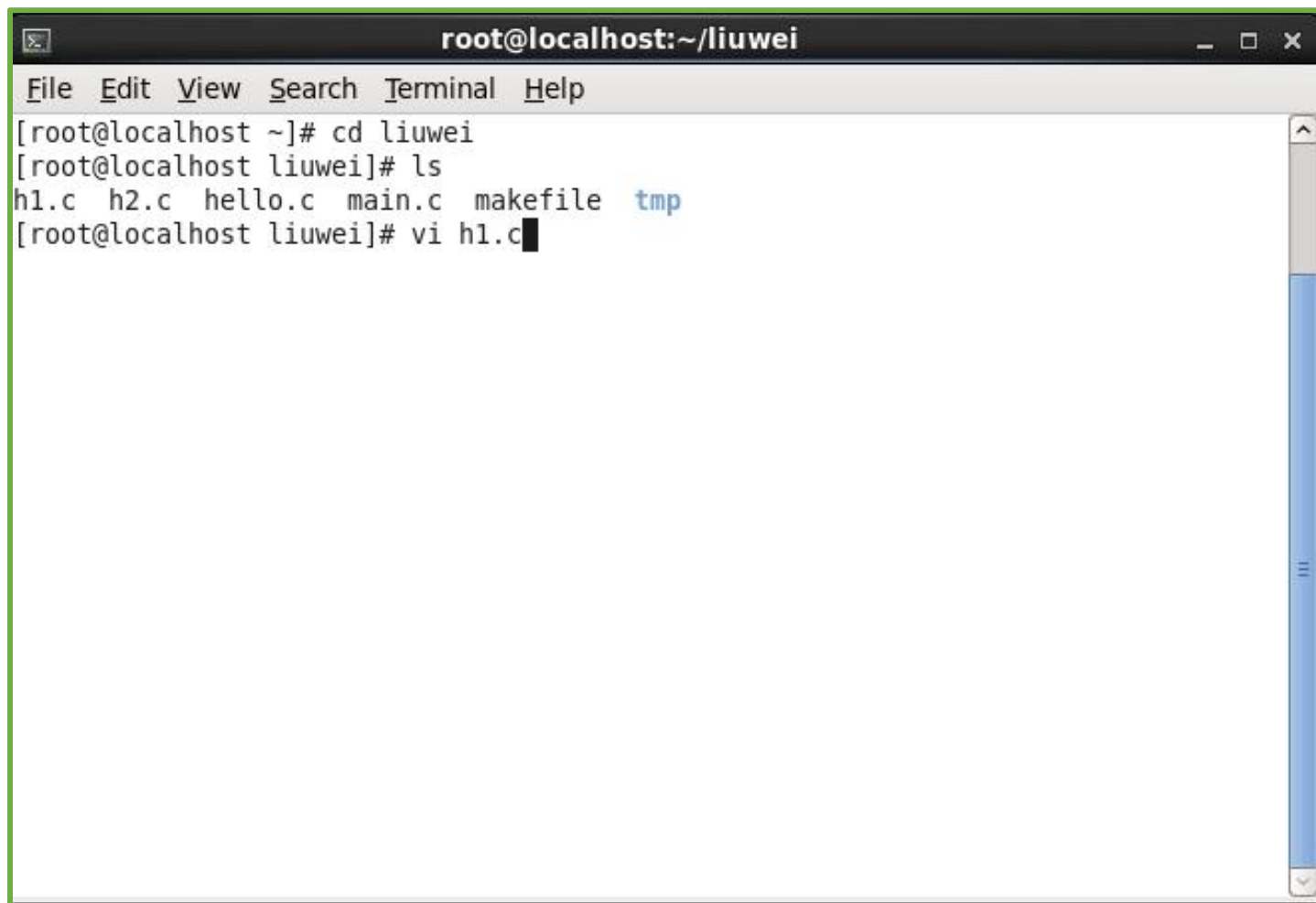
```
root@localhost:~/liuwei
File Edit View Search Terminal Help
[root@localhost liuwei]# ls
h1.c h2.c hello.c main.c makefile tmp
[root@localhost liuwei]# gedit h1.c
```



命令行打开

2.2 文本编辑器 gedit 和 Vi 的使用

- 文本编辑器 vi



```
root@localhost:~/liuwei
File Edit View Search Terminal Help
[root@localhost ~]# cd liuwei
[root@localhost liuwei]# ls
h1.c h2.c hello.c main.c makefile tmp
[root@localhost liuwei]# vi h1.c
```

命令行打开

• vi的三种模式

■ Command Mode (命令模式)

用户在使用vi编辑文件时，最初进入的模式。在该模式中可以通过上下移动光标进行删除字符或整行删除等操作，也可以进行复制、粘贴等操作，但无法编辑文件。用户可按a、o、i键进入插入模式。按冒号进入末行模式。

■ Input Mode (插入模式)

只用在该模式下，用户才能进行文字编辑输入，用户可按ESC键返回命令行模式。

■ Last Mode (末行模式)

在命令行模式下，用户按冒号可进入底行模式。此时光标位于屏幕的底行。用户可以进行保存或退出操作，也可以设置编辑环境，如寻找字符串、列出行号等。

练习：分别使用gedit和vi编辑c源文件：

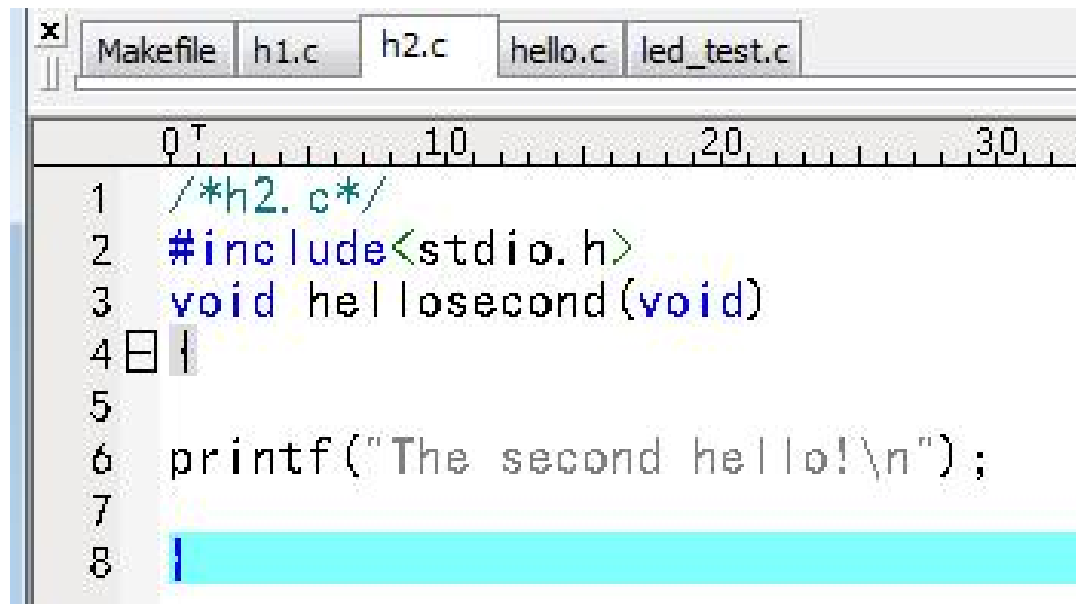
h1.c

h2.c

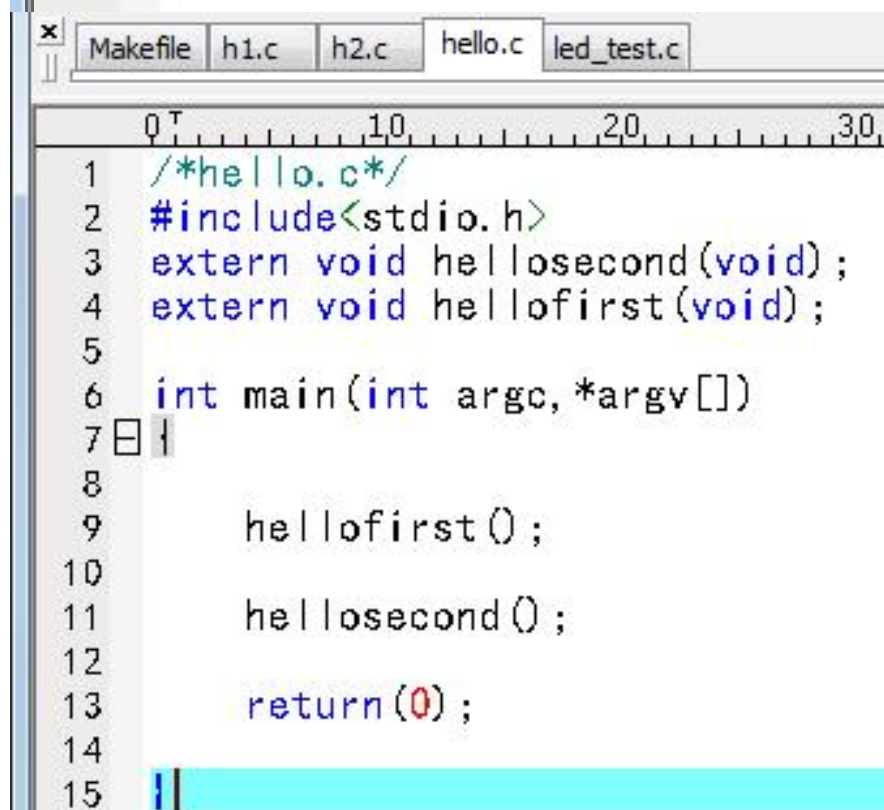
hello.c



```
1 /*h1.c*/
2 #include<stdio.h>
3 void hellofirst(void)
4 {
5
6 printf("The first hello!\n");
7
8 |
```



```
1 /*h2.c*/
2 #include<stdio.h>
3 void hellosecond(void)
4 {
5
6 printf("The second hello!\n");
7
8 |
```



```
1 /*hello.c*/
2 #include<stdio.h>
3 extern void hellosecond(void);
4 extern void hellofirst(void);
5
6 int main(int argc, *argv[])
7 {
8
9 hellofirst();
10
11 hellosecond();
12
13 return(0);
14
15 |
```

2.3 编译工具GCC

GGC 的使用

- 编译工具gcc
- 基本语法

`gcc [options] [filenames]`

其中options: 编译器所需要的编译选项

filenames: 要编译的文件名

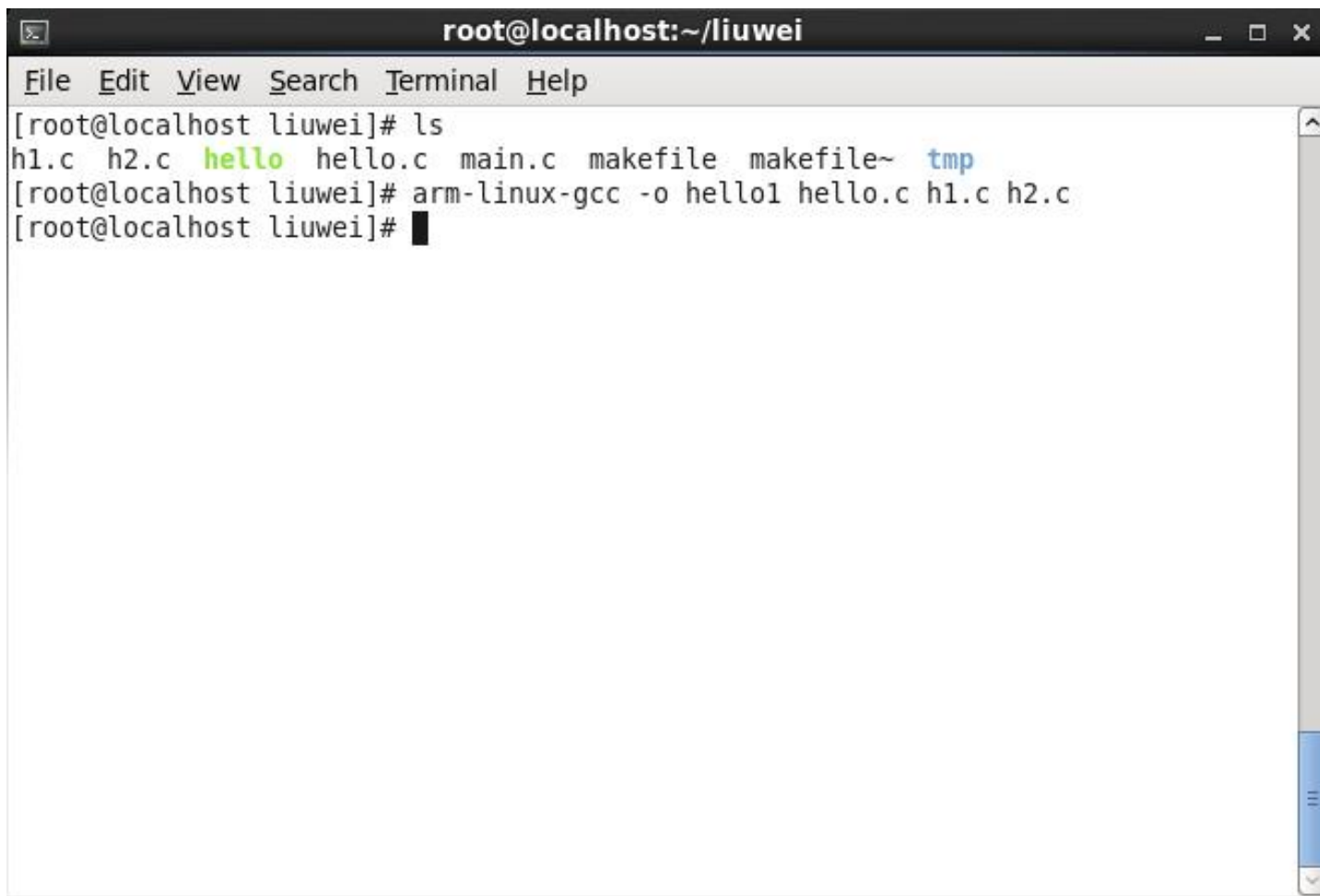
- 说明:
 - Gcc编译选项有一百多个, 在gcc后面可以有多个不冲突的编译选项, 同时进行多个编译操作。很多的gcc选项包括一个以上的字符。
 - 当你不用任何选项编译一个程序时, GCC将会建立(假定编译成功)一个名为a.out的可执行文件。
- 可通过以下命令了解所有选项:
 - man gcc
 - Info gcc

2.4 交叉编译工具arm-linux-gcc

- arm-linux-gcc是在Linux下生成arm开发板可执行的应用程序。

我们之前使用的ADS编译源文件，同样也属于交叉编译。

其使用同gcc



```
root@localhost:~/liuwei
File Edit View Search Terminal Help
[root@localhost liuwei]# ls
h1.c h2.c hello hello.c main.c makefile makefile~ tmp
[root@localhost liuwei]# arm-linux-gcc -o hello1 hello.c h1.c h2.c
[root@localhost liuwei]#
```

2.5 工程编译工具make/makefile

GNU make项目管理

Make+Makefile提供了一种简单有效的工程管理方式。

- 决定编译哪些文件
- 怎样编译这些文件
- 按什么顺序连接这些文件

GNU makefile

- **Makefile**: 一个决定怎样编译工程的文本文件。按照一定的规则书写
- **make工具**: 根据当前的makefile进行工程编译。



在Linux环境下使用GNU 的**make工具**能够比较容易的构建一个属于自己的工程，整个工程的编译只需要一个命令就可以完成编译、连接以至于最后的执行。

make是一个命令工具，它解释**Makefile**中的指令。在Makefile文件中描述了整个工程所有文件的编译顺序、编译规则。


```
arm$ls  
led.c  led.h  makefile
```

第一步通过交叉编译器生成目标文件

```
arm$arm-cortex_a9-linux-gnueabi-gcc -c -nostdlib led.c -o led.o  
arm$ls  
led.c  led.h  led.o  makefile
```


第二步通过链接器生成arm下可执行文件led.elf

```
arm$arm-cortex_a9-linux-gnueabi-ld -nostartfiles -nostdlib -Ttext=0x48000000 -el  
ed_test led.o -o led  
arm$ls  
led  led.c  led.h  led.o  makefile
```

第三步二进制文件拷贝处理命令将elf头信息剥离

```
arm$arm-cortex_a9-linux-gnueabi-objcopy -O binary led led.bin  
arm$ls  
led  led.bin  led.c  led.h  led.o  makefile
```

创建makefile



The image shows a gedit window titled "makefile (/opt/arm) - gedit". The window contains a Makefile with the following rules:

```
led.bin:led
    arm-cortex_a9-linux-gnueabi-objcopy -O binary led led.bin

led:led.o
    arm-cortex_a9-linux-gnueabi-ld -nostartfiles -nostdlib -Ttext=0x48000000 -eled_test led.o -o led

led.o:led.c
    arm-cortex_a9-linux-gnueabi-gcc -c -nostdlib led.c -o led.o

clean:
    rm led.o
    rm led|
```

The window has a menu bar with "打开(O)" (Open) and a search icon, and a "保存(S)" (Save) button. The text is color-coded: targets (led.bin, led, led.o, clean) are in blue, and commands (rm) are in red.

用makefile编译文件

```
arm$ls
led.c led.h makefile
arm$make
arm-cortex_a9-linux-gnueabi-gcc -c -nostdlib led.c -o led.o
arm-cortex_a9-linux-gnueabi-ld -nostartfiles -nostdlib -Ttext=0x48000000 -e led_test led.o -o led
arm-cortex_a9-linux-gnueabi-objcopy -O binary led led.bin
arm$ls
led led.bin led.c led.h led.o makefile
```

用make clean清除多余文件

```
arm$ls
led led.bin led.c led.h led.o makefile
arm$make clean
rm led.o
rm led
arm$ls
led.bin led.c led.h makefile
arm$
```

Makefile文件

```
objects=asm.elf asm.o
```

```
asm.bin:asm.elf
```

```
    arm-linux-objcopy -O binary -S $< $@
```

```
asm.elf:asm.o
```

```
    arm-linux-ld -Ttext 0x30000000 -o $@ $^
```

```
%.o:%.s
```

```
    arm-linux-gcc -c -o $@ $<
```

```
clean:
```

```
    rm -f $(objects)
```

- **\$@** 目标文件名称。
- **\$<** 第一个依赖文件的名称。
- **\$^** 所有的依赖文件，以空格分开
- **%** 通配符，和一字符串中任意个数的字符匹配。

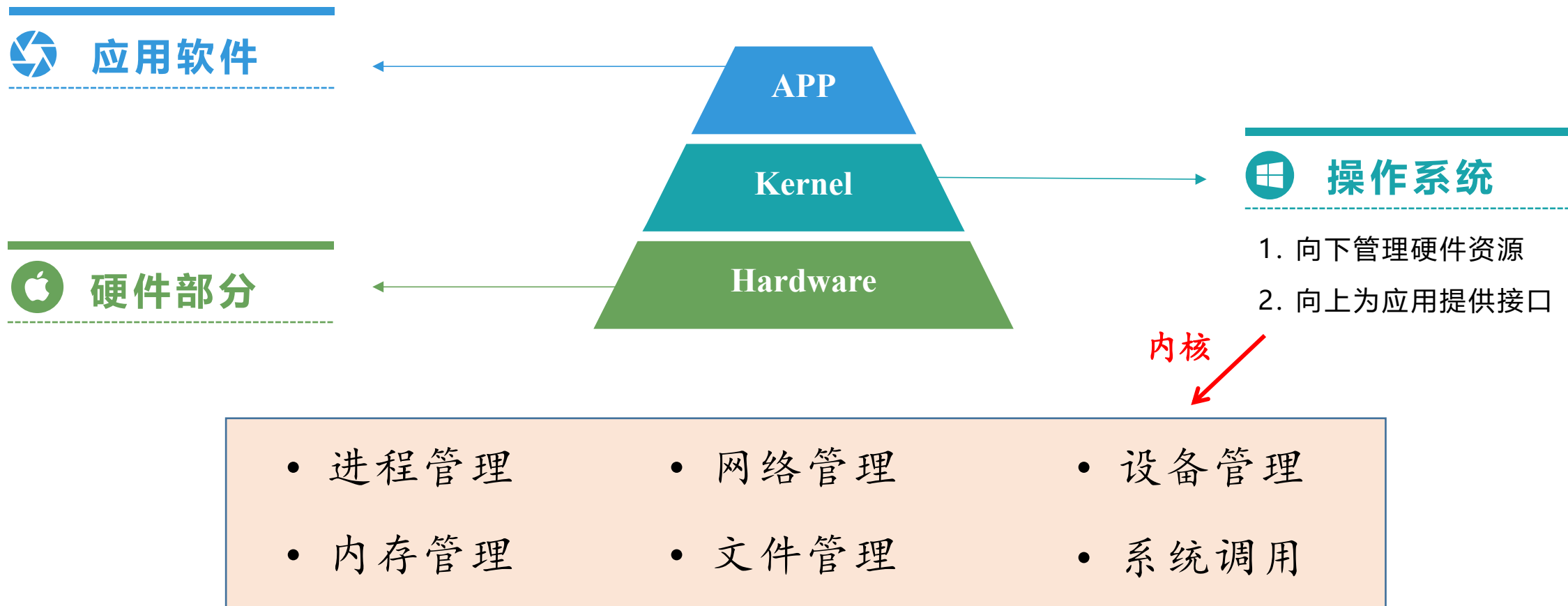
3. 第四次实验

实验指导书中第三篇的实验十一、实验十二

	嵌入式Linux	
第三篇	Linux 操作系统移植与驱动初体验.....	39
实验十一	Linux 系统移植初体验.....	39
实验十二	Linux 操作系统下驱动开发初体验.....	42

3.1 嵌入式Linux系统移植

(1) 嵌入式系统的结构（带操作系统）



(2) 什么是移植？

使操作系统与硬件匹配

将已有的软件，根据硬件平台的差异，做少量的代码修改（或裁剪），使得该软件可以在新的硬件平台上运行起来的过程。

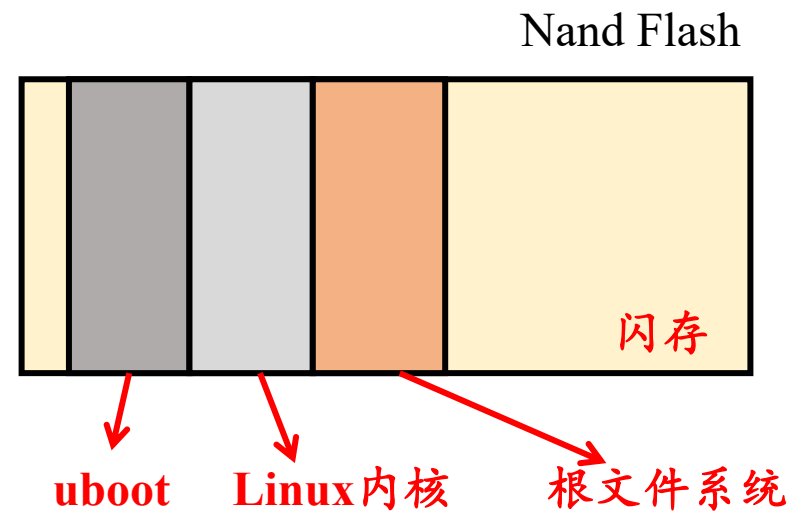
Linux是一个通用的内核（不为特定处理器或硬件板设计），所以获取Linux源码后，需要先经过相应的配置，使其与硬件平台相匹配后才能进行编译和安装。

(3) Linux移植的内容

① bootloader (uboot)

② Linux内核

③ 根文件系统 rootfs



① bootloader

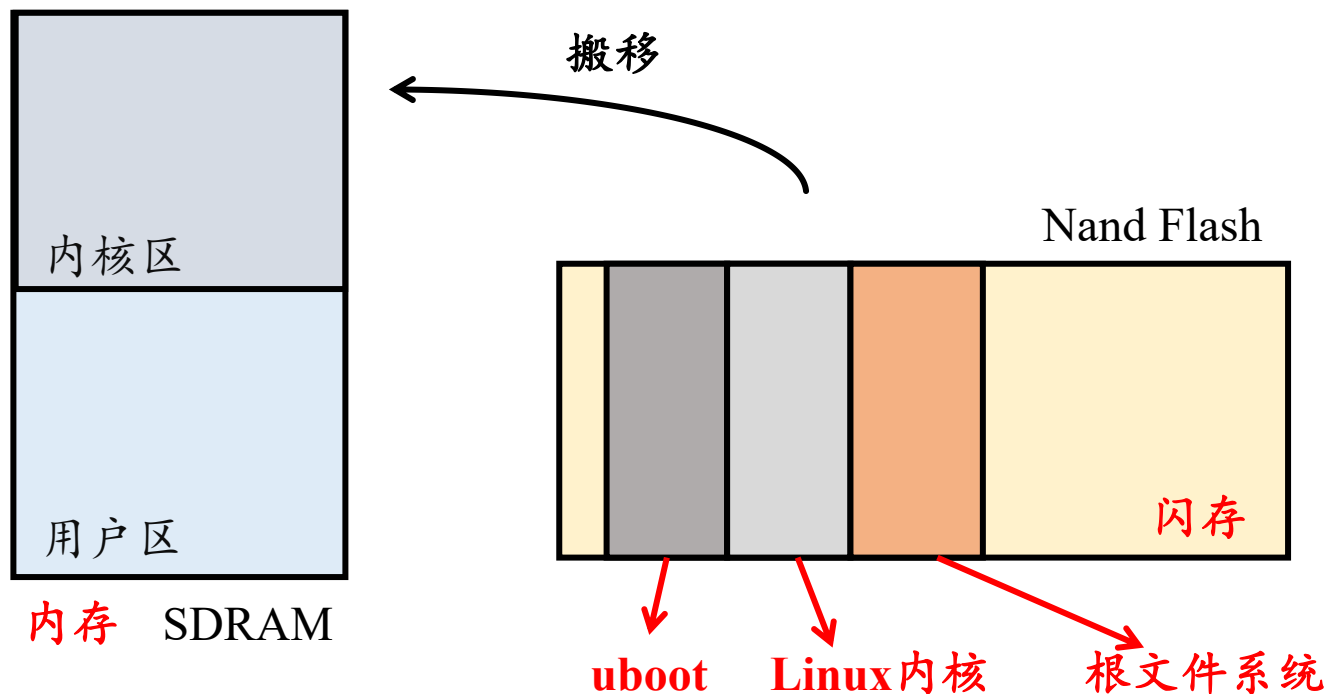
启动

加载

上电后，在操作系统运行之前运行的一小段代码，用于将软硬件环境初始化到一个合适的状态，为操作系统的加载和运行做准备。

Bootloader基本功能：

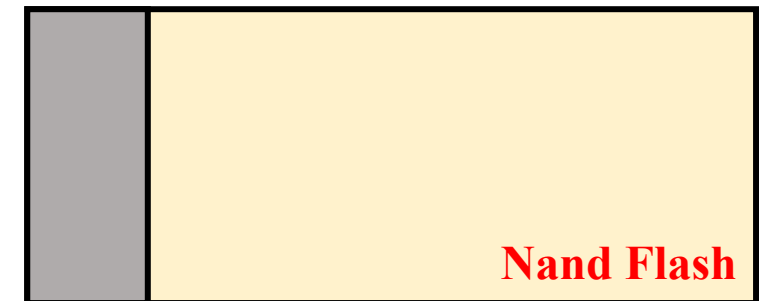
- 初始化软硬件环境；
- 引导加载Linux内核；
- 给Linux内核传参
- 执行用户命令。



注：Bootloader是启动加载程序的统称，嵌入式Linux常用的Bootloader工具是u-boot。

```
#####      Boot for Nor Flash Main Menu      #####
#####      EmbedSky USB download mode        #####
```

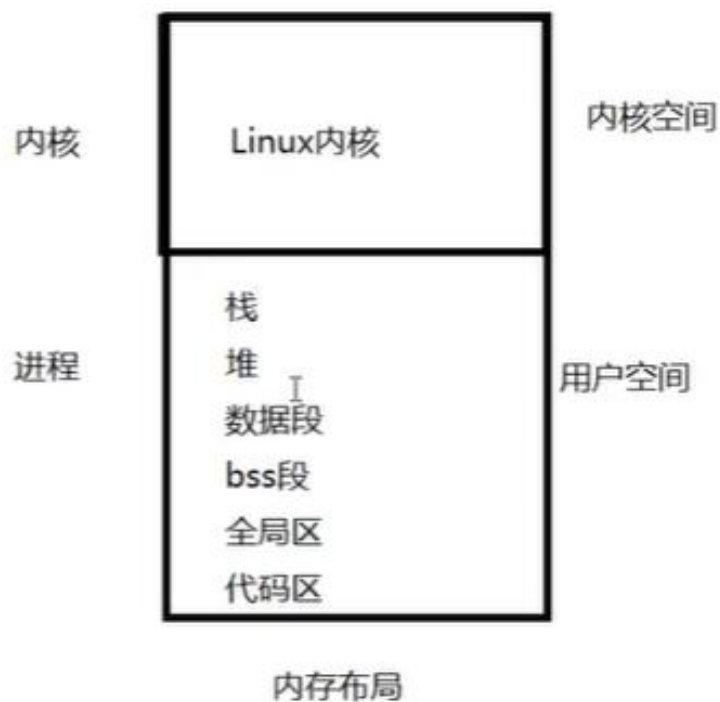
- [1] Download u-boot or STEPLDR.nb1 or other bootloader to Nand Flash
- [2] Download Eboot (eboot.nb0) to Nand Flash
- [3] Download Linux Kernel (zImage.bin) to Nand Flash
- [5] Download CRAMFS image to Nand Flash
- [6] Download YAFFS image (root.bin) to Nand Flash
- [7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it



↑
u-boot

② Linux内核

必须由bootLoader加载和启动。



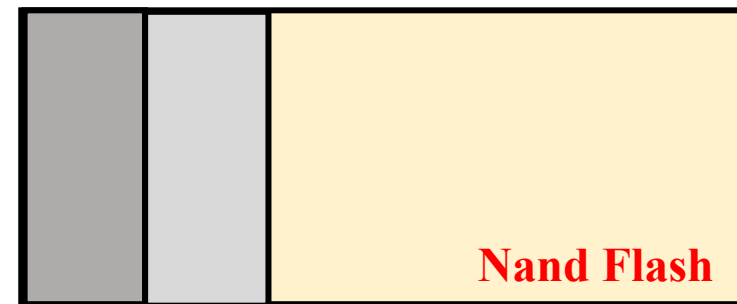
内核可以看成是一个大的裸机程序，自上而下运行，最后进入while(1)循环；

内核加载后，会根据bootloader传递的参数去闪存中寻找根文件系统；

将内核控制权交给根文件系统。

```
#####      Boot for Nor Flash Main Menu      #####
#####      EmbedSky USB download mode        #####

[1] Download u-boot or STEPLDR.nb1 or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Get the boot parameters
```



Linux内核

③ 根文件系统（rootfs）

根目录下所有文件构成“根文件系统”，常见目录：

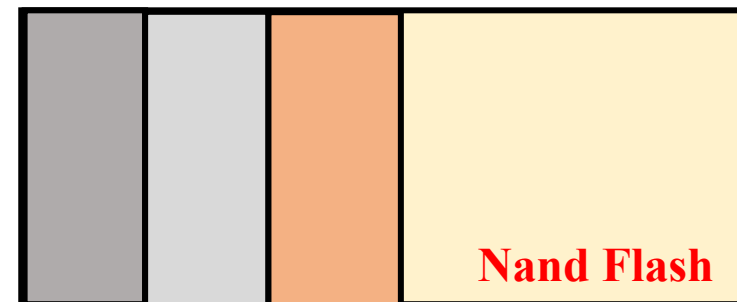
- bin目录：保存普通用户命令；
- sbin目录：保存超级用户命令；
- lib目录：保存库文件；
- etc目录：保存配置文件；

```
[root@192 ~]# cd /  
[root@192 /]# ls  
bin    dev    home  lost+found  mnt  proc  sbin    srv  tmp  var  
boot  etc    lib   media      opt  root  selinux sys  usr  
[root@192 /]#
```

当内核根据参数找到根文件系统后，运行/sbin/init第一号进程。

```
#####      Boot for Nor Flash Main Menu      #####
#####      EmbedSky USB download mode        #####

[1] Download u-boot or STEPLDR.nb1 or other bootloader to Nand Flash
[2] Download Eboot (eboot.nb0) to Nand Flash
[3] Download Linux Kernel (zImage.bin) to Nand Flash
[5] Download CRAMFS image to Nand Flash
[6] Download YAFFS image (root.bin) to Nand Flash
[7] Download Program (uCOS-II or TQ2440_Test) to SDRAM and Run it
[8] Boot the system
[9] Format the Nand Flash
[0] Set the boot parameters
```

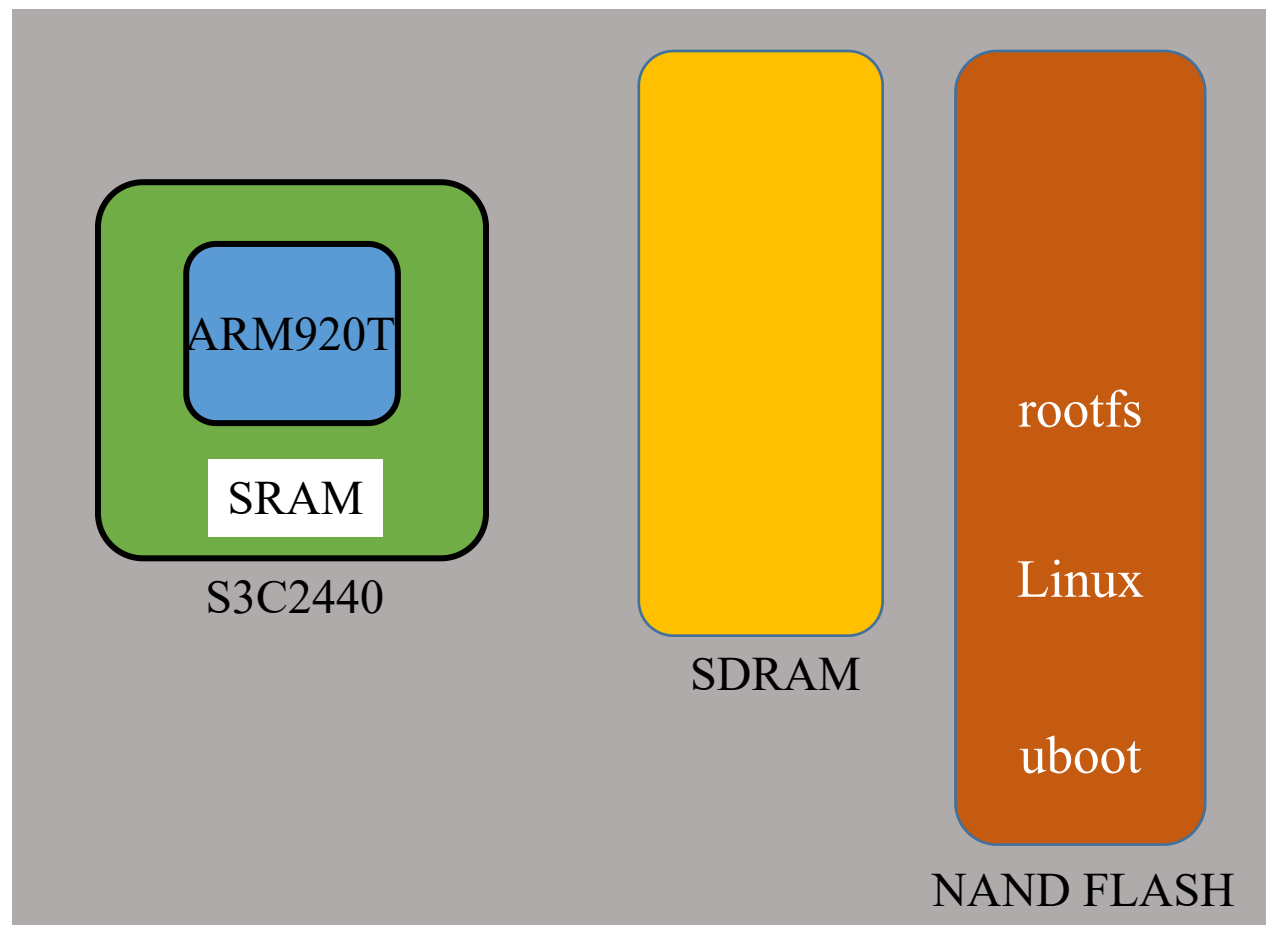


↑
根文件系统

(4) 嵌入式Linux系统启动过程

开发板上电后：

- 运行uboot→硬件初始化→加载内核→向内核传递参数→启动内核；
- 内核启动→uboot生命结束→七大功能初始化→根据参数寻找rootfs；
- 找到挂载rootfs→运行/sbin下init一号进程→运行/bin/sh程序启动shell→等待输入命令。



2.2 嵌入式Linux驱动开发

(1) Linux驱动开发与裸机开发区别

- 裸机开发回顾

- 底层，跟寄存器打交道；

- Linux驱动开发思维

- Linux下不直接操作寄存器；
- 根据Linux下的各种驱动框架进行开发；（掌握Linux下各种驱动框架）
- 设备驱动以内核模块（module）的形式出现（好处：动态加载）；
- 硬件驱动最终表现为设备文件/dev/xxx文件；（打开，关闭，读写，创建.....）

```
tq2440-leds initialized  
[root@Embedsky /udisk]# ls /dev/tq2440-leds  
/dev/tq2440-leds
```


(2) Linux设备驱动程序特点

- Linux下对外设的访问只能通过驱动程序；
- Linux对驱动程序有统一的**框架**，以**文件的形式**定义系统的驱动程序；
- 驱动程序是**内核的一部分**，可以使用中断、DMA操作；
- Linux设备驱动可以以**内核模块(module)**的形式**加载**和**卸载**。

Linux内核模块编程

Linux 内核模块编程

包含模块头文件 ←

```
NAME: EmbedSky_hello.c
COPYRIGHT: www.embedsky.net

#include <linux/module.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/init.h>
#include <linux/miscdevice.h>
#include <linux/delay.h>
#include <asm/irq.h>
#include <asm/arch/regs-gpio.h>
#include <asm/hardware.h>
```

需使用内核
专用函数 ←

```
MODULE_LICENSE("GPL");

static int __init EmbedSky_hello_init(void)
{
    printk(<1>\n    Hello, EmbedSky!\n");
    printk(<1>\nThis is first driver program.\n\n");

    return 0;
}

static void __exit EmbedSky_hello_exit(void)
{
    printk("<1>\n    Exit!\n");
    printk("<1>\nGoodbye EmbedSky!\n\n");
}
```

将函数封装成模块 ←

```
module_init(EmbedSky_hello_init);
module_exit(EmbedSky_hello_exit);
```

Linux内核模块编译及运行

编译后可生成相应的模块文件 EmbedSky_hello.ko

加载模块：

```
insmod EmbedSky_hello.ko
```

停止卸载模块：

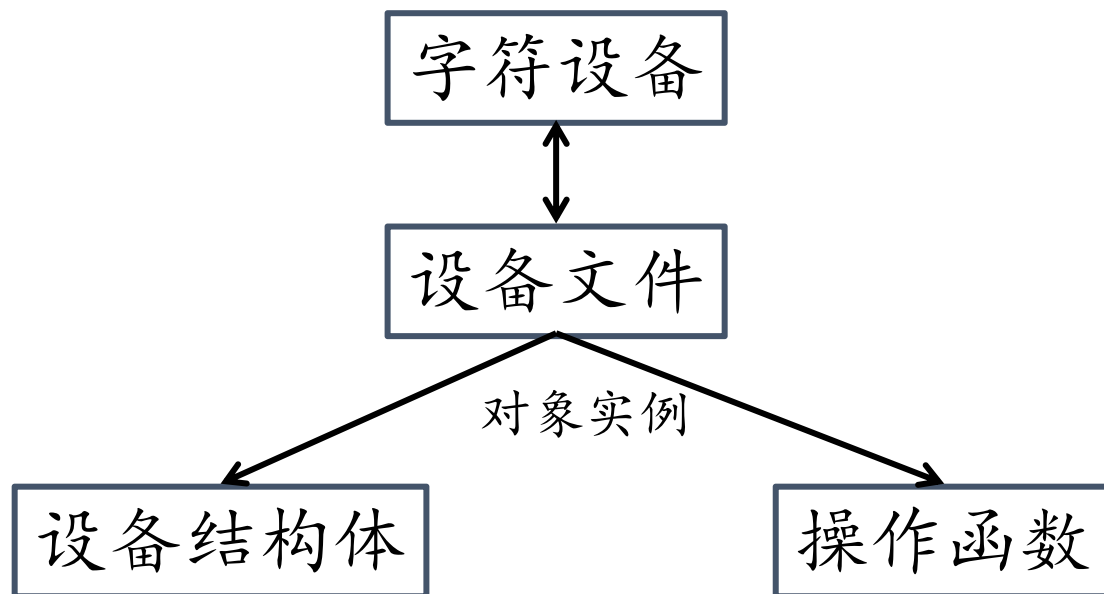
```
rmmod EmbedSky_hello
```

(3) Linux设备分类

- **字符设备** 鼠标、键盘、显示器
 - 以字节为单位逐个进行I/O操作；
 - 字符设备中缓存可有可无；
 - 不支持随机访问；
- **块设备** 硬盘、U盘
 - 块设备的存取时通过buffer、cache进行；
 - 可以随机访问；
 - 可以支持安装文件系统；
- **网络设备** 蓝牙、Wifi、网卡
 - 通过BSD套接口访问；

led是什么设备？

(4) 字符设备驱动程序



继承

```
struct hello_char_dev{  
    struct cdev cdev;  
    char c;  
};
```

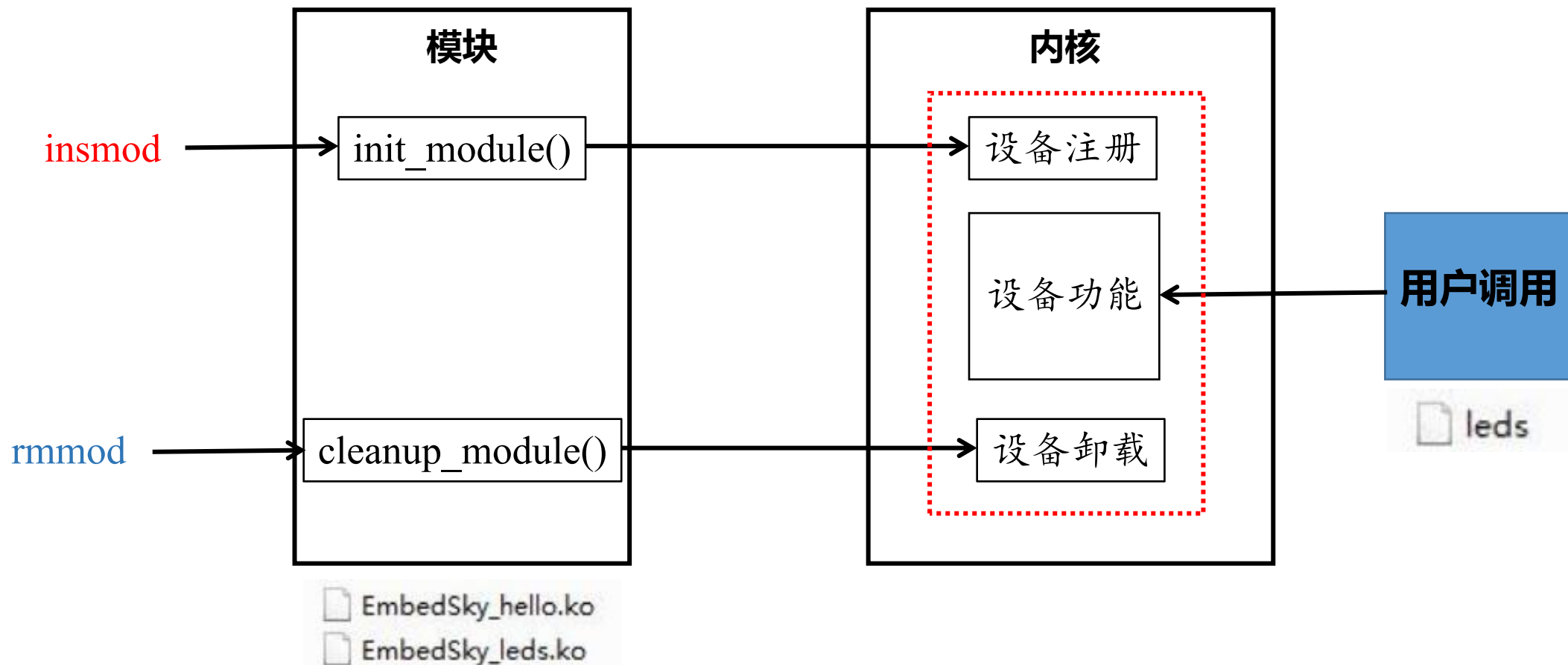
属性

```
struct file_operation hc_fops = {  
    .owner = THIS_MODULE,  
    .read = hc_read,  
    .write = hc_write,  
    .open = hc_open,  
    .release = hc_release,  
};
```

方法

(5) 嵌入式Linux驱动程序代码结构

- 设备驱动单独作为一个模块编译，在需要时**动态加载**到内核中，不需要时可以从内核中**卸载**。



4. 实验报告及验收标准

4.1 实验报告

- ① 结合实验五，说说你所理解的Linux操作系统；
- ② 列出你在实验六中执行的一些常规命令行，给出该行命令的功能解释，并附上命令行执行截图；
- ③ 列出实验六、实验八的实验过程，编写并编译自己的第一个c程序（硬件无关），并附实验截图；
- ④ 列出实验九、十的实验过程，并附实验截图；
- ⑤ 思考题
- ⑥ 体会和建议

4.2 思考题

- ① 思考windows环境下与Linux环境下开发裸机程序的区别有什么？
- ② make及makefile的作用是什么？
- ③ 第二篇实验与第三篇实验的Linux系统一样吗？如不同，不同之处有什么？
- ④ 关于Linux操作系统，你还想要了解什么？
- ⑤ 关于嵌入式Linux操作系统的开发，你还希望学习什么？

4.3 验收标准

	常用命令	vi&gedit编辑器	gcc编译	makefile	交叉编译	现象及代码分析
第四次验收	20	20	10	20	10	20

	系统移植初体验	驱动开发初体验	应用开发初体验
第五次验收	50	30	20

实验规范：

- 1、实验板带电运行时，尽量勿用手触碰实验板背面，以免导致短路；
- 3、实验过程中，爱护实验板，轻拿轻放；
- 4、实验结束后，将开发板按照原样收拾好包装，放回原处；
- 5、离开前，请及时关闭电脑，清理好桌面，放好板凳；
- 6、离开实验室时，请将产生的垃圾带出实验室。