嵌入式操作系统软件开发

题型

1。选择 30

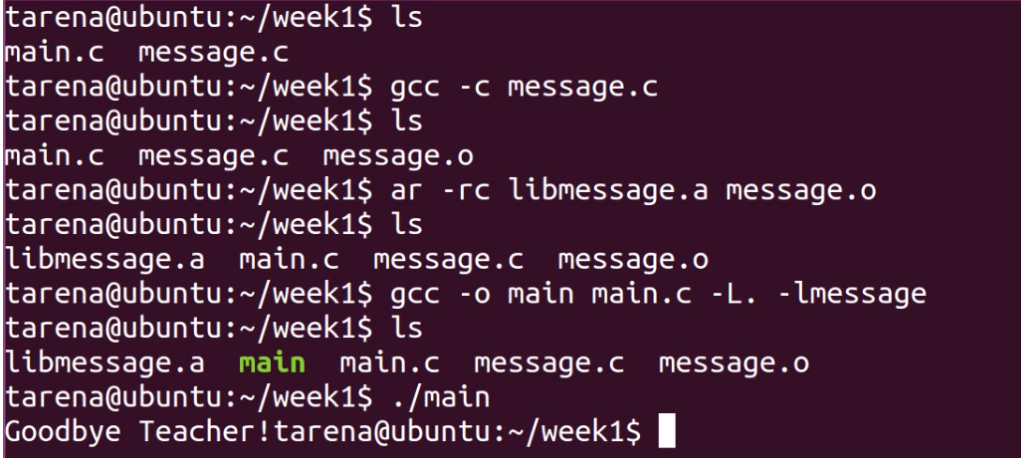
2。简答 30

3。程序 40

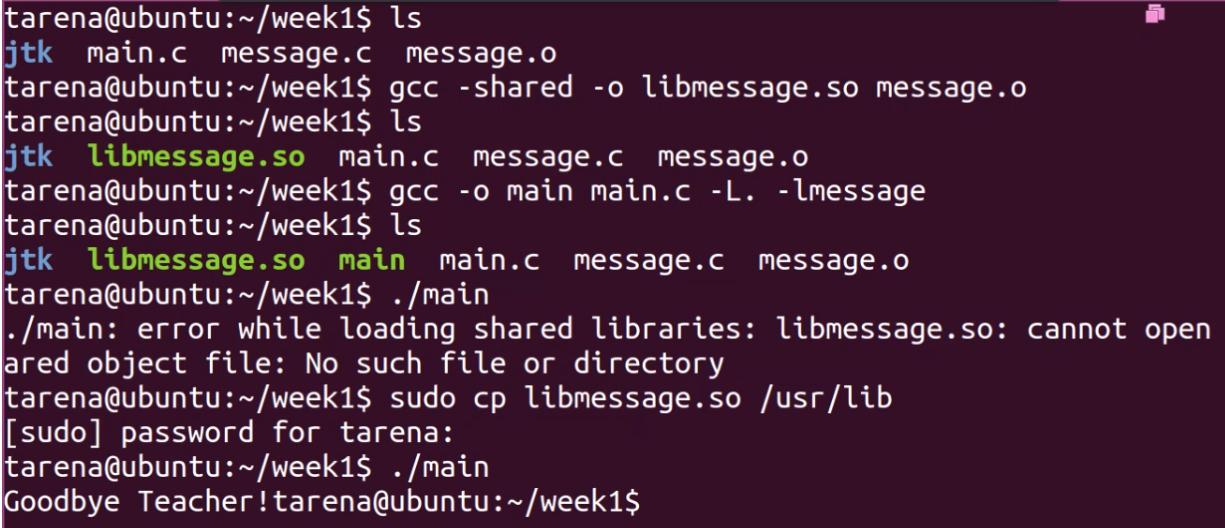
主要内容

1. 多文件编译
2. 静态库和动态库（共享库）文件制作：概念，编译过程，区别以及各自的特点

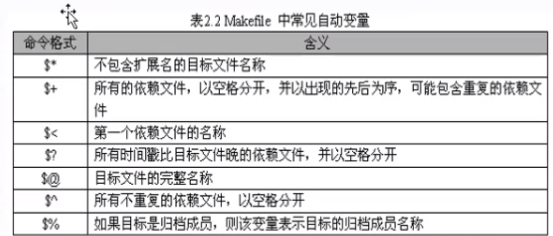
静态库：



动态库：要将动态库文件转移到/usr/lib目录下才可以运行



1. Makefile文件制作：基本格式，变量的定义和使用，隐式规则



1. 线程函数：线程的创建，线程的阻塞，线程号查询，线程的结束cancle（在别的线程中取消）和exit（在本线程中取消）和取消

[pthread\_create](http://baike.baidu.com/item/pthread_create" \t "http://baike.baidu.com/item/Pthread/_blank)()：创建一个线程

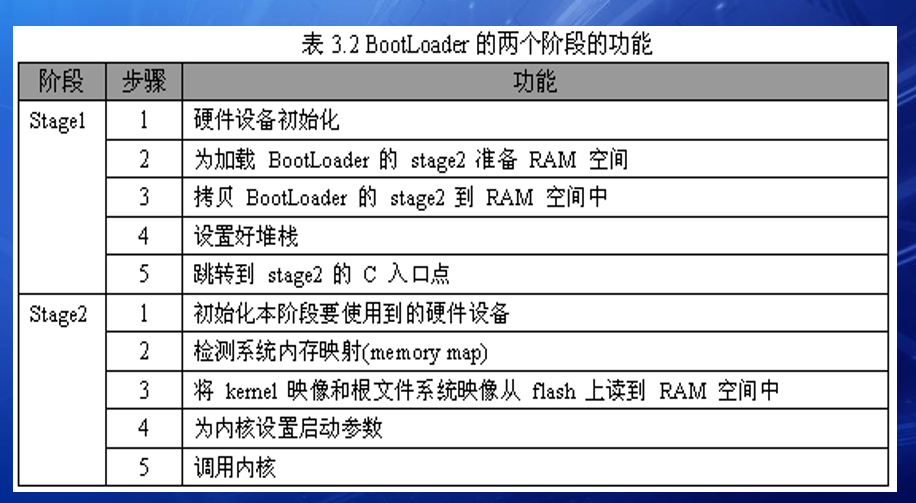
[pthread\_join](http://baike.baidu.com/item/pthread_join" \t "http://baike.baidu.com/item/Pthread/_blank)()：阻塞当前的线程，直到另外一个线程运行结束

[pthread\_self](http://baike.baidu.com/item/pthread_self" \t "http://baike.baidu.com/item/Pthread/_blank)(): 查询线程自身线程标识号

[pthread\_exit](http://baike.baidu.com/item/pthread_exit" \t "http://baike.baidu.com/item/Pthread/_blank)()：终止当前线程

pthread\_cancel()：中断另外一个线程的运行

1. 互斥锁，信号量，条件变量
   1. BootLoaber:两个阶段的任务



* 1. Kernel：Kconfig，make Menuconfig给Kconfig做菜单，Makefile文件的修改（arch=arm；cross-compile:arm-linux-gcc（交叉编译器）），

嵌入式Linux系统编译过程

在linux-2.6.35.7/drives/char下编写helloworld.c字符设备驱动文件，helloworld.c代码如下:

**#include <linux/init.h>**

**#include <linux/module.h>**

**static int \_\_init helloworld\_init(void)**

**{**

**printk(KERN\_ALERT "My first kernel module !\n");**

**return 0;**

**}**

**static void \_\_exit helloworld\_exit(void)**

**{**

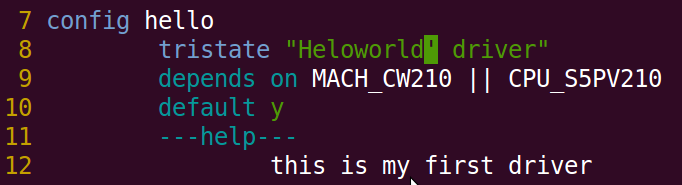
**printk(KERN\_ALERT "Bye, my module !\n");**

**}**

**module\_init(helloworld\_init);**

**module\_exit(helloworld\_exit);**

1. 修改linux-2.6.35.7/drivers/char下的Kconfig文件，增加bool型的helloworld选项。



1. 修改linux-2.6.35.7/drivers/char下的Makefile文件，增加obj-$(CONFIG\_HELLO) += helloworld.o行内容。
2. cp arch/arm/configs/cw210\_defconfig .config
3. 修改linux-2.6.35.7下的Makefile文件，增加以下内容：

191 ARCH ?=arm

192 CROSS\_COMPILE ?=arm-linux-

5、在linux-2.6.35.7下配置menuconfig，利用Ctrl+-键缩小屏幕字体，输入命令make menuconfig，在Device Drivers下的Character devices选项下选中helloworld字符设备。

6、在linux-2.6.35.7下输入make zImage命令，编译完成后会在linux-2.6.35.7/arch/arm/boot文件夹中生成系统压缩镜像文件zImage。

* 1. 根文件系统：Busybox

**linux根文件系统的制作**

**1.创建根文件目录**

**mkdir rootfs**

**2.创建子目录**

**cd rootfs**

**mkdir bin dev etc lib proc sbin sys usr mnt tmp var**

**mkdir usr/bin usr/lib usr/sbin lib/modules**

**3.创建设备文件**

**该目录存放的是设备文件，其创建方法主要有3种：手动创建、使用devfs文件系统、使用udev。系统启动涉及到的设备有：/dev/mtdblock\*（MTD块设备）、/dev/ttySAC\*(串口设备)、dev/console、/dev/null，只要手动创建这些设备系统就能启动。其他设备可以在启动系统后，查看/proc/devices，然后一一创建对应的设备文件。**

**cd dev**

**[root@localhost etc]# mknod console c 5 1**

**[root@localhost etc]# mknod null c 1 3**

**[root@localhost etc]# mknod /ttySAC0 c 4 64**

**[root@localhost etc]# mknod mtdblock0 b 31 0**

**[root@localhost etc]# mknod mtdblock1 c 31 1**

**[root@localhost etc]# mknod mtdblock2 c 31 2**

**4.配置busybox**

**busybox源代码在www.busybox.net下载**

**解压busybox压缩包，例如：tar -xvzf busybox-1.15.2**

**然后进入源码包，修改Makefile**

**第164行，CROSS\_COMPILE=arm-linux-**

**第190行，ARCH=arm**

**然后对busybox进行配置make menuconfig,进入菜单后有几项需要修改**

**（1）busybox settings-->build options-->build busybox as static binary(no shared libs)选上**

**（2）busybox settings-->installation options-->don't use /usr 选上**

**（3）busybox settings-->installation options-->busybox installation prefix(编好之后的安装路径../rootfs)**

**(4)busybox settings-->busybox library tuning-->username conpletion**

**(5)busybox settings-->busybox library tuning-->fancy shell prompts**

**如果不选择上面4 ，5两项，文件系统是不识别PS1这个环境变量的参数的。那么运行linux时候命令行就只显示[\u@\h \W] 不进行动态链接。**

**然后对busybox进行编译make**

**安装make install（安装其实就是进行拷贝把编译好的命令拷贝到（3）指定的路径）**

**5 进入etc添加文件**

**cd etc**

**拷贝busybox-1.15.2/examples/bootfloopy/etc/\* 到当前目录下**

**cp -r ../../busybox-1.15.2/examples/bootfloopy/etc/\*  ./**

**包括文件：fstab init.d inittab profile**

**拷贝 /etc/passwd  /etc/group /etc/shadow到当前目录下**

**cp /etc/passwd ./**

**cp /etc/group ./**

**cp /etc/shadow ./**

**修改iniitab文件：**

**# /etc/inittab**

**::sysinit:/etc/init.d/rcS**

**console::askfirst:-/bin/sh**

**//::once:/usr/sbin/telnetd -l /bin/login**

**::ctrlaltdel:/sbin/reboot**

**::shutdown:/bin/umount -a -r**

**修改fstab文件**

**proc /proc proc defaults 0 0**

**tmpfs /tmp tmpfs defaults 0 0**

**sysfs /sys sysfs defaults 0 0**

**tmpfs /dev tmpfs defaults 0 0**

**var /dev tmpfs defaults 0 0**

**修改init.d/rcS文件**

**#!/bin/sh**

**PATH=/bin:/sbin:/usr/bin:/usr/sbin**

**runlevel=S**

**prevlevel=N**

**export PATH runlevel prevlevel**

**mount -a**

**mkdir /dev/pts**

**mount -t devpts devpts /dev/pts**

**echo /sbin/mdev > /proc/sys/kernel/hotplug**

**mdev -s**

**mkdir -p /var/lock**

**/bin/hostname -F /etc/sysconfig/HOSTNAME**

**修改profile文件**

**#!/bin/sh**

**exportHOSTNAME=imau**

**export USER=root**

**export HOME=root**

**#exportPS1="[$USER@$HOSTNAME \W]#"**

**PATH=/bin:/sbin:/usr/bin:/usr/sbin**

**LD\_LIBRARY\_PATH=lib:/usr/lib:$LD\_LIBRARY\_PATH**

**export PATHLD\_LIBRARY\_PATH**

**在etc中建sysconfig文件夹在sysconfig文件夹中建HOSTNAME文件在里面写上用户名，此名字将在linux系统的命令行中显示**

**添加库**

**xxx@Ubuntu:~/win/busybox-1.17.3/\_install/lib$cp /~/toolchain/arm-none-linux-gnueabi/lib/\*so\* ./lib/ -a**

**6.编译内核模块**

**进入内核模块的目录（例如：linux 2.6.29）**

**make modules ARCH =arm CROSS\_COMPILE=arm-linux**

**7 内核模块的安装**

**make modules\_install ARCH=arm INSTALL\_MOD\_PATH=/.../roofs(根文件系统所在的目录)**

**这样就将我们内核编译好的内核模块拷贝到了跟文件系统中**

**8.根据想要做的文件系统的类型，使用不同的工具对其进行编译，然后拷进开发板**

**以做yaffs2为例：mkyaffs2image rootfs(文件系统名字) rootfs.bin(制作的yaffs2文件系统的名字)**

1. 驱动开发：字符设备驱动：特点；块设备；网络设备；杂项设备；

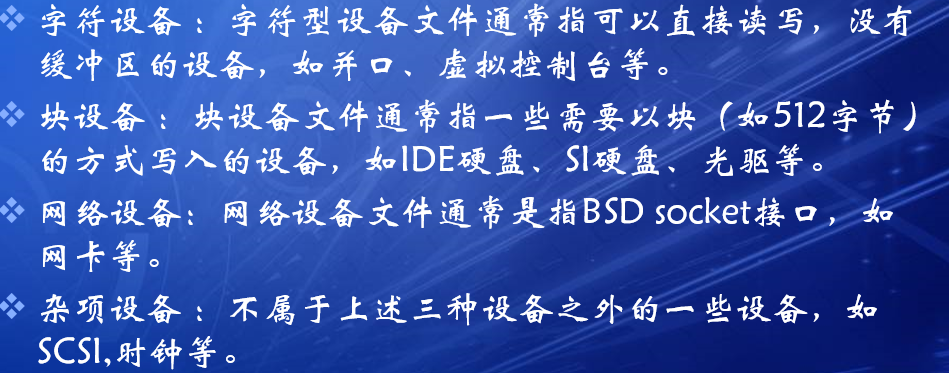
字符设备与块设备的区别，特点

(1)字符设备：提供连续的数据流，应用程序可以顺序读取，通常不支持随机存取。相反，此类设备支持按字节/字符来读写数据。举例来说，键盘、串口、调制解调器都是典型的字符设备。

(2)块设备：应用程序可以随机访问设备数据，程序可自行确定读取数据的位置。硬盘、软盘、CD-ROM驱动器和闪存都是典型的块设备，应用程序可以寻址磁盘上的任何位置，并由此读取数据。此外，数据的读写只能以块(通常是512B)的倍数进行。与字符设备不同，块设备并不支持基于字符的寻址。

总结一下，这两种类型的设备的根本区别在于它们是否可以被随机访问。字符设备只能顺序读取，块设备可以随机读取。

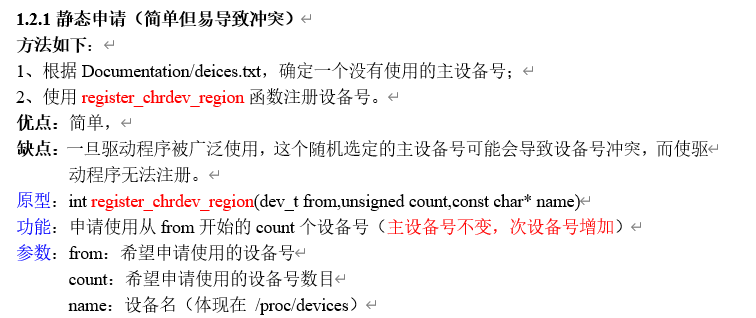
块和字符设备的区别：仅仅是驱动的接口函数与内核的接口函数不同。

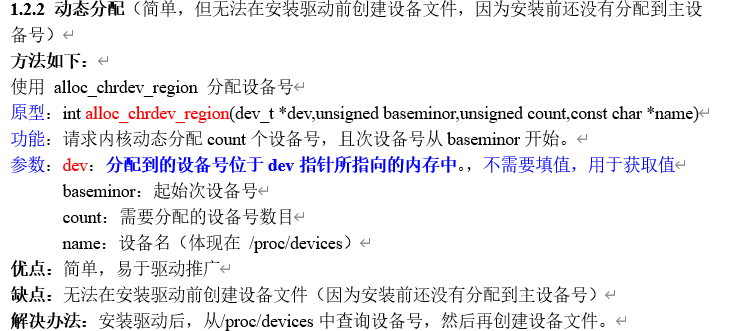


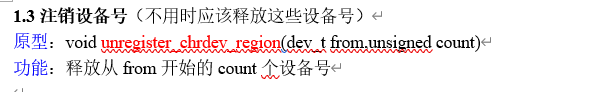
1. 驱动编写：框架，主要函数，主要数据结构，驱动程序加载过程（编译、插入内核、查看主设备号、创建设备节点），应用态（用户）程序编写（读写）

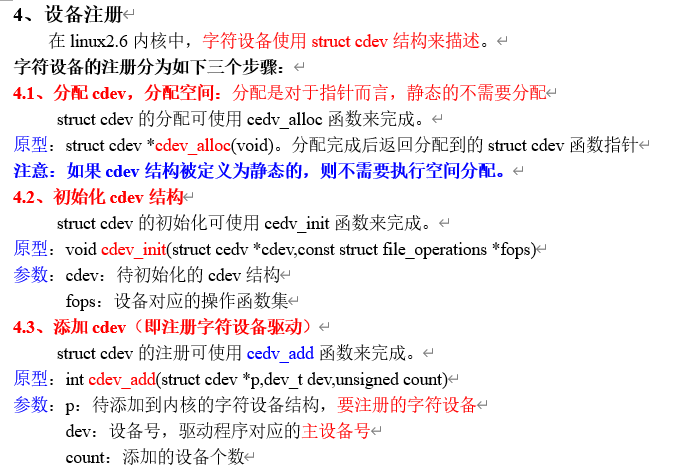
主要函数：

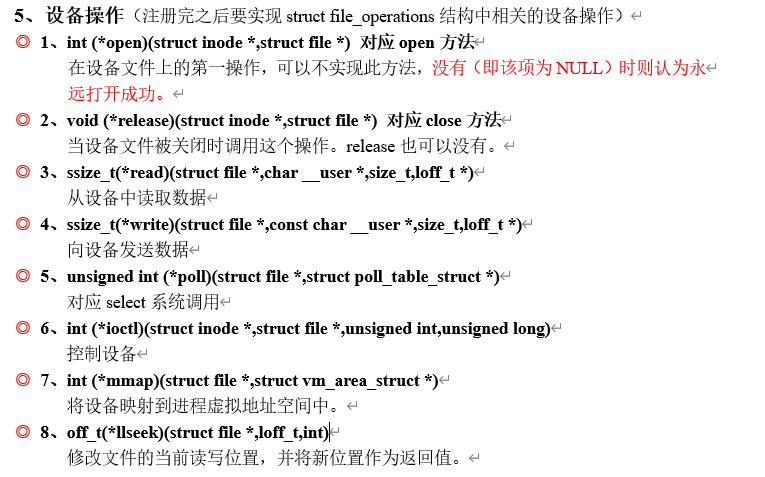
分配主设备号：











主要数据结构：



嵌入式linux驱动编译过程

1. **编写驱动源文件my\_char.c**
2. **编写Makefile文件（借用成型模板）**
3. **编译驱动程序，获得my\_char.ko文件**
4. **往内核中加载驱动程序：sudo insmod my\_char.ko**
5. **查看驱动是否加载成功:lsmod**
6. **在/dev下创建设备节点：sudo mknod /dev/my\_char c 250 0，创建以前首先查看设备主设备号：cat /proc/devices (注意/proc/devices下显示的设备名称是驱动程序register\_chrdev函数里注册的设备名,而/dev下注册的设备名是应用程序需打开的设备名)**
7. **编写应用测试程序：test\_char.c**
8. **编译test\_char.c:gcc –o test test\_char.c**
9. **运行可执行文件:sudo ./test测试驱动程序是否成功**
10. **卸载驱动：sudo rmmod my\_char.ko**
11. **查看是否卸载成功：lsmod或cat /proc/devices**
12. **查看卸载信息：dmesg**
13. 嵌入式数据库：sqlite3