

实验 08_生成设备节点

8.1 本章导读

一部分驱动要和上层通信，都需要生成设备节点，上层应用通过一套标准的接口函数调用设备节点就可以控制底层以及和底层通信。

本章就给大家介绍最简单易用的杂项设备节点如何生成。

8.1.1 工具

8.1.1.1 硬件工具

- 1) iTOP4412 开发板
- 2) U 盘或者 TF 卡
- 3) PC 机
- 4) 串口

8.1.1.2 软件工具

- 1) 虚拟机 Vmware
- 2) Ubuntu12.04.2
- 3) 超级终端 (串口助手)
- 4) 实验配套源码文件夹 “devicenode_linux_module”

8.1.2 预备课程

实验 06_设备注册

实验 07_驱动注册

8.1.3 视频资源

本节配套视频为“视频 08_生成设备节点”

8.2 学习目标

本章需要学习以下内容：

理解什么是杂项设备

生成杂项设备的设备节点

8.3 为什么引入杂项设备

在虚拟机的 Ubuntu 系统上，如下图所示，使用命令“cat /proc/misc”，可以查看到 PC 机 Ubuntu 系统的杂项设备。

```
root@ubuntu: ~  
root@ubuntu:~# cat /proc/misc  
53 vsock  
184 microcode  
54 vmci  
55 network_throughput  
56 network_latency  
57 cpu_dma_latency  
58 alarm  
59 ashmem  
60 binder  
236 device-mapper  
223 uinput  
1 psaux  
200 tun  
237 loop-control  
175 agpgart  
228 hpet  
229 fuse  
61 ecryptfs  
231 snapshot  
227 mcelog  
62 rfkill  
63 vga_arbiter  
root@ubuntu:~#
```

启动开发板，在超级终端中输入命令“cat /proc/misc”也可以查看对应的杂项设备。

```
[root@iTOP-4412]# cat /proc/misc
47 network_throughput
48 network_latency
49 cpu_dma_latency
50 xt_qtaguid
251 srp_ctrl
253 srp
236 device-mapper
130 watchdog
51 alarm
223 uinput
52 keychord
53 usb_accessory
54 mtp_usb
55 android_adb
1 pmem_gpul
0 pmem
56 relay_ctl
57 adc
58 buzzer_ctl
59 leds
60 max485_ctl_pin
61 AGPS
229 fuse
62 ashmem
63 ion
[root@iTOP-4412]#
```

前面介绍过主设备号只有 256 个，设备又非常多，所以引入了子设备号。

其中杂项设备的主设备号是 10，在任何 Linux 系统中它都是固定的。

一般将 Linux 驱动分为字符设备、块设备、网络设备，但是这个分类不能包含所有的设备，所以将无法归类的设备统称为杂项设备，杂项设备可能用到字符设备、块设备、网络设备中的一项或者多项设备。

如下图所示，进入源码文件夹，使用命令 “ls drivers/char/” ，可以查看到杂项设备的文件 “misc.c” 。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:~# cd /home/topeet/android4.0/iTop4412_Kernel_3.0
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0# ls drivers/char/
agp                hangcheck-timer.c  mem.o              raw.c
apm-emulation.c    hpet.c             misc.o             rtc.c
applicom.c          hw_random           misc.o             s3c_mem.c
applicom.h          i8k.c              mmtimer.c          s3c_mem.h
bfin-otp.c          ipmi                modules.builtin    s3c_mem.o
briq_panel.c        itop4412_adc.c      modules.order       scc.h
bsr.c               itop4412_adc.o      msm_smd_pkt.c       scx200_gpio.c
built-in.o          itop4412_buzzer.c   mspec.c             snsc.c
dcc_tty.c           itop4412_buzzer.o   mwave              snsc_event.c
ds1302.c            itop4412_leds.c     nsc_gpio.c          snsc.h
ds1620.c            itop4412_leds.o     nvram.c             sonypi.c
dsp56k.c            itop4412_relay.c    nwbutton.c          tb0219.c
dtlk.c              itop4412_relay.o    nwbutton.h          tlclk.c
efirtc.c            Kconfig             nwflash.c           toshiba.c
exynos_mem.c        lp.c                pc8736x_gpio.c      tpm
exynos_mem.o        Makefile             pcmcia              ttyprintk.c
generic_nvram.c     max485_ctl.c         ppdev.c             uv_mmtimer.c
genrtc.c            max485_ctl.o         ps3flash.c          viotape.c
gps.c               mbc.c               ramoops.c           virtio_console.c
gps.h               mbc.h               random.c             xilinx_hwicap
gps.o               mem.c               random.o
root@ubuntu:/home/topeet/android4.0/iTop4412_Kernel_3.0#
```

如上图所示，可以看到它被编译为“misc.o”，也就是被编译进了内核 zImage 文件。

使用命令“vim drivers/char/Makefile”打开杂项设备文件的编译文件。如下图所示，可以看到，这个文件相当于被强制编译的。

```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#
# Makefile for the kernel character device drivers.
#
obj-y               += mem.o random.o
obj-$(CONFIG_TTY_PRINTK) += ttyprintk.o
obj-y               += misc.o
obj-$(CONFIG_ATARI_DSP56K) += dsp56k.o
obj-$(CONFIG_VIRTIO_CONSOLE) += virtio_console.o
obj-$(CONFIG_RAW_DRIVER) += raw.o
obj-$(CONFIG_SGI_SNSC) += snsc.o snsc_event.o
obj-$(CONFIG_MSM_SMD_PKT) += msm_smd_pkt.o
obj-$(CONFIG_MSPEC) += mspec.o
obj-$(CONFIG_MMTIMER) += mmtimer.o
obj-$(CONFIG_UV_MMTIMER) += uv_mmtimer.o
obj-$(CONFIG_VIOTAPE) += viotape.o
obj-$(CONFIG_IBM_BSR) += bsr.o
obj-$(CONFIG_SGI_MBCS) += mbc.o
obj-$(CONFIG_BRIQ_PANEL) += briq_panel.o
obj-$(CONFIG_BFIN_OTP) += bfin-otp.o
obj-$(CONFIG_PRINTER) += lp.o

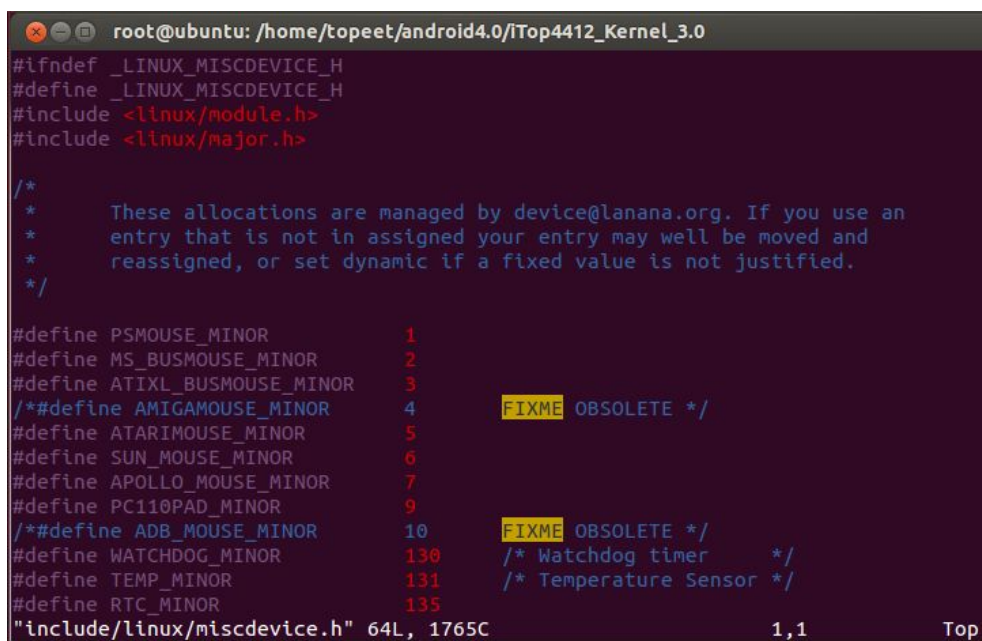
"drivers/char/Makefile" 77L, 2287C                               1,1                               Top
```

杂项设备设备部分完全制作好了，只需要添加子设备，非常方便，在后面实验操作中大家就可以感受到。

这样杂项设备的引入即解决了设备号数量少的问题，又降低了使用难度，还能防止碎片化，一举多得。

8.4 杂项设备注册函数以及结构体

杂项设备的头文件在“include/linux/miscdevice.h”，有两个需要掌握的函数和一个结构体，如下图所示，在源码目录下使用命令“vim include/linux/miscdevice.h”。



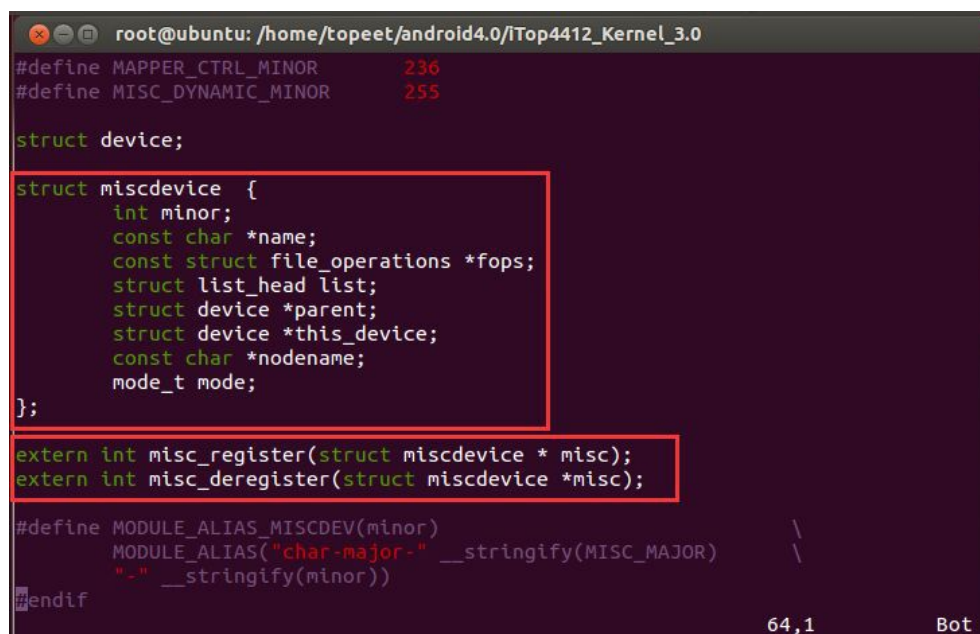
```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#ifndef _LINUX_MISCDEVICE_H
#define _LINUX_MISCDEVICE_H
#include <linux/module.h>
#include <linux/major.h>

/*
 * These allocations are managed by device@lanana.org. If you use an
 * entry that is not in assigned your entry may well be moved and
 * reassigned, or set dynamic if a fixed value is not justified.
 */

#define PSMOUSE_MINOR          1
#define MS_BUSMOUSE_MINOR     2
#define ATIXL_BUSMOUSE_MINOR  3
/*#define AMIGAMOUSE_MINOR    4      FIXME OBSOLETE */
#define ATARIMOUSE_MINOR      5
#define SUN_MOUSE_MINOR       6
#define APOLLO_MOUSE_MINOR    7
#define PC110PAD_MINOR        9
/*#define ADB_MOUSE_MINOR    10     FIXME OBSOLETE */
#define WATCHDOG_MINOR       130    /* Watchdog timer */
#define TEMP_MINOR            131    /* Temperature Sensor */
#define RTC_MINOR             135

#include/linux/miscdevice.h" 64L, 1765C 1,1 Top
```

如下图所示，到最底行。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#define MAPPER_CTRL_MINOR      236
#define MISC_DYNAMIC_MINOR    255

struct device;

struct miscdevice {
    int minor;
    const char *name;
    const struct file_operations *fops;
    struct list_head list;
    struct device *parent;
    struct device *this_device;
    const char *nodename;
    mode_t mode;
};

extern int misc_register(struct miscdevice * misc);
extern int misc_deregister(struct miscdevice *misc);

#define MODULE_ALIAS_MISCDEV(minor) \
    MODULE_ALIAS("char-major-" __stringify(MISC_MAJOR) \
    "-" __stringify(minor))
#endif
```

如上图所示，红色框中有两个函数。

`extern int misc_register(struct miscdevice * misc);`

杂项设备注册函数；一般在 probe 中调用，参数是 miscdevice

`extern int misc_deregister(struct miscdevice *misc);`

杂项设备卸载函数；一般是在 hello_remove 中用于卸载驱动。

结构体 miscdevice 中参数很多，下面几个是常用的。

`int .minor`；设备号，赋值为 MISC_DYNAMIC_MINOR，这个宏定义可以查到为 10

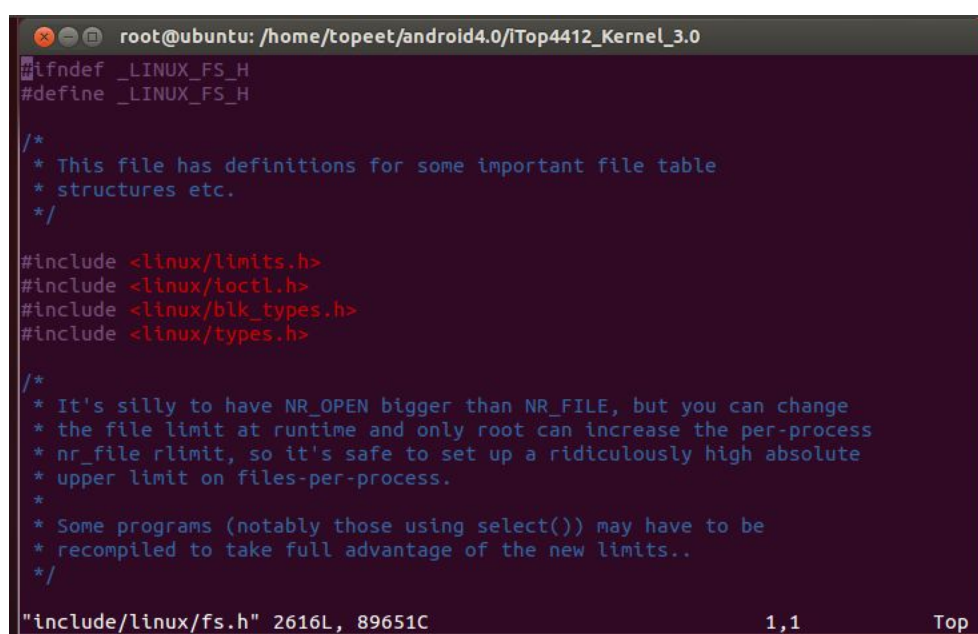
`const char *name`；设备名称

`const struct file_operations *fops`；file_operations 结构体，在下一小节专门介绍。

8.5 file_operations 结构体

file_operations 结构体的成员函数属于驱动设计的主体内容，里面的函数和 Linux 系统给应用程序提供系统接口一一对应。

file_operations 结构体在头文件 “include/linux/fs.h” 中，如下图所示，使用命令 “vim include/linux/fs.h” 打开头文件。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
#ifndef _LINUX_FS_H
#define _LINUX_FS_H

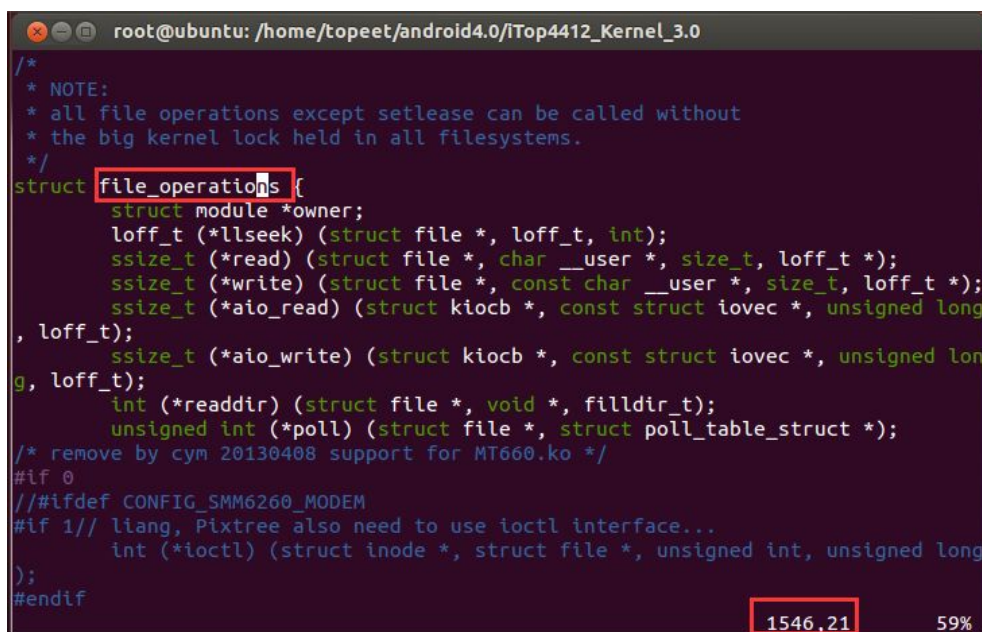
/*
 * This file has definitions for some important file table
 * structures etc.
 */

#include <linux/limits.h>
#include <linux/ioctl.h>
#include <linux/blk_types.h>
#include <linux/types.h>

/*
 * It's silly to have NR_OPEN bigger than NR_FILE, but you can change
 * the file limit at runtime and only root can increase the per-process
 * nr_file rlimit, so it's safe to set up a ridiculously high absolute
 * upper limit on files-per-process.
 *
 * Some programs (notably those using select()) may have to be
 * recompiled to take full advantage of the new limits..
 */

"include/linux/fs.h" 2616L, 89651C 1,1 Top
```

查找 “file_operations”，如下图所示，在 1546 行可以找到其定义。



```
root@ubuntu: /home/topeet/android4.0/iTop4412_Kernel_3.0
/*
 * NOTE:
 * all file operations except setlease can be called without
 * the big kernel lock held in all filesystems.
 */
struct file_operations {
    struct module *owner;
    loff_t (*llseek) (struct file *, loff_t, int);
    ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
    ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
    ssize_t (*aio_read) (struct kiocb *, const struct iovec *, unsigned long
, loff_t);
    ssize_t (*aio_write) (struct kiocb *, const struct iovec *, unsigned lon
g, loff_t);
    int (*readdir) (struct file *, void *, filldir_t);
    unsigned int (*poll) (struct file *, struct poll_table_struct *);
/* remove by cym 20130408 support for MT660.ko */
#ifdef CONFIG_SMM6260_MODEM
//liang, Pixtree also need to use ioctl interface...
    int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long
);
#endif
};
```

如上图所示，可以看到结构体中包含的参数非常多。

struct module *owner;一般是 THIS_MODULE。

int (*open) (struct inode *, struct file *);对应上层的 open 函数，打开文件。

int (*release) (struct inode *, struct file *);对应上层的 close 函数，打开文件操作之后一般需要关闭。

ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);读函数，上层应用从底层读取函数。

ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);写函数，上层应用向底层传输数据。

long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);这个函数功能和写函数稍微有点重合，但是这个函数占用的内存非常小，主要针对 IO 口的控制。

其它结构体中的参数，具体用到再介绍。

8.6 实验操作

下面来具体介绍如何写一个杂项设备。

将上一期中的 “probe_linux_module.c” 改写为 “devicenode_linux_module.c” ,然后添加头文件具体代码等。

如下图所示，将头文件 “linux/platform_device.h” 、 “linux/miscdevice.h” 以及 “linux/fs.h” 添加到文件中。然后定义一个 DEVICE_NAME，将其赋值为 “hello_ctl123”，帮助大家理解这个设备节点名称和前面介绍的注册设备名称是不同的。

```
#include <linux/init.h>
#include <linux/module.h>

/*驱动注册的头文件，包含驱动的结构体和注册和卸载的函数*/
#include <linux/platform_device.h>
/*注册杂项设备头文件*/
#include <linux/miscdevice.h>
/*注册设备节点的文件结构体*/
#include <linux/fs.h>

#define DRIVER_NAME "hello_ctl"
#define DEVICE_NAME "hello_ctl123"
MODULE_LICENSE("Dual BSD/GPL");
MODULE_AUTHOR("TOPEET");
```

如下图所示，向添加 hello_probe 中添加注册杂项设备的函数 misc_register，如下图所示，将 miscdevice 参数定义为 hello_dev。

```
static int hello_probe(struct platform_device *pdv) {  
    printk(KERN_EMERG "\tinitialized\n");  
    misc_register(&hello_dev);  
    return 0;  
}  
  
static int hello_remove(struct platform_device *pdv) {  
    printk(KERN_EMERG "\tremove\n");  
    misc_deregister(&hello_dev);  
    return 0;  
}
```

如下图所示，定义了 miscdevice hello_dev。

参数 minor 为 MISC_DYNAMIC_MINOR，也就是 10

参数 name 为 DEVICE_NAME，也就是 hello_ctl123

参数 fops 为 "hello_ops"

```
static struct miscdevice hello_dev = {  
    .minor = MISC_DYNAMIC_MINOR,  
    .name = DEVICE_NAME,  
    .fops = &hello_ops,  
};
```

如下图所示，定义了 miscdevice hello_dev，其中有四个参数。

参数 owner 为 THIS_MODULE，

参数 open 为 hello_open，

参数 release 为 hello_release，

参数 unlocked_ioctl 为 hello_ioctl，

```
static struct file_operations hello_ops = {  
    .owner = THIS_MODULE,  
    .open = hello_open,  
    .release = hello_release,  
    .unlocked_ioctl = hello_ioctl,  
};
```

如下图所示，对 hello_ops 结构体中的函数挨个定义。

```
static long hello_ioctl( struct file *files, unsigned int cmd, unsigned long arg){  
  
    printk("cmd is %d,arg is %d\n",cmd,arg);  
    return 0;  
}  
  
static int hello_release(struct inode *inode, struct file *file){  
    printk(KERN_EMERG "hello release\n");  
    return 0;  
}  
  
static int hello_open(struct inode *inode, struct file *file){  
    printk(KERN_EMERG "hello open\n");  
    return 0;  
}  
  
static struct file_operations hello_ops = {  
    .owner = THIS_MODULE,  
    .open = hello_open,  
    .release = hello_release,  
    .unlocked_ioctl = hello_ioctl,  
};
```

如上图所示，函数中只做简单的打印。

在调用 hello_ioctl 的时候打印 “cmd is %d,arg is %d” ，

在调用 hello_release 的时候打印 “hello release” ，

在调用 hello_open 的时候打印 “hello open” 。

将前一章实验的 Makefile 文件简单修改一下，将 “probe_linux_module.o” 改为

“devicenode_linux_module.o” ，注释部分用户可以自己修改，如下图所示。

```
#!/bin/bash
#通知编译器我们要编译模块的哪些源码
#这里是编译itop4412 hello.c这个文件编译成中间文件itop4412_hello.o
obj-m += devicenode_linux_module.o

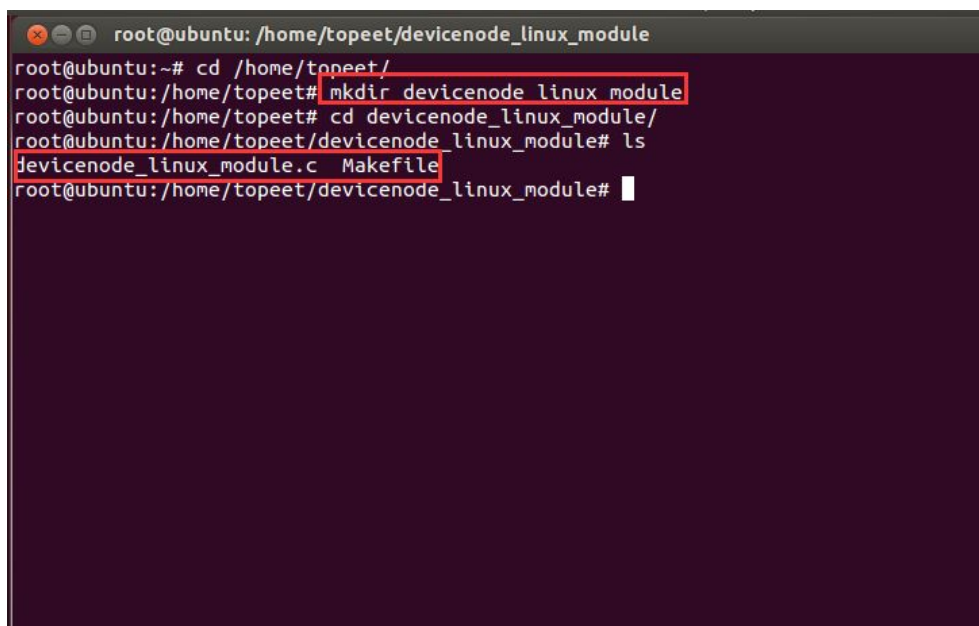
#源码目录变量，这里用户需要根据实际情况选择路径
#作者是将Linux的源码拷贝到目录/home/topeet/android4.0下并解压的
KDIR := /home/topeet/android4.0/iTop4412_Kernel_3.0

#当前目录变量
PWD ?= $(shell pwd)

#make命名默认寻找第一个目标
#make -C就是指调用执行的路径
#$(KDIR) Linux源码目录，作者这里指的是/home/topeet/android4.0/iTop4412_Kernel_3.0
#$(PWD) 当前目录变量
#modules要执行的操作
all:
    make -C $(KDIR) M=$(PWD) modules

#make clean执行的操作是删除后缀为o的文件
clean:
    rm -rf *.o
```

在 Ubuntu 中的目录 “/home/topeet” 下新建 “devicenode_linux_module” 目录，拷贝驱动文件 “devicenode_linux_module.c” 和编译文件 “Makefile” 到新建目录中，如下图所示。



```
root@ubuntu: /home/topeet/devicenode_linux_module
root@ubuntu:~# cd /home/topeet/
root@ubuntu:/home/topeet# mkdir devicenode_linux_module
root@ubuntu:/home/topeet# cd devicenode_linux_module/
root@ubuntu:/home/topeet/devicenode_linux_module# ls
devicenode_linux_module.c  Makefile
root@ubuntu:/home/topeet/devicenode_linux_module#
```


如下图所示，使用命令“make”，编译生成模块“devicenode_linux_module.ko”文件。

```
root@ubuntu: /home/topeet/devicenode_linux_module
root@ubuntu:~# cd /home/topeet/
root@ubuntu:/home/topeet# mkdir devicenode_linux_module
root@ubuntu:/home/topeet# cd devicenode_linux_module/
root@ubuntu:/home/topeet/devicenode_linux_module# ls
devicenode_linux_module.c  Makefile
root@ubuntu:/home/topeet/devicenode_linux_module# make
make -C /home/topeet/android4.0/iTop4412_Kernel_3.0 M=/home/topeet/devicenode_linux_module modules
make[1]: Entering directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
  CC [M] /home/topeet/devicenode_linux_module/devicenode_linux_module.o
/home/topeet/devicenode_linux_module/devicenode_linux_module.c: In function 'hello_ioctl':
/home/topeet/devicenode_linux_module/devicenode_linux_module.c:19: warning: format '%d' expects type 'int', but argument 3 has type 'long unsigned int'
  Building modules, stage 2.
  MODPOST 1 modules
  CC /home/topeet/devicenode_linux_module/devicenode_linux_module.mod.o
  LD [M] /home/topeet/devicenode_linux_module/devicenode_linux_module.ko
make[1]: Leaving directory `/home/topeet/android4.0/iTop4412_Kernel_3.0'
root@ubuntu:/home/topeet/devicenode_linux_module# ls
devicenode_linux_module.c  devicenode_linux_module.mod.o  modules.order
devicenode_linux_module.ko  devicenode_linux_module.o      Module.symvers
devicenode_linux_module.mod.c  Makefile
root@ubuntu:/home/topeet/devicenode_linux_module#
```

启动开发板，拷贝“devicenode_linux_module.ko”到U盘，将U盘插入开发板，加载驱动文件“devicenode_linux_module.ko”，如下图所示。


```
[root@iTOP-4412]# mount /dev/sda1 /mnt/udisk/
[root@iTOP-4412]# insmod /mnt/udisk/devicenode_linux_module.ko
[ 4112.512749] HELLO WORLD enter!
[ 4112.514620] initialized
[ 4112.530912] DriverState is 0
[root@iTOP-4412]#
```

如下图所示，加载之后使用命令“ls /dev”，可以看到新生成了设备节点 hello_ctl123，也就是设备节点和驱动名以及设备名没有一关系，不过最好设备节点的命名便于识别。

```

[root@iTOP-4412]# insmod /mnt/udisk/devicenode_linux_module.ko
[ 4112.512749] HELLO WORLD enter!
[ 4112.514620] initialized
[ 4112.530912] DriverState is 0
[root@iTOP-4412]# ls /dev/
AGPS          leds          ram10         tty1
adc           log           ram11         tty2
alarm         loop0         ram12         tty3
android_adb   loop1         ram13         tty4
ashmem        loop2         ram14         ttyGS0
bus           loop3         ram15         ttyGS1
buzzer_ctl    loop4         ram2          ttyGS2
console       loop5         ram3          ttyGS3
cpu_dma_latency loop6         ram4          ttyS0
exynos-mem    loop7         ram5          ttyS1
fb0           mapper        ram6          ttyS2
fb1           max485_ctl_pin ram7          ttyS3
fb2           mem           ram8          ttySAC0
fb3           mmcblk0       ram9          ttySAC1
fb4           mmcblk0p1     random        ttySAC2
full          mmcblk0p2     rc522         ttySAC3
fuse          mmcblk0p3     relay_ctl     uinput
hello_ctl123  mmcblk0p4     root          urandom
i2c-0         mtp_usb       rtc0          usb_accessory
i2c-1         network_latency rtc1          usbdev1.1
i2c-3         network_throughput s3c-mem       usbdev1.2
i2c-4         null          sda           usbdev1.3
i2c-5         pmem          sdal          usbdev1.4
i2c-7         pmem_gpu1     sg0           watchdog
input         ppp           shm           xt_qtaguid
ion           ptmx          snd           zero
keychord      pts           srp
knem          ram0          srp_ctrl
kmsg          ram1          tty

```



如下图所示，使用命令 “rmmod devicenode_linux_module” 卸载驱动。

```

[root@iTOP-4412]# lsmod
devicenode_linux_module 1873 0 - Live 0xbf000000
[root@iTOP-4412]# rmmod devicenode_linux_module
[ 4312.465228] HELLO WORLD exit!
[ 4312.466824] remove
[root@iTOP-4412]#

```

如下图所示，使用命令 “ls /dev” ，发现设备节点 hello_ctl123 已经去掉了。


```
[root@iTOP-4412]# ls /dev/
AGPS          leds          ram1          srp_ctrl
adc           log           ram10         tty
alarm         loop0        ram11         tty1
android_adb   loop1        ram12         tty2
ashmem        loop2        ram13         tty3
bus           loop3        ram14         tty4
buzzer_ctl    loop4        ram15         ttyGS0
console       loop5        ram2          ttyGS1
cpu_dma_latency loop6       ram3          ttyGS2
exynos-mem    loop7        ram4          ttyGS3
fb0           mapper       ram5          ttyS0
fb1           max485_ctl_pin ram6          ttyS1
fb2           mem          ram7          ttyS2
fb3           mmcblk0      ram8          ttyS3
fb4           mmcblk0p1    ram9          ttySAC0
full         mmcblk0p2    random        ttySAC1
fuse         mmcblk0p3    rc522         ttySAC2
i2c-0        mmcblk0p4    relay_ctl     ttySAC3
i2c-1        mtp_usb      root          uinput
i2c-3        network_latency rtc0          urandom
i2c-4        network_throughput rtc1          usb_accessory
i2c-5        null         s3c-mem       usbdev1.1
i2c-7        pmem         sda           usbdev1.2
input        pmem_gpu1    sda1          usbdev1.3
ion          ppp          sg0           usbdev1.4
keychord     ptmx         shm           watchdog
kmem         pts          snd           xt_qtaguid
kmsg         ram0         srp           zero
```