Linux 驱动开发 四十六：Linux MISC驱动实验

原创

lqonlylove 于 2022-02-04 17:01:08 发布

分类专栏： Linux驱动开发 文章标签： linux

本文链接：https://blog.csdn.net/OnlyLove_/article/details/122784245

misc 的意思是混合、杂项的，因此MISC 驱动也叫做杂项驱动，也就是当我们板子上的某些外设无法进行分类的时候就可以使用 MISC 驱动。

MISC 驱动其实就是最简单的字符设备驱动，通常嵌套在 platform 总线驱动中，实现复杂的驱动。

一、MISC设备驱动简介

所有的 MISC 设备驱动的主设备号都为 10，不同的设备使用不同的从设备号。随着 Linux 字符设备驱动的不断增加，设备号变得越来越紧张，尤其是主设备号，MISC 设备驱动就用于解 决此问题。MISC 设备会自动创建 cdev，不需要像我们以前那样手动创建，因此采用 MISC 设备驱动可以简化字符设备驱动的编写。

我们需要向 Linux 注册一个miscdevice 设备，miscdevice 是一个结构体，定义在文件 include/linux/miscdevice.h 中，内容如下：

```
struct miscdevice  {
 int minor;        /* 子设备号 */
 const char *name;      /* 设备名字 */
 const struct file_operations *fops;  /* 设备操作集 */
 struct list_head list;
 struct device *parent;
 struct device *this_device;
 const struct attribute_group **groups;
 const char *nodename;
 umode_t mode;
};
```

定义一个 MISC 设备（miscdevice 类型）需要设置 minor、name 和 fops 这三个成员变量。minor 表示**子设备号**，MISC 设备的主设备号为 10，这个是**固定的**，需要用户指定子设备 号。name就是此MISC设备名字，当此设备注册成功以后就会在/dev目录下生成一个名为name的设备文件。fops 就是字符设备的操作集合，MISC 设备驱动最终是需要使用用户提供的 fops 操作集合。

Linux 系统已经预定义了一些 MISC 设备的子设备号，这些预定义的子设备号定义在 include/linux/miscdevice.h 文件中，如下所示：

```
/*
 * 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 /* unused */
#define ATIXL_BUSMOUSE_MINOR 3 /* unused */
/*#define AMIGAMOUSE_MINOR 4 FIXME OBSOLETE */
#define ATARIMOUSE_MINOR 5 /* unused */
#define SUN_MOUSE_MINOR  6 /* unused */
#define APOLLO_MOUSE_MINOR 7 /* unused */
#define PC110PAD_MINOR  9 /* unused */
/*#define ADB_MOUSE_MINOR 10 FIXME OBSOLETE */
#define WATCHDOG_MINOR  130 /* Watchdog timer     */
#define TEMP_MINOR  131 /* Temperature Sensor */
#define RTC_MINOR  135
#define EFI_RTC_MINOR  136 /* EFI Time services */
#define VHCI_MINOR  137
#define SUN_OPENPROM_MINOR 139
#define DMAPI_MINOR  140 /* unused */
#define NVRAM_MINOR  144
#define SGI_MMTIMER  153
#define STORE_QUEUE_MINOR 155 /* unused */
#define I2O_MINOR  166
#define MICROCODE_MINOR  184
#define VFIO_MINOR  196
#define TUN_MINOR  200
#define CUSE_MINOR  203
#define MWAVE_MINOR  219 /* ACP/Mwave Modem */
#define MPT_MINOR  220
#define MPT2SAS_MINOR  221
#define MPT3SAS_MINOR  222
#define UINPUT_MINOR  223
#define MISC_MCELOG_MINOR 227
#define HPET_MINOR  228
#define FUSE_MINOR  229
#define KVM_MINOR  232
#define BTRFS_MINOR  234
#define AUTOFS_MINOR  235
#define MAPPER_CTRL_MINOR 236
#define LOOP_CTRL_MINOR  237
#define VHOST_NET_MINOR  238
#define UHID_MINOR  239
#define MISC_DYNAMIC_MINOR 255
```

我们在使用的时候可以从这些预定义的子设备号中挑选一个，当然也可以自己定义，只要这个子设备号没有被其他设备使用接口。

二、MISC 相关 API

1、misc_register

```
/* 注册一个 MISC 设备
 *
 * misc：要注册的MISC 设备。
 * 返回值：负数，失败；0，成功
 */
int misc_register(struct miscdevice *misc)

/* 特别说明：以前我们需要自己调用一堆的函数去创建设备，比如在以前的字符设备驱动中我们会使用如下几个函数完成设备创建过程：
 *
 * alloc_chrdev_region(); 申请设备号
 * cdev_init();     初始化 cdev
 * cdev_add();     添加 cdev
 * class_create();    创建类
 * device_create();    创建设备
 */
```

2、misc_deregister

```
/* 注销一个 MISC 设备
 *
 * misc：要注销的MISC 设备。
 * 返回值：负数，失败；0，成功
 */
int misc_deregister(struct miscdevice *misc)
```

三、MISC 驱动编写思路

1、驱动注册函数调用 platform_driver_register 对 platform 平台驱动进行注册。

2、platform 平台驱动 probe 函数调用 misc_register 函数完成字符设备驱动注册。

3、驱动卸载函数调用 platform_driver_unregister 对 platform 平台驱动进行卸载。

4、platform 平台驱动 remove 函数调用 misc_deregister 函数完成字符设备驱动卸载。

5、misc_register 和 misc_deregister 使用 struct miscdevice 结构体自动注册和注销字符设备驱动。

```
#include "linux/init.h"
#include "linux/module.h"
#include "linux/platform_device.h"
#include "linux/miscdevice.h"
#include "linux/fs.h"

#define NEWCHRDEV_MINOR 144   /* 次设备号 */
#define NEWCHRDEV_NAME  "imx6ull-misc" /* 名子 */


/* 设备操作函数 */
static struct file_operations misc_fops = {
 .owner = THIS_MODULE,
};

/* MISC设备结构体 */
static struct miscdevice miscdev = {
 .minor = NEWCHRDEV_MINOR,
 .name = NEWCHRDEV_NAME,
 .fops = &misc_fops,
};

/*
```

```
 * @description  : flatform驱动的probe函数，当驱动与设备匹配以后此函数就会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_probe(struct platform_device *dev)
{
    int ret = 0;

    printk("misc probe!\r\n");

    /* 注册 misc 设备 */
    ret = misc_register(&miscdev);
    if(ret < 0){
  printk("misc device register failed!\r\n");
  return -EFAULT;
 }

    return 0;
}

/*
 * @description  : platform驱动的remove函数，移除platform驱动的时候此函数会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_remove(struct platform_device *dev)
{
    printk("misc remove!\r\n");

    /* 注销 misc 设备 */
    misc_deregister(&miscdev);
    return 0;
}

/* 匹配列表 */
static const struct of_device_id misc_of_match[] = {
 { .compatible = "gpioled" },
 { /* Sentinel */ }
};

/* platform驱动结构体 */
static struct platform_driver led_driver = {
 .driver  = {
  .name = "imx6ul-misc",   /* 驱动名字，用于和设备匹配 */
        .of_match_table = misc_of_match, /* 设备树匹配表     */
 },
 .probe  = misc_probe,
 .remove  = misc_remove,
};

/*
 * @description : 驱动模块加载函数
 * @param   : 无
 * @return   : 无
 */
static int __init misc_device_init(void)
{
    return platform_driver_register(&led_driver);
}
```

```
/*
 * @description : 驱动模块卸载函数
 * @param   : 无
 * @return   : 无
 */
static void __exit misc_device_exit(void)
{
    platform_driver_unregister(&led_driver);
}

module_init(misc_device_init);
module_exit(misc_device_exit);
MODULE_LICENSE("GPL");
```

四、MISC 驱动源码

通过 led 灯相关驱动进行验证。

1、Makefile

```
KERNELDIR := /home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga
CURRENT_PATH := $(shell pwd)
obj-m := misc.o

build: kernel_modules

kernel_modules:
 $(MAKE) -C $(KERNELDIR) M=$(CURRENT_PATH) modules
clean:
 $(MAKE) -C $(KERNELDIR) M=$(CURRENT_PATH) clean
```

2、搭建 platform 平台框架

```
#include "linux/init.h"
#include "linux/module.h"
#include "linux/platform_device.h"
#include "linux/miscdevice.h"
#include "linux/fs.h"

/*
 * @description  : flatform驱动的probe函数，当驱动与设备匹配以后此函数就会执行
 * @param - dev  : platform设备
 * @return    : 0，成功;其他负值,失败
 */
static int misc_probe(struct platform_device *dev)
{
    printk("misc probe!\r\n");
    return 0;
}

/*
 * @description  : platform驱动的remove函数，移除platform驱动的时候此函数会执行
 * @param - dev  : platform设备
 * @return    : 0，成功;其他负值,失败
 */
static int misc_remove(struct platform_device *dev)
{
    printk("misc remove!\r\n");
```

```
    return 0;
}

/* 匹配列表 */
static const struct of_device_id misc_of_match[] = {
 { .compatible = "gpioled" },
 { /* Sentinel */ }
};

/* platform驱动结构体 */
static struct platform_driver led_driver = {
 .driver  = {
  .name = "imx6ul-misc",   /* 驱动名字，用于和设备匹配 */
        .of_match_table = misc_of_match, /* 设备树匹配表    */
 },
 .probe  = misc_probe,
 .remove  = misc_remove,
};

/*
 * @description : 驱动模块加载函数
 * @param   : 无
 * @return   : 无
 */
static int __init misc_device_init(void)
{
    return platform_driver_register(&led_driver);
}

/*
 * @description : 驱动模块卸载函数
 * @param   : 无
 * @return   : 无
 */
static void __exit misc_device_exit(void)
{
    platform_driver_unregister(&led_driver);
}

module_init(misc_device_init);
module_exit(misc_device_exit);
MODULE_LICENSE("GPL");
```

3、搭建 misc 平台框架

```
#include "linux/init.h"
#include "linux/module.h"
#include "linux/platform_device.h"
#include "linux/miscdevice.h"
#include "linux/fs.h"

#define NEWCHRDEV_MINOR 144   /* 次设备号 */
#define NEWCHRDEV_NAME  "imx6ull-misc" /* 名子 */


/* 设备操作函数 */
static struct file_operations misc_fops = {
 .owner = THIS_MODULE,
};
```

```c
/* MISC设备结构体 */
static struct miscdevice miscdev = {
 .minor = NEWCHRDEV_MINOR,
 .name = NEWCHRDEV_NAME,
 .fops = &misc_fops,
};

/*
 * @description  : flatform驱动的probe函数，当驱动与设备匹配以后此函数就会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_probe(struct platform_device *dev)
{
    int ret = 0;

    printk("misc probe!\r\n");

    /* 注册 misc 设备 */
    ret = misc_register(&miscdev);
    if(ret < 0){
  printk("misc device register failed!\r\n");
  return -EFAULT;
 }

    return 0;
}

/*
 * @description  : platform驱动的remove函数，移除platform驱动的时候此函数会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_remove(struct platform_device *dev)
{
    printk("misc remove!\r\n");

    /* 注销 misc 设备 */
    misc_deregister(&miscdev);
    return 0;
}

/* 匹配列表 */
static const struct of_device_id misc_of_match[] = {
 { .compatible = "gpioled" },
 { /* Sentinel */ }
};

/* platform驱动结构体 */
static struct platform_driver led_driver = {
 .driver  = {
  .name = "imx6ul-misc",   /* 驱动名字，用于和设备匹配 */
        .of_match_table = misc_of_match, /* 设备树匹配表     */
 },
 .probe  = misc_probe,
 .remove  = misc_remove,
};

/*
```

```
 * @description : 驱动模块加载函数
 * @param   : 无
 * @return   : 无
 */
static int __init misc_device_init(void)
{
    return platform_driver_register(&led_driver);
}

/*
 * @description : 驱动模块卸载函数
 * @param   : 无
 * @return   : 无
 */
static void __exit misc_device_exit(void)
{
    platform_driver_unregister(&led_driver);
}

module_init(misc_device_init);
module_exit(misc_device_exit);
MODULE_LICENSE("GPL");
```

编译：

```
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
Makefile  misc.c
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ make
make -C /home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga M=/home/onlylove/linux/driver/linux_driver/11_misc modules
make[1]: Entering directory '/home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga'
  CC [M]  /home/onlylove/linux/driver/linux_driver/11_misc/misc.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/onlylove/linux/driver/linux_driver/11_misc/misc.mod.o
  LD [M]  /home/onlylove/linux/driver/linux_driver/11_misc/misc.ko
make[1]: Leaving directory '/home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga'
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
Makefile  misc.c  misc.ko  misc.mod.c  misc.mod.o  misc.o  modules.order  Module.symvers
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$
```

测试：

```
/ # ls
bin      etc      linuxrc  mnt      root     sys      usr
dev      lib      misc.ko  proc     sbin     tmp
/ # lsmod
Module                  Size  Used by    Not tainted
/ # ls /dev/imx6ull-misc
ls: /dev/imx6ull-misc: No such file or directory
/ #
/ # insmod misc.ko
misc probe!
/ # lsmod
Module                  Size  Used by    Tainted: G
misc                    1800  0
/ # ls /dev/imx6ull-misc -l
crw-rw----    1 0        0          10, 144 Jan  1 00:31 /dev/imx6ull-misc
/ # rmmod misc.ko
misc remove!
/ #
/ # lsmod
Module                  Size  Used by    Tainted: G
/ #
/ # ls /dev/imx6ull-misc -l
ls: /dev/imx6ull-misc: No such file or directory
/ #
```

4、添加 led 驱动源码

```
#include "linux/init.h"
#include "linux/module.h"
#include "linux/platform_device.h"
#include "linux/miscdevice.h"
#include "linux/fs.h"
#include "linux/of.h"
#include "asm/uaccess.h"
#include "linux/of_gpio.h"
#include "linux/gpio.h"

#define NEWCHRDEV_MINOR 144   /* 次设备号 */
#define NEWCHRDEV_NAME  "imx6ull-misc" /* 名子 */

#define LEDOFF    0
#define LEDON    1

typedef struct{
    struct device_node *node; /* LED设备节点 (声明在 "linux/of.h")*/
 int led0;     /* LED灯GPIO标号 */
}newchrdev_t;

newchrdev_t newchrdev;

int imx6ull_led_init(void)
{
    newchrdev.node = of_find_node_by_path("/gpioled");
 if (newchrdev.node == NULL){
  printk("gpioled node nost find!\r\n");
  return -EINVAL;
 }

 newchrdev.led0 = of_get_named_gpio(newchrdev.node, "led-gpio", 0);
```

```
 if (newchrdev.led0 < 0) {
  printk("can't get led-gpio\r\n");
  return -EINVAL;
 }

 gpio_request(newchrdev.led0, "led0");
 gpio_direction_output(newchrdev.led0, 1); /* led0 IO设置为输出，默认高电平 */
    printk("imx6ull led init!\r\n");
    return 0;
}

int imx6ull_led_exit(void)
{
    gpio_set_value(newchrdev.led0, 1);  /* 卸载驱动的时候关闭LED */
    printk("imx6ull led exit!\r\n");
    return 0;
}

/*
 * @description  : LED打开/关闭
 * @param - sta  : LEDON(0) 打开LED，LEDOFF(1) 关闭LED
 * @return    : 无
 */
void led0_switch(u8 sta)
{
 if (sta == LEDON )
  gpio_set_value(newchrdev.led0, 0);
 else if (sta == LEDOFF)
  gpio_set_value(newchrdev.led0, 1);
}

// struct inode 声明在 linux/fs.h 中
// struct file 声明在 linux/fs.h 中
int led_open (struct inode *i, struct file *f)
{
    return 0;
}

int led_release (struct inode *i, struct file *f)
{
    return 0;
}

// ssize_t 定义在 linux/types.h 中
// __user 定义在 linux/compiler.h 中
// size_t 定义在 linux/types.h 中
// loff_t 定义在 linux/types.h 中
ssize_t led_read (struct file *f, char __user *b, size_t c, loff_t * l)
{
    printk("led read!\r\n");
    return 0;
}

/*
 * @description : 向设备写数据
 * @param - f  : 设备文件，表示打开的文件描述符
 * @param - b  : 要写给设备写入的数据
 * @param - c  : 要写入的数据长度
 * @param - l  : 相对于文件首地址的偏移
 * @return   : 写入的字节数，如果为负值，表示写入失败
```

```
 */
ssize_t led_write (struct file *f, const char __user *b, size_t c, loff_t *l)
{
    int retvalue;
 unsigned char databuf[2];
 unsigned char ledstat;

 retvalue = copy_from_user(databuf, b, c);
 if(retvalue < 0) {

  printk("kernel write failed!\r\n");
  return -EFAULT;
 }

 ledstat = databuf[0];
 if (ledstat == LEDON) {
  led0_switch(LEDON);
 } else if (ledstat == LEDOFF) {
  led0_switch(LEDOFF);
 }
    printk("led write!\r\n");

    return 0;
}

/* 设备操作函数 */
static struct file_operations misc_fops = {
 .owner = THIS_MODULE,
    .open = led_open,
    .release = led_release,
    .read = led_read,
    .write = led_write,
};

/* MISC设备结构体 */
static struct miscdevice miscdev = {
 .minor = NEWCHRDEV_MINOR,
 .name = NEWCHRDEV_NAME,
 .fops = &misc_fops,
};

/*
 * @description  : flatform驱动的probe函数，当驱动与设备匹配以后此函数就会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_probe(struct platform_device *dev)
{
    int ret = 0;

    printk("misc probe!\r\n");
    ret = imx6ull_led_init();
    if(ret != 0){
        printk("imx6ull led init failed!\r\n");
        return -EFAULT;
    }
    /* 注册 misc 设备 */
    ret = misc_register(&miscdev);
    if(ret < 0){
```

```
  printk("misc device register failed!\r\n");
  return -EFAULT;
 }

    return 0;
}

/*
 * @description  : platform驱动的remove函数，移除platform驱动的时候此函数会执行
 * @param - dev  : platform设备
 * @return       : 0，成功;其他负值,失败
 */
static int misc_remove(struct platform_device *dev)
{
    printk("misc remove!\r\n");
    imx6ull_led_exit();
    /* 注销 misc 设备 */
    misc_deregister(&miscdev);
    return 0;
}

/* 匹配列表 */
static const struct of_device_id misc_of_match[] = {
 { .compatible = "gpioled" },
 { /* Sentinel */ }
};

/* platform驱动结构体 */
static struct platform_driver led_driver = {
 .driver  = {
  .name = "imx6ul-misc",   /* 驱动名字，用于和设备匹配 */
        .of_match_table = misc_of_match, /* 设备树匹配表    */
 },
 .probe  = misc_probe,
 .remove  = misc_remove,
};

/*
 * @description : 驱动模块加载函数
 * @param   : 无
 * @return   : 无
 */
static int __init misc_device_init(void)
{
    return platform_driver_register(&led_driver);
}

/*
 * @description : 驱动模块卸载函数
 * @param   : 无
 * @return   : 无
 */
static void __exit misc_device_exit(void)
{
    platform_driver_unregister(&led_driver);
}

module_init(misc_device_init);
module_exit(misc_device_exit);
MODULE_LICENSE("GPL");
```

五、app 程序

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include "stdio.h"

int main(int argc, char *argv[])
{
    int fd = 0, retvalue = 0;
    char writebuf[1] = "";
    fd = open(argv[1],O_RDWR);
    if(fd < 0){
        printf("Can't open file %s\r\n", argv[1]);
        return -1;
    }
     writebuf[0] = atoi(argv[2]);
    // 打开 led
    write(fd, writebuf, 1);
    retvalue = close(fd);
    if(retvalue < 0){
        printf("Can't close file %s\r\n", argv[1]);
        return -1;
    }

    return 0;
}
```

编译：

```
arm-linux-gnueabihf-gcc app.c -o app
```

六、编译

1、驱动编译

```
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
app.c  Makefile  misc.c
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ make
make -C /home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga M=/home/onlylove/linux/driver/linu
x_driver/11_misc modules
make[1]: Entering directory '/home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga'
  CC [M]  /home/onlylove/linux/driver/linux_driver/11_misc/misc.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /home/onlylove/linux/driver/linux_driver/11_misc/misc.mod.o
  LD [M]  /home/onlylove/linux/driver/linux_driver/11_misc/misc.ko
make[1]: Leaving directory '/home/onlylove/linux/linux/lq_linux/linux-imx-rel_imx_4.1.15_2.1.0_ga'
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
app.c  Makefile  misc.c  misc.ko  misc.mod.c  misc.mod.o  misc.o  modules.order  Module.symvers
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ cp misc.ko /home/onlylove/linux/nfs/rootfs-l
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$
```

2、app 程序编译

```
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
app.c  Makefile  misc.c  misc.ko  misc.mod.c  misc.mod.o  misc.o  modules.order  Module.symvers
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ arm-linux-gnueabihf-gcc app.c -o app
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$ ls
app  app.c  Makefile  misc.c  misc.ko  misc.mod.c  misc.mod.o  misc.o  modules.order  Module.symvers
onlylove@ubuntu:~/linux/driver/linux_driver/11_misc$
```

七、测试

```
/ # ls
app      dev      lib      misc.ko  proc     sbin     tmp
bin      etc      linuxrc  mnt      root     sys      usr
/ # lsmod
Module                  Size  Used by    Not tainted
/ # ls /dev/imx6ull-misc -l
ls: /dev/imx6ull-misc: No such file or directory
/ #
/ # insmod misc.ko
misc probe!
imx6ull led init!
/ #
/ # lsmod
Module                  Size  Used by    Tainted: G
misc                    3155  0
/ #
/ # ls /dev/imx6ull-misc -l
crw-rw----    1 0        0          10, 144 Jan  1 01:01 /dev/imx6ull-misc
/ #
/ # rmmod misc.ko
misc remove!
imx6ull led exit!
/ #
/ # lsmod
Module                  Size  Used by    Tainted: G
/ #
/ # insmod misc.ko
misc probe!
imx6ull led init!
/ #
/ # lsmod
Module                  Size  Used by    Tainted: G
misc                    3155  0
/ #
/ # ls /dev/imx6ull-misc -l
crw-rw----    1 0        0          10, 144 Jan  1 01:03 /dev/imx6ull-misc
/ #
/ # ./app /dev/imx6ull-misc 0
led write!
/ # ./app /dev/imx6ull-misc 1
led write!
/ # ./app /dev/imx6ull-misc 0
led write!
/ # ./app /dev/imx6ull-misc 1
led write!
/ # ./app /dev/imx6ull-misc 0
led write!
/ # ./app /dev/imx6ull-misc 1
led write!
/ #
/ # rmmod misc.ko
misc remove!
imx6ull led exit!
/ #
```

通过以上测试，led 工作正常。