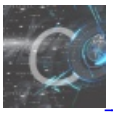
# Linux驱动编程——misc设备驱动框架

原创

SixerL 于 2021-09-14 19:56:44 发布 1114 收藏

分类专栏： Linux驱动编程 文章标签： linux 物联网 运维







## Linux驱动编程——misc设备驱动框架

### 主要概念：

misc：杂项设备

杂项设备是字符设备的一种，杂项设备可以自动生成设备节点。

设备节点： dev 目录下，全部都是生成的设备节点

`cat /proc/misc` 查看系统里的砸向设备

misc设备主设备号都是10

设备号：

主设备号 用来标识一个类型的驱动

次设备号 用来标识同一类型中不同的设备号

`cat /proc/devices` 查看设备号

### 实验目的：

熟悉杂项设备驱动编写框架，编写一个简易的杂项设备驱动，并编译加载到系统上。

### 主要函数调用：

```
struct miscdevice {      //misc设备结构体
    int minor;         //次设备号
    const char *name;       //设备节点的名字
    const struct file_operations *fops; //文件操作集
    struct list_head list;
    struct device *parent;
    struct device *this_device;
    const struct attribute_group **groups;
    const char *nodename;
    umode_t mode;
};

int misc_register(struct miscdevice *misc);  //向系统中注册一个 misc 设备
int misc_deregister(struct miscdevice *misc) //注销掉 misc 设备
```

## 代码：

```
//misc.c
#include <linux/init.h>
#include <linux/module.h>
#include <linux/miscdevice.h>
#include <linux/fs.h>

struct file_operations misc_fops = {
 .owner = THIS_MODULE,
};

struct miscdevice misc_dev = {
 .minor = MISC_DYNAMIC_MINOR,
 .name = "hello_misc",
 .fops = &misc_fops,
};

static int misc_init(void)
{
 int ret;
 ret = misc_register(&misc_dev);
 if(ret < 0)
 {
  printk("register is error\r\n");
  return -1;
 }

 printk("misc register is succeed\r\n");

 return 0;
}

static void misc_exit(void)
{
 misc_deregister(&misc_dev);
 printk("misc deregister!\r\n");
}

module_init(misc_init);
module_exit(misc_exit);

MODULE_LICENSE("GPL");
MODULE_AUTHOR("SIXER");
```

Makefile

```
KERNELDIR := /home/sixer/imx_kernel/linux-imx-rel_imx_4.1.15_2.1.0_ga
CURRENT_PATH := $(shell pwd)

export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabihf-

target := misc
obj-m := $(target).o
APP_NAME := $(target)_app

build: kernel_modules

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

clean:
    $(MAKE) -C $(KERNELDIR) M=$(CURRENT_PATH) clean
#   rm -rf $(APP_NAME)

dir_exist = $(shell if [ -d "/nfsroot/$(target)/" ]; then echo "exist"; else echo "noexist"; fi)
$(info $(dir_exist))

install:
ifeq ("$(dir_exist)", "noexist")
    $(shell mkdir /nfsroot/$(target))
    #"文件夹不存在，已重新创建文件夹"
endif
#   arm-linux-gnueabihf-gcc $(APP_NAME).c -o  $(APP_NAME)
    cp *.ko  /nfsroot/$(target)/
```

## 编译运行：

`make` 编译

`make install` 将目标文件拷贝到 NFS 目录中

```
sixer@ubuntu:~/imx_usr/driver/00_misc$ make
noexist
make -C /home/sixer/imx_kernel/linux-imx-rel_imx_4.1.15_2.1.0_ga M=/home/sixer/imx_usr/driver/00_misc modules
make[1]: Entering directory '/home/sixer/imx_kernel/linux-imx-rel_imx_4.1.15_2.1.0_ga'
noexist
  CC [M]  /home/sixer/imx_usr/driver/00_misc/misc.o
  Building modules, stage 2.
noexist
  MODPOST 1 modules
  CC      /home/sixer/imx_usr/driver/00_misc/misc.mod.o
  LD [M]  /home/sixer/imx_usr/driver/00_misc/misc.ko
make[1]: Leaving directory '/home/sixer/imx_kernel/linux-imx-rel_imx_4.1.15_2.1.0_ga'
sixer@ubuntu:~/imx_usr/driver/00_misc$ make install
noexist
#"文件夹不存在，已重新创建文件夹"
cp *.ko  /nfsroot/misc/
sixer@ubuntu:~/imx_usr/driver/00_misc$
```

## 实验现象：

```
[root@iTOP-iMX6UL]# cd misc/
[root@iTOP-iMX6UL]# ls
misc.ko
[root@iTOP-iMX6UL]# insmod misc
insmod: can't insert 'misc': No such file or directory
[root@iTOP-iMX6UL]# insmod misc.ko
misc register is succeed
[root@iTOP-iMX6UL]# ls /dev/hello_misc
/dev/hello_misc
[root@iTOP-iMX6UL]# rmmod misc
misc deregister!
```

驱动成功加载，在/dev目录中生成 hello_misc 设备节点文件。

## 总结：

注册 misc 设备在驱动框架的基础上增加了miscdevice结构体，另外多了注册和注销两个操作，加载驱动模块 misc.ko 的过程中，系统是自动在 /dev 目录下生成了 hello_misc 节点文件，这样应用程序就可以通过设备节点文件访问系统内核资源了。

注册misc 设备的三个步骤：

填充 miscdevice 这个结构体
填充 file_operations 这个结构体misc.c
注册杂项设备并生生成设备节点

## 公众号：

InsertingAll