华清远见 FARSIGHT 嵌入式培训专家

嵌入式学院 华清远见旗下品牌

Linux字符设备驱动

浅析内核设计与实现

潘友华

版权

嵌入式学院
华清远见旗下品牌

- 华清远见嵌入式培训中心版权所有；
- 未经华清远见明确许可，不能为任何目的以任何形式复制或传播此文档的任何部分；
- 本文档包含的信息如有更改，恕不另行通知；
- 保留所有权利。

- 什么是字符设备驱动
- 字符设备驱动框架
- 注册字符设备驱动过程
- 创建设备节点
- 打开字符设备如何完成
- 简单模拟字符设备驱动

什么是字符设备

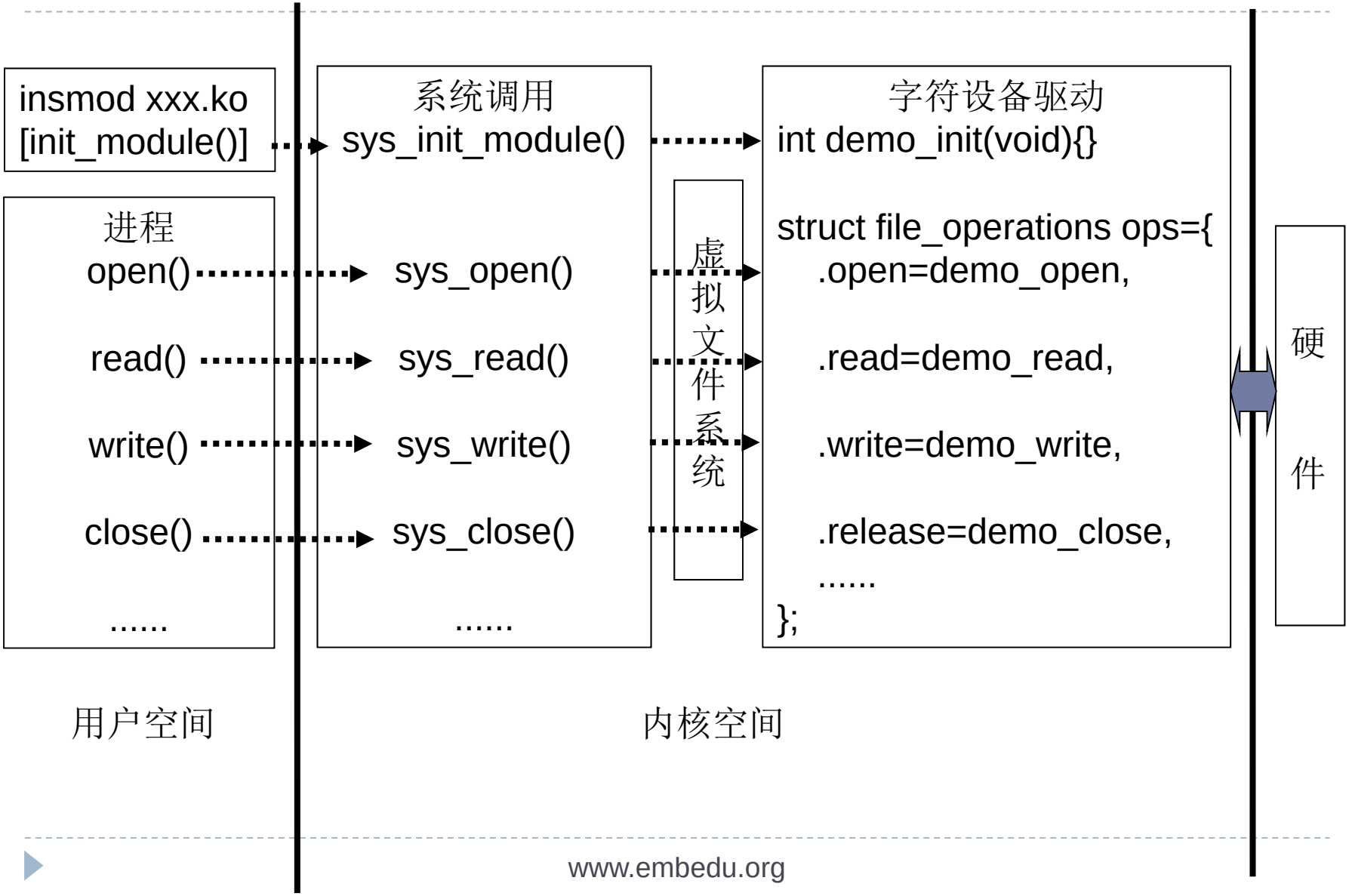
Linux设备驱动分为三种类型：字符设备驱动（char device driver）、块设备驱动（block device driver）和网络设备驱动（network device driver）。
字符设备（char device），就是以字节为单位进行顺序访问的一类设备的总称。典型的字符设备有：键盘，串口，控制台等。字符设备驱动（char device driver）就是提供操作字符设备的机制。

字符设备驱动框架

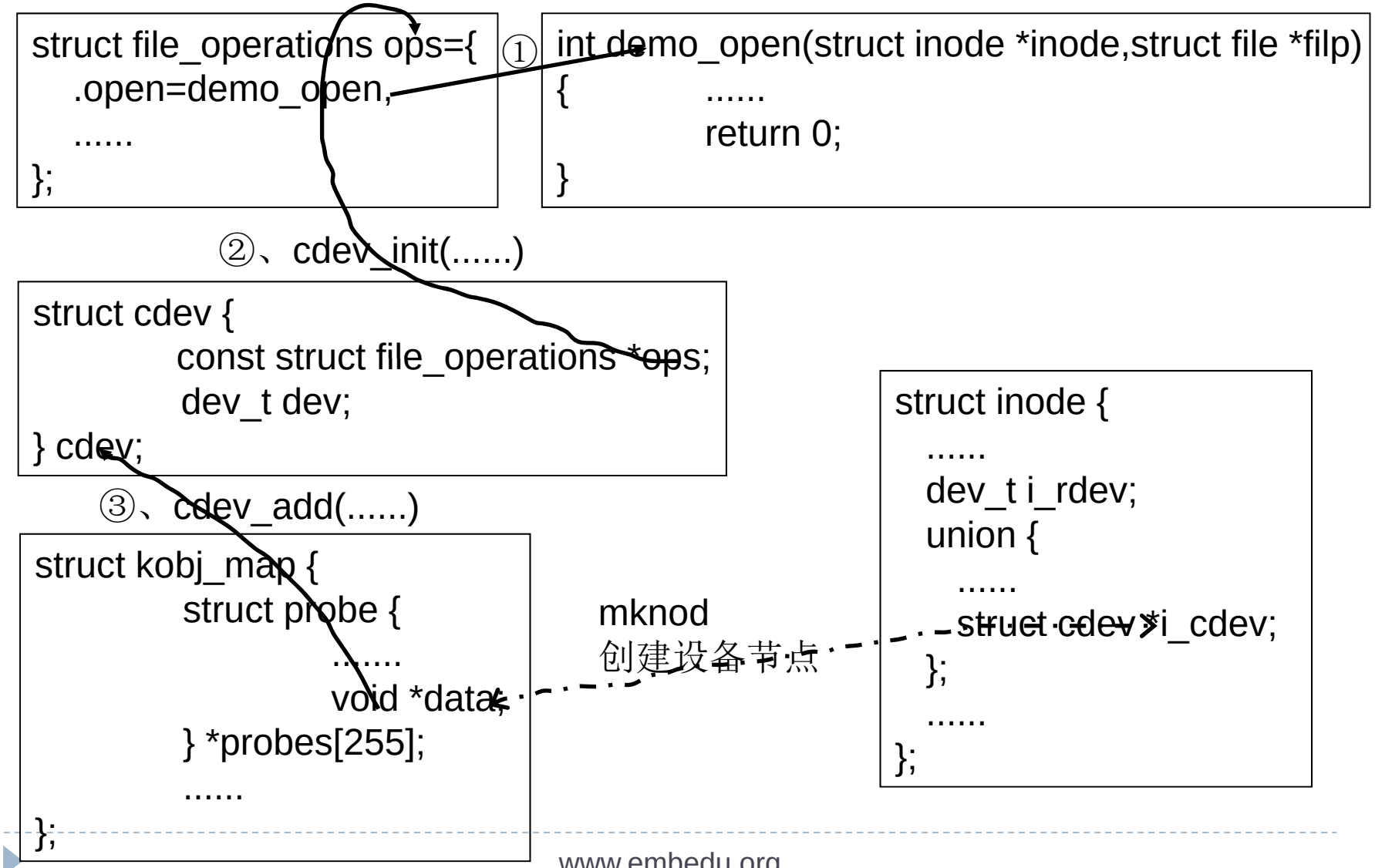
嵌入式学院
华清远见旗下品牌

1、申请设备号
静态指定设备号
```
int register_chrdev_region(dev_t from, unsigned count, const char *name);
```
动态申请设备号
```
int alloc_chrdev_region(dev_t *dev, unsigned baseminor, unsigned count,
                        const char *name);
```
2、操作方法集合
```
struct file_operations {
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        int (*open) (struct inode *, struct file *);
        int (*release) (struct inode *, struct file *);
        ......
};
```
3、注册字符设备驱动
初始化cdev对象
```
void cdev_init(struct cdev *cdev, const struct file_operations *fops);
```
注册字符设备
```
int cdev_add(struct cdev *p, dev_t dev, unsigned count);
```

字符设备驱动框架


insmod xxx.ko
[init_module()]
进程
open()
read()
write()
close()
.....
系统调用
sys_init_module()
sys_open()
sys_read()
sys_write()
sys_close()
.....
虚拟文件系统
字符设备驱动
int demo_init(void){}
struct file_operations ops={
.open=demo_open,
.read=demo_read,
.write=demo_write,
.release=demo_close,
.....
};
硬件
用户空间
内核空间

注册字符设备驱动过程


struct file_operations ops={
.open=demo_open,
.....
};
①
int demo_open(struct inode *inode,struct file *filp)
{
.....
return 0;
}
②、cdev_init(.....)
struct cdev {
const struct file_operations *ops;
dev_t dev;
} cdev;
③、cdev_add(.....)
struct kobj_map {
struct probe {
.....
void *data;
} *probes[255];
.....
};
mknod
创建设备节点
struct inode {
.....
dev_t i_rdev;
union {
.....
struct cdev *i_cdev;
};
.....
};

创建设备节点


用户空间
mknod("/dev/xxx",...)
内核空间
sys_mknod("/dev/xxx",...)
sys_mknodat(...)
vfs_mknod(...)
回调文件系统mknod方法
以为ext4文件系统
ext4_mknod(...)
init_special_inode(...)
登记回调函数chrdev_open(...)

打开字符设备如何完成


用户空间
open("/dev/xxx",...)
内核空间
sys_open("/dev/xxx",...)
do_filp_open(...)
1、创建file对象
2、查找inode
3、初始化file对象
4、回调inode缺省open()方法
chrdev_open(...)
1、查找cdev
2、修改file的操作方法集
3、回调用驱动登记的open()方法

简单模拟字符设备驱动

linux字符设备驱动的基本设计思路，简而言之就是：就是把一组控制设备的方法封装在一个对象中，以所谓注册的形式把对象的首地址放到特殊的数据区，供上层使用者以某种形式查找并回调这些方法。

mycdev.c模拟字符设备操作

myinode.c模拟虚拟文件系统操作

demo.c模拟驱动及系统调用程序

谢谢！

潘友华
panyh@farsight.com.cn