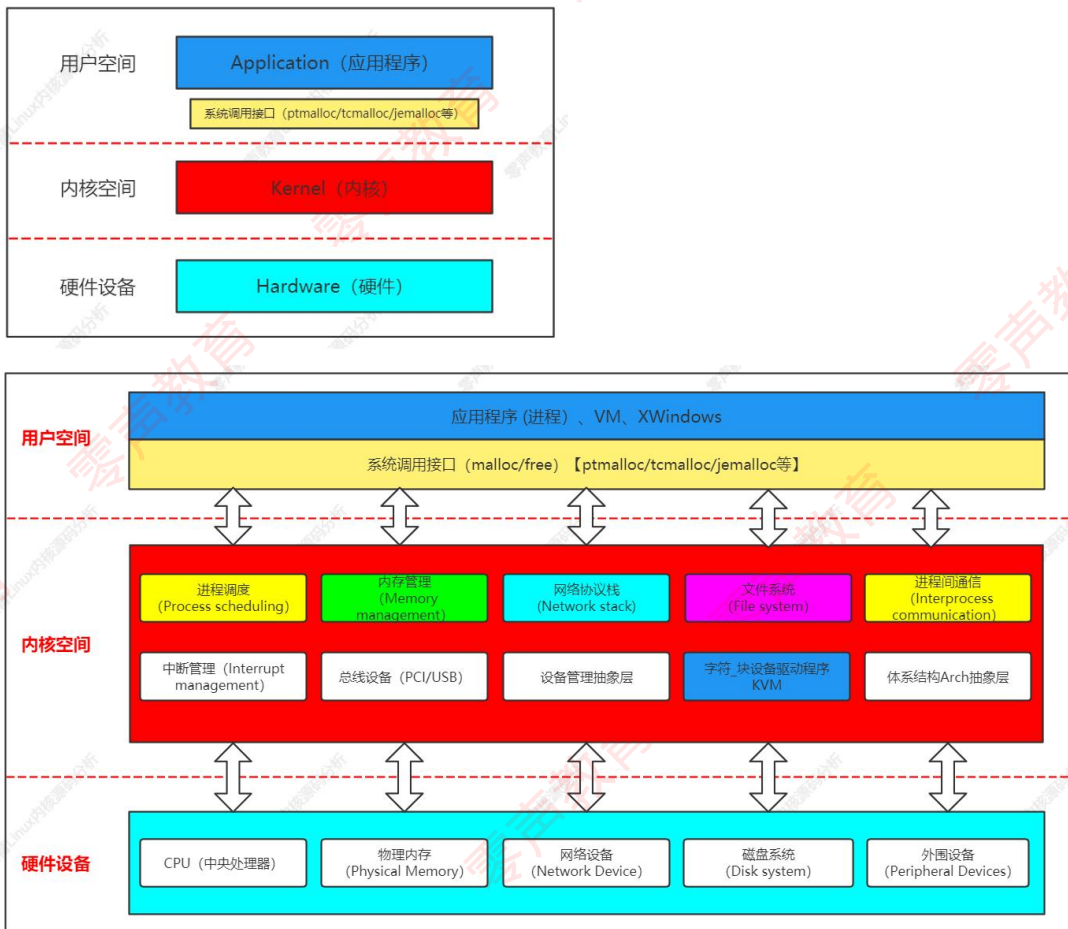


## 第 002 讲 Linux 内核《设备驱动管理专题》

Linux 内核源码分析架构图



### 一、块设备驱动分析

#### 1、块设备基础

块设备是一种能够随机访问的存储介质，与字符设备不同，块设备能够保存文件系统数据。块设备特性如下：

- 可以在一次 I/O 操作中传送固定大小的数据块。
- 可以随机访问设备中所存放的块：传送数据块所需要的时间独立于块在设备中的位置，也独立于当前设备的状态。

#### 2、常见存储技术与设备驱动程序相关

IDE/ATA (PC 存储接口技术)：

```
drivers > ide > C ide-io.c > ...  
27 #include <linux/module.h>
```

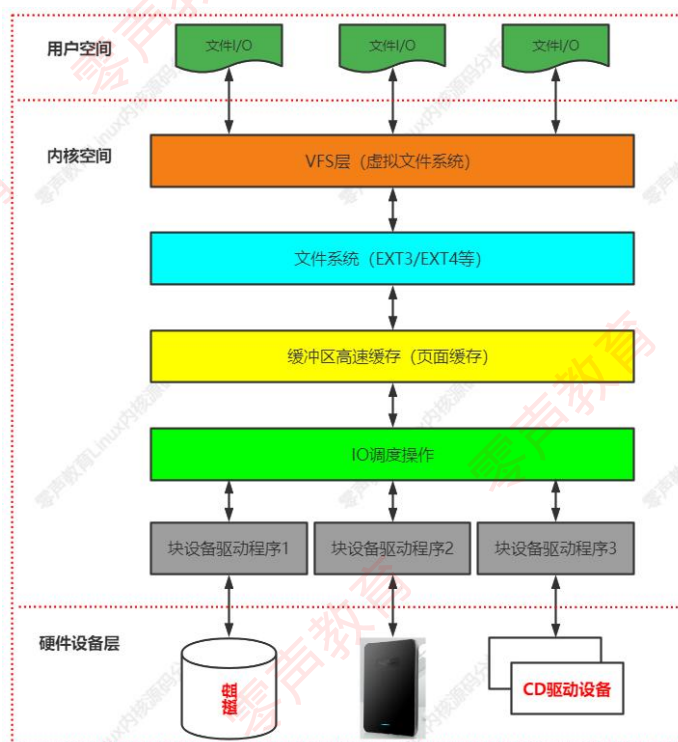
```
fs > C block_dev.c > ...  
8 #include <linux/init.h>  
9 #include <linux/mm.h>
```

```
drivers > ide > C ide-probe.c
18
19 #include <linux/module.h>
20 #include <linux/types.h>
21 #include <linux/string.h>
```

SCSI (服务器环境常用存储技术) :

```
Linux-5.0  drivers > ide > C ide-probe.c > ...
> sbus
> vscsi
18
19 #include <linux/module.h>
20 #include <linux/types.h>
21 #include <linux/string.h>
22 #include <linux/kernel.h>
23 #include <linux/timer.h>
```

### 3、Linux 块 I/O 层架构视图



### 4、Linux 内核块驱动程序数据结构和方法

```
include > linux > C genhd.h > genhd > devnode
175 struct gendisk {
176     /* major, first_minor and minors are input parameters only,
177      * don't use directly. Use disk_devt() and disk_max_parts().
178      */
179     int major; /* major number of driver */
180     int first_minor;
181     int minors; /* maximum number of minors, =1 for
```

每个块驱动程序对应 I/O 请求队列使用 request\_queue 结构体描述如下:

```
include > linux > C blkdev.h > blkdev > request
131
132 struct request {
133     struct request_queue *q;
134     struct blk_mq_ctx *mq_ctx;
135     struct blk_mq_hw_ctx *mq_hctx;
136 }
```

块驱动对应操作入口函数为:

```
include > linux > C blkdev.h > bdev_read_page(block_device *, sector_t, page *)
1642
1643 struct block_device_operations {
1644     int (*open) (struct block_device *, fmode_t);
1645     void (*release) (struct gendisk *, fmode_t);
1646     int (*rw_page)(struct block_device *, sector_t, struct page *, unsigned int);
1647     int (*ioctl) (struct block_device *, fmode_t, unsigned, unsigned long);
1648     int (*compat_ioctl) (struct block_device *, fmode_t, unsigned, unsigned long);
1649     unsigned int (*check_events) (struct gendisk *disk,
1650     | | | | | unsigned int clearing);
1651     /* ->media_changed() is DEPRECATED, use ->check_events() instead */
1652     int (*media_changed) (struct gendisk *);
1653     void (*unlock_native_capacity) (struct gendisk *);
1654     int (*revalidate_disk) (struct gendisk *);
1655     int (*getgeo)(struct block_device *, struct hd_geometry *);
1656     /* this callback is with swap_lock and sometimes page table lock held */
1657     void (*swap_slot_free_notify) (struct block_device *, unsigned long);
1658     int (*report_zones)(struct gendisk *, sector_t sector,
1659     | | | struct blk_zone *zones, unsigned int *nr_zones,
1660     | | | gfp_t gfp_mask);
1661     struct module *owner;
1662     const struct pr_ops *pr_ops;
1663 };
```

## 二、字符设备驱动分析

### 1、字符设备特性

- 可以在一次 I/O 操作中传送任意大小的数据。实际上, 诸如打印机之类的字符设备可以一次传送一个字节, 而诸如磁带之类的设备可以一次传送可变大小的数据块。
- 通常访问连续的字符。

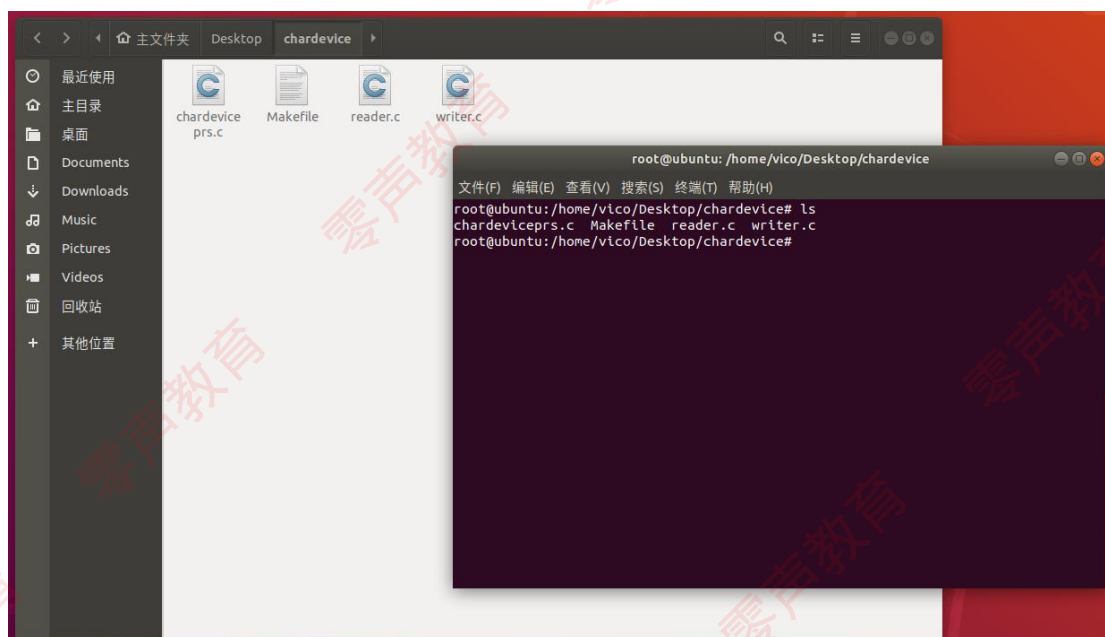
### 2、字符设备数据结构

```
include > linux > C cdev.h > ...
15 struct cdev {
16     struct kobject kobj;
17     struct module *owner;
18     const struct file_operations *ops;
19     struct list_head list;
20     dev_t dev;
21     unsigned int count;
22 } __randomize_layout;
23
```

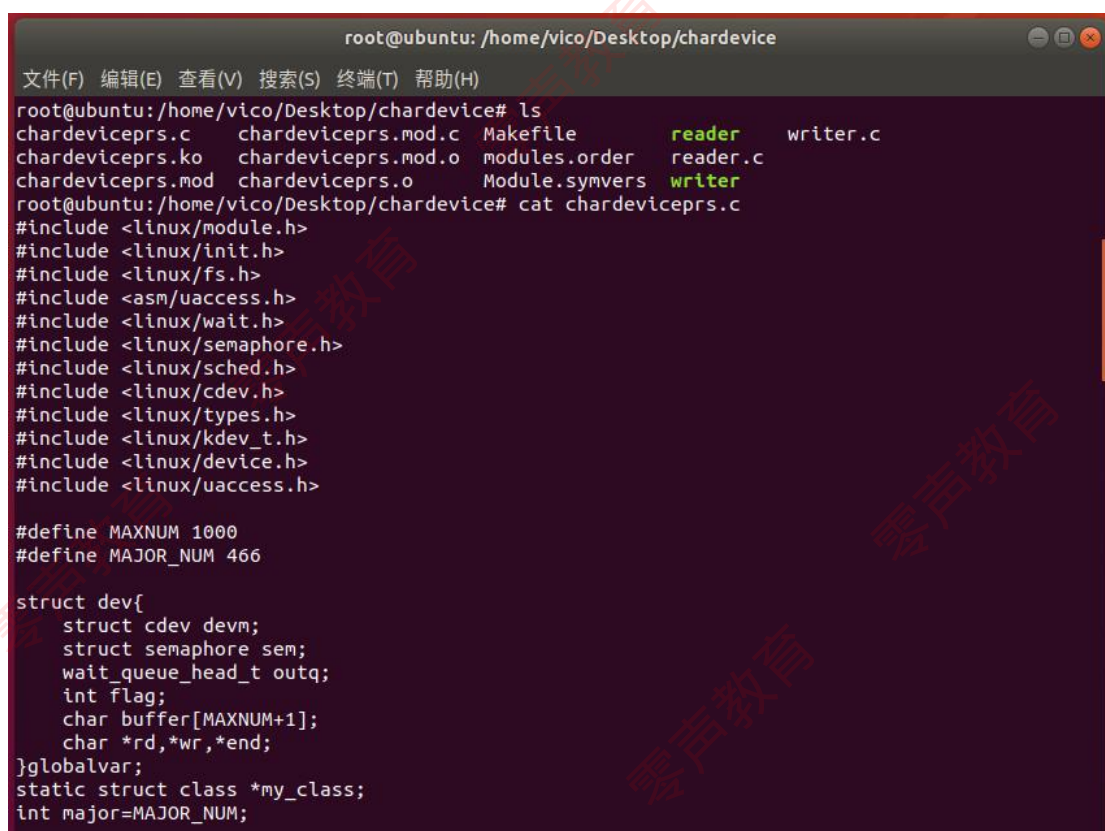
```
include > linux > C fs.h > file_operations
1781
1782 struct file_operations {
1783     struct module *owner;
1784     loff_t (*llseek) (struct file *, loff_t, int);
1785     ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
1786     ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
1787     ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
1788     ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
1789     int (*iterate) (struct file *, struct dir_context *);
1790     int (*iterate_shared) (struct file *, struct dir_context *);
1791     __poll_t (*poll) (struct file *, struct poll_table_struct *);
```

### 3、字符设备驱动程序通信实战

a、编码实现字符设备驱动程序，通过对字符设备的同步操作完成应用程序之间的数据通信。



b、字符驱动设备源码





### c、数据发送端源码

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

root@ubuntu:/home/vico/Desktop/chardevice# ls
chardeviceprs.c  chardeviceprs.mod.c  Makefile  reader  writer.c
chardeviceprs.ko  chardeviceprs.mod.o  modules.order  reader.c
chardeviceprs.mod  chardeviceprs.o  Module.symvers  writer
root@ubuntu:/home/vico/Desktop/chardevice# cat writer.c
#include<sys/types.h>
#include<unistd.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
#include<string.h>

int main(int argc,char* argv[])
{
    int iFd;
    char SendMsg[200];
    iFd= open("/dev/chardev0",O_RDWR,S_IRUSR|S_IWUSR);
    if(-1!=iFd)
    {
        while(1)
        {
            printf("Please enter the transmission message:");
            scanf("%s",SendMsg);
        }
    }
}
```

### d、数据读取端源码

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

root@ubuntu:/home/vico/Desktop/chardevice# ls
chardeviceprs.c  chardeviceprs.mod.c  Makefile  reader  writer.c
chardeviceprs.ko  chardeviceprs.mod.o  modules.order  reader.c
chardeviceprs.mod  chardeviceprs.o  Module.symvers  writer
root@ubuntu:/home/vico/Desktop/chardevice# cat reader.c
#include<sys/types.h>
#include<unistd.h>
#include<sys/stat.h>
#include<stdio.h>
#include<fcntl.h>
#include<string.h>

int main(int argc,char* argv[])
{
    int iFd=0,i=0;
    char ReceieMsg[200];
    iFd= open("/dev/chardev0",O_RDWR,S_IRUSR|S_IWUSR);
    if(-1!=iFd)
    {
    }
```

### e、Makefile 文件设计

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

root@ubuntu:/home/vico/Desktop/chardevice# ls
chardeviceprs.c  chardeviceprs.mod.c  Makefile  reader  writer.c
chardeviceprs.ko  chardeviceprs.mod.o  modules.order  reader.c
chardeviceprs.mod  chardeviceprs.o  Module.symvers  writer
root@ubuntu:/home/vico/Desktop/chardevice# cat Makefile
obj-m:=chardeviceprs.o

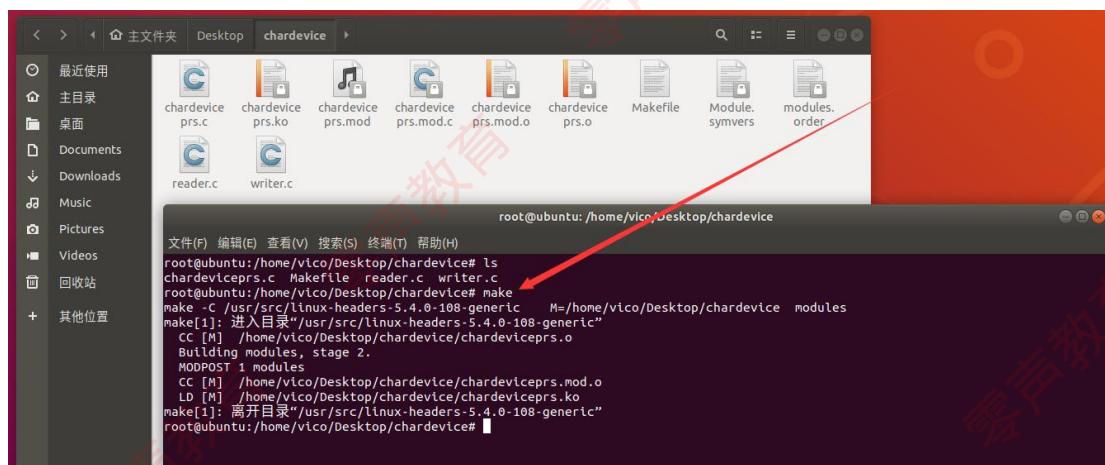
CURRENT_PAHT:=$(shell pwd)
LINUX_KERNEL:=$(shell uname -r)

LINUX_KERNEL_PATH:=/usr/src/linux-headers-$(LINUX_KERNEL)
all:
    make -C $(LINUX_KERNEL_PATH) M=$(CURRENT_PAHT) modules

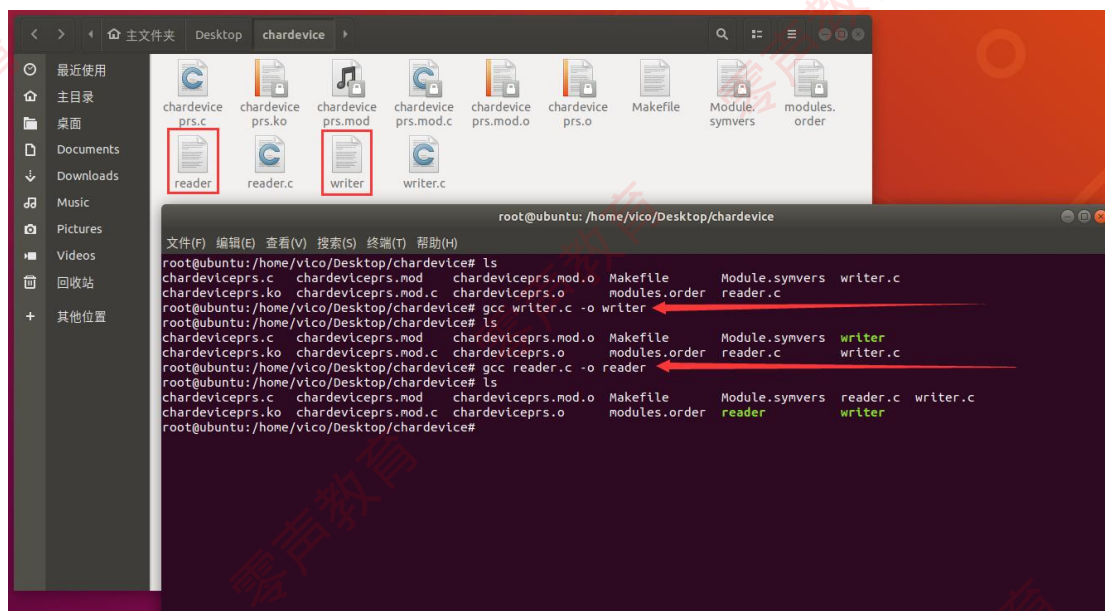
clean:
    make -C $(LINUX_KERNEL_PATH) M=$(CURRENT_PAHT) cleals

root@ubuntu:/home/vico/Desktop/chardevice#
```

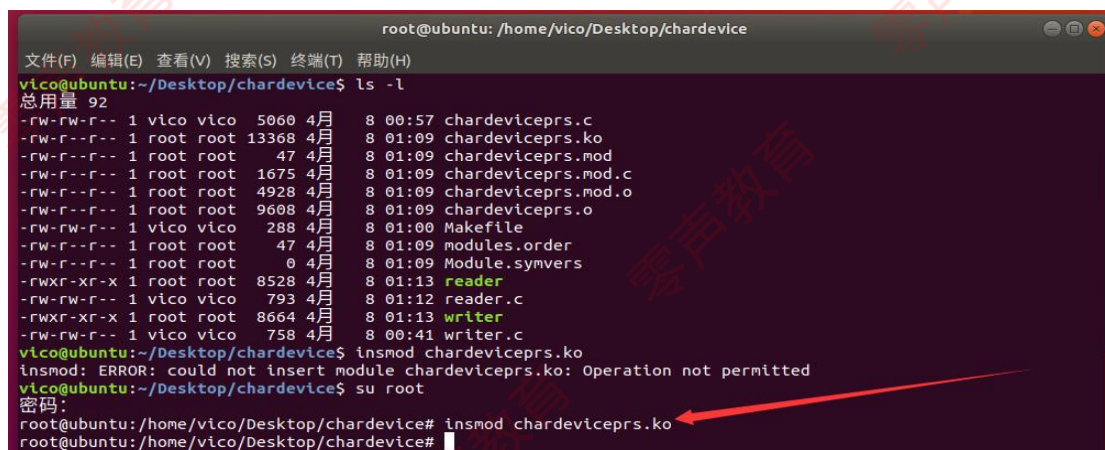
## f、程序执行效果



## g、【编译】【数据发送端和数据读取端程序】



## h、【插入字符设备驱动模块】





i、执行【数据发送端与数据读取端】

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@ubuntu:/home/vico/Desktop/chardevice# ls -l
总用量 92
-rw-rw-r-- 1 vico vico 5060 4月 8 00:57 chardeviceprs.c
-rw-r--r-- 1 root root 13368 4月 8 01:09 chardeviceprs.ko
-rw-r--r-- 1 root root 47 4月 8 01:09 chardeviceprs.mod
-rw-r--r-- 1 root root 1675 4月 8 01:09 chardeviceprs.mod.c
-rw-r--r-- 1 root root 4928 4月 8 01:09 chardeviceprs.mod.o
-rw-r--r-- 1 root root 9608 4月 8 01:09 chardeviceprs.o
-rw-rw-r-- 1 vico vico 288 4月 8 01:00 Makefile
-rw-r--r-- 1 root root 47 4月 8 01:09 modules.order
-rw-r--r-- 1 root root 0 4月 8 01:09 Module.symvers
-rwxr-xr-x 1 root root 8528 4月 8 01:13 reader
-rw-rw-r-- 1 vico vico 793 4月 8 01:12 reader.c
-rwxr-xr-x 1 root root 8664 4月 8 01:13 writer
-rw-rw-r-- 1 vico vico 758 4月 8 00:41 writer.c
root@ubuntu:/home/vico/Desktop/chardevice# insmod chardeviceprs.ko
root@ubuntu:/home/vico/Desktop/chardevice# ./writer
Please enter the transmission message:helloworld
Please enter the transmission message:howdoyoudo
Please enter the transmission message:goodbye
Please enter the transmission message:exit
root@ubuntu:/home/vico/Desktop/chardevice# ./reader
Read data information: helloworldhowdoyoudogoodbyeexit
```

j、通过 dmesg 查看消息

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@ubuntu:/home/vico/Desktop/chardevice# dmesg
```

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
[ 17.074743] Bluetooth: RFCOMM TTY layer initialized
[ 17.074750] Bluetooth: RFCOMM socket layer initialized
[ 17.074767] Bluetooth: RFCOMM ver 1.11
[ 18.231033] rfkill: input handler disabled
[ 46.615158] perf: interrupt took too long (2629 > 2500), lowering kernel.perf_event_max_sample_rate to 76000
[ 85.988715] perf: interrupt took too long (3322 > 3286), lowering kernel.perf_event_max_sample_rate to 60000
[ 729.005520] perf: interrupt took too long (4285 > 4152), lowering kernel.perf_event_max_sample_rate to 46500
[ 832.352478] chardeviceprs: loading out-of-tree module taints kernel.
[ 832.352520] chardeviceprs: module verification failed: signature and/or required key missing - tainting kernel
[ 832.352928] Globalvar register success.
[ 838.430249] This chrdev is in open.
[ 843.620278] the write len is 9
[ 843.620282] the write buffer is helloworld
[ 843.620284] the len of buffer is 9
[ 854.526881] the write len is 10
[ 854.526884] the write buffer is helloworldhowdoyoudo
[ 854.526885] the len of buffer is 19
[ 893.587623] the write len is 7
[ 893.587625] the write buffer is helloworldhowdoyoudogoodbye
[ 893.587626] the len of buffer is 26
[ 902.564702] the write len is 4
[ 902.564706] the write buffer is helloworldhowdoyoudogoodbyeexit
[ 902.564708] the len of buffer is 30
[ 902.564717] This chrdev is in release.
[ 909.802002] This chrdev is in open.
[ 909.802005] Into the read function
[ 909.802006] The rd is h
[ 909.802007] The len is 30
[ 909.802008] The read buffer is helloworldhowdoyoudogoodbyeexit
[ 1345.862302] perf: interrupt took too long (5401 > 5356), lowering kernel.perf_event_max_sample_rate to 37000
root@ubuntu:/home/vico/Desktop/chardevice#
```

### k、删除字符设备驱动模块

```
root@ubuntu: /home/vico/Desktop/chardevice
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
root@ubuntu: /home/vico/Desktop/chardevice# rm /dev/chardev0
root@ubuntu: /home/vico/Desktop/chardevice# dmesg
```

## 三、设备驱动数据结构及 I/O 调度

### 1、device 表示元数据（代表一个物理设备）

```
include > linux > C device.h > ...
971 struct device {
972     struct device *parent;
973
974     struct device_private *p;
975
976     struct kobject kobj;
977     const char *init_name; /* initial name of the device */
978     const struct device_type *type;
979 }
```

### 2、字符设备

```
include > linux > C cdev.h > ops
15 struct cdev {
16     struct kobject kobj;
17     struct module *owner;
18     const struct file_operations *ops;
19     struct list_head list;
20     dev_t dev;
21     unsigned int count;
22 } __randomize_layout;
23
```

### 3、块设备

```
include > linux > C genhd.h > gendisk
174
175 struct gendisk {
176     /* major, first_minor and minors are input parameters only,
177      * don't use directly. Use disk_devt() and disk_max_parts().
178      */
179     int major; /* major number of driver */
180     int first_minor;
181     int minors; /* maximum number of minors, =1 for
182                  * disks that can't be partitioned.
183      */
184     char disk_name[DISK_NAME_LEN]; /* name of major driver */
185     char *(*devnode)(struct gendisk *gd, umode_t *mode);
186
187     unsigned int events; /* supported events */
188 }
```

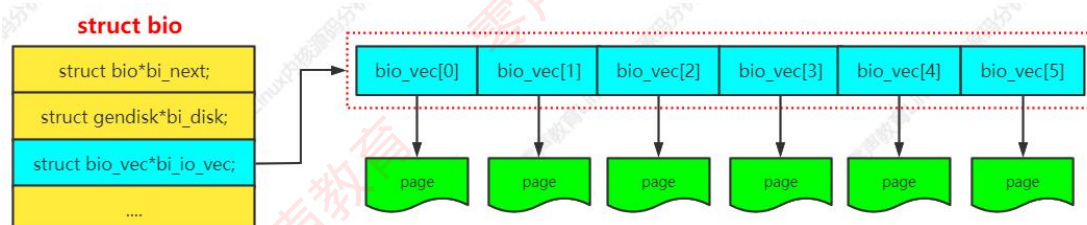


3、bio 结构体包含大量的基础信息，这些都是一个基本单元的属性，他们代表着当前这个 bio 的状态，比如是读还是写或者是一些特殊的操作命令等等。一个 bio 所请求的数据块在设备当中是连续的，bio 所携带的数据大小有上限。

bio 数据结构分析如下：

```
include > linux > blk_types.h > bio
139
140 /*
141  * main unit of I/O for the block layer and lower layers (ie drivers and
142  * stacking drivers)
143  */
144 struct bio {
145     struct bio      *bi_next; /* request queue link */
146     struct gendisk  *bi_disk;
147     unsigned int    bi_opf; /* bottom bits req flags,
148                             * top bits REQ_OP. Use
149                             * accessors.
150                             */
151     unsigned short  bi_flags; /* status, etc and bvec pool number */
152     unsigned short  bi_ioprio;
153     unsigned short  bi_write_hint;
154     blk_status_t    bi_status;
155     u8              bi_partno;
156
157     /* Number of segments in this BIO after
158      * physical address coalescing is performed.
159      */
160     unsigned int    bi_phys_segments;
```

bio/bi\_io\_vec/page 之间的联系如下：



#### 4、I/O 调度操作

Linux 内核采用的各种用于调度和重排 I/O 操作请求以获得最优的性能的算法，称为 I/O 调度器。在将请求提交给块设备时提供各种调度策略，具体通过 elevator\_type 来管理及实现 I/O 调度器。具体源码如下：

```
include > linux > elevator.h > elevator_type
62
63 /*
64  * identifies an elevator type, such as AS or deadline
65  */
66 struct elevator_type
67 {
68     /* managed by elevator core */
69     struct kmem_cache *icq_cache;
70
71     /* fields provided by elevator implementation */
72     struct elevator_mq_ops ops;
73
74     size_t icq_size; /* see iocontext.h */
75     size_t icq_align; /* ditto */
```