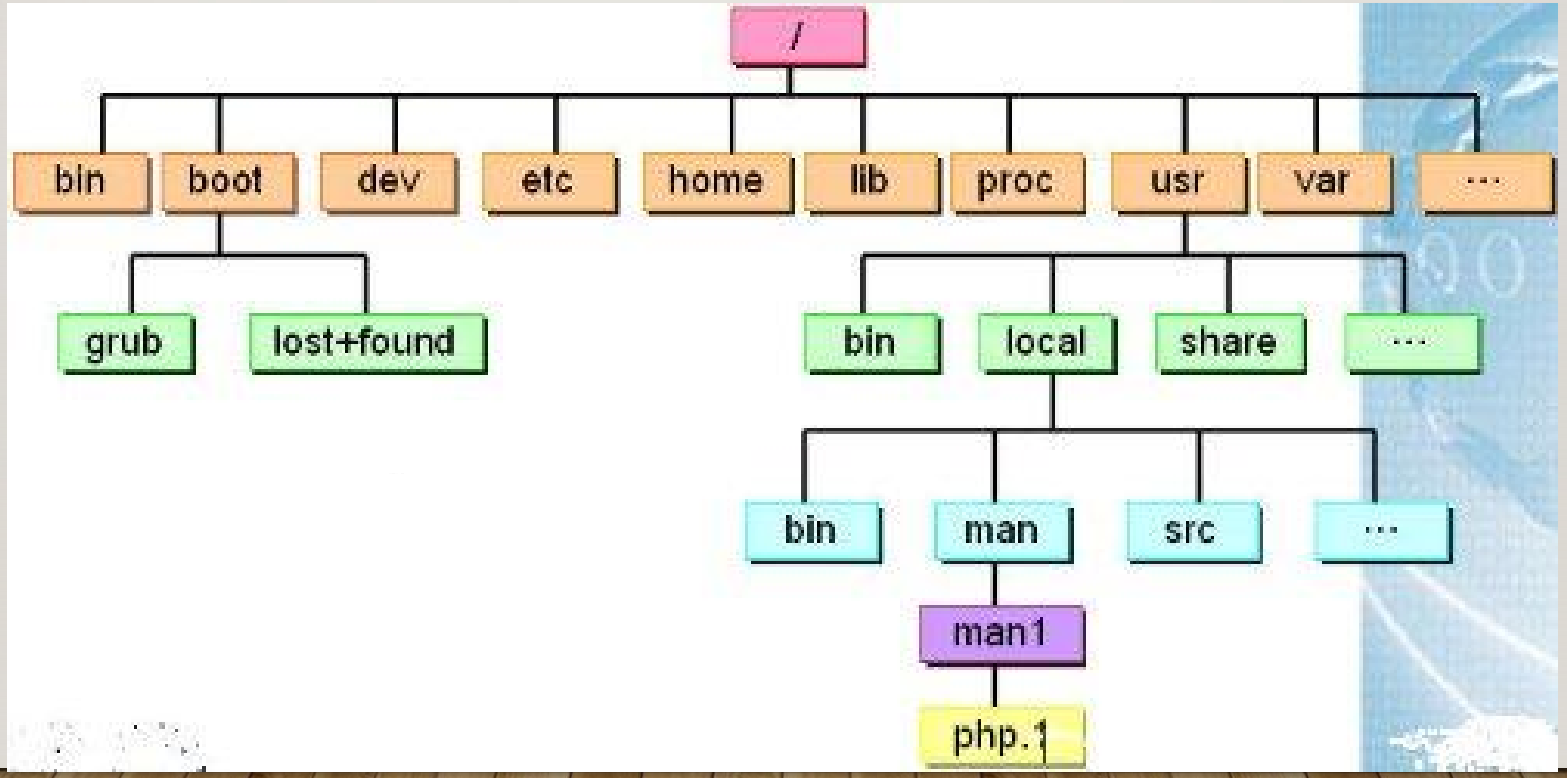
第十章：LINUX文件系统

本章主要内容

1. Linux文件系统特点
2. Linux虚拟文件系统
3. 文件系统注册与挂装
4. 进程与文件系统的联系
5. ext2文件系统
6. 块设备与字符设备

10.1 LINUX文件系统特点

- Linux文件系统：在Linux文件系统中，所有的文件被组织到一个统一的树型目录结构中。



10.1 LINUX文件系统特点

- Linux文件系统特点
  - 文件是无结构的字符流式文件。
  - 文件可以动态地增长或减少。
  - 文件数据可由文件拥有者设置相应的访问权限而受到保护。
  - 外部设备，如磁盘设备、键盘、鼠标等都被看作是文件。

10.1 LINUX文件系统特点

- 文件类型
  - 普通文件：用户和系统的有关的数据和程序文件。无结构、无记录概念的字符流式文件。
  - 目录文件：文件系统中目录所形成的文件。无结构、无记录概念的字符流式文件。
  - 设备文件
  - 有名管道
  - 软链接
  - 网络套接字

10.2 LINUX的虚拟文件系统

- 虚拟文件系统VFS
  - 由于存在不同的文件系统，Linux系统为了解决这个问题，采用了虚拟文件系统。
  - 虚拟文件系统是由面向对象的思想发展起来的。VFS提供了一个抽象的基类，由这个基类派生出的子类支持具体的文件系统
  - Linux虚拟文件系统支持
    - 基于磁盘的文件系统，如ext、fat32、ntfs等
    - 基于网络的文件系统，如NFS 、SMB等
    - 特殊的文件系统，如proc等

10.2 LINUX的虚拟文件系统

- Linux虚拟文件系统的数据结构
  - C语言中没有类，所有VFS是通过数据结构实现的。该模型主要的对象：
    - 超级块super-block：存放已挂载文件系统的信息。
    - 索引节点inode：存放关于一个具体文件的一般信息。
    - 文件file：存放打开文件与进程之间交互的信息。
    - 目录项dentry：保存目录项与相应文件进行链接的信息。

10.2 LINUX的虚拟文件系统

VFS的超级块super_blok

该数据结构位于include/linux/fs.h

内核为每一个已挂载文件系统分配一个超级块

```
struct super_block{
    struct list_head  s_list;  //所有超级块对象构成双向循环链表
    struct file_system_type *s_type;  //文件系统类型
    struct super_operations *s_op; //与超级块关联的超级块操作
    struct dentry *s_root;  //文件系统挂装目录的目录项结构
    void  *s_fs_info; //指向特定文件系统超级块信息数据结构的指针
    ....
}
```

10.2 LINUX的虚拟文件系统

索引节点inode

文件系统处理文件所需要的所有信息都存放在索引节点数据结构中。

该数据结构位于include/linux/fs.h

```
struct inode{
    struct hlist_node i_hash;  //散列表，加速内核对索引节点对象的搜索
    struct list_head i_dentry; //指向目录项链表的指针
    unsigned long i_ino; //唯一的索引节点好
    struct inode_operations *i_op; //指向对索引节点的操作
    ……
}
```

10.2 LINUX的虚拟文件系统

文件file

文件对象描述是进程和一个打开文件交互的过程。文件对象是在文件被打开时创建的。

该结构位于include/linux/fs.h

```
struct file{
    struct dentry *f_dentry; //指向与文件相关的目录项结构的指针
    struct vfsmount *f_vfsmnt; //文件所在的文件系统信息
    struct file_operations *f_op; //指向文件操作表的指针
    loff_t f_pos; //文件的偏移，在文件操作中表示文件当前位置的指针
    unsigned int f_uid, f_gid; //打开该文件进程所属的用户
    ……
    }
```

10.2 LINUX的虚拟文件系统

目录项dentry

目录也是一类文件，当目录被读入内存，VFS把它转换为基于dentry结构的一个目录项。目录项对象将每个目录与其对应的索节点向相联系。

该结构位于include/linux/dcache.h

```
struct dentry{
    struct inode * d_inode; //与文件名关联的索引节点
    struct dentry * d_parent; //父目录的目录项对象
    struct qstr d_name; //文件名
    struct dentry_operations *d_op; //目录项方法
    struct super_block *d_sb; //文件的超级块对象
    ……
        }
```

10.2 LINUX的虚拟文件系统

- VFS的系统调用
  - VFS是位于应用程序和具体文件系统之间的一层，它实现了与文件系统相关的所有系统调用，为各种文件系统提供了一个通用的接口。应用程序不需要关心文件系统的实现细节，只需要与VFS进行交互。

10.2 LINUX的虚拟文件系统

mount( ) umount( )	挂装/卸载文件系统
sysfs( )	获取文件系统信息
statfs( ) fstatfs( ) ustat( )	获取文件系统统计信息
chroot( )	更改根目录
chdir( ) fchdir( ) getcwd( )	操纵当前目录
mkdir( ) rmdir( )	创建/删除目录
stat( ) fstat( ) lstat( ) access( )	读取文件状态
open( ) close( ) creat( ) umask( )	打开/关闭文件
dup( ) dup2( ) fcntl( )	对文件描述符进行操作
select( ) poll( )	异步 I/O 通告
read( ) write( ) readv( ) writev( ) sendfile( )	进行文件 I/O 操作
readlink( )symlink( )	对软链接进行操作
chown( ) fchown( ) lchown( )	更改文件所有者
chmod( ) fchmod( ) utime( )	更改文件属性
pipe( )	进行通信操作

10.3 文件系统的注册和挂装

- 文件系统注册
  - 只有当完成对某个文件系统的注册，内核才能使用这个具体文件系统的各种操作函数，将这个文件系统挂装在系统的目录树上。
  - 所谓注册就是把具体的文件系统的操作代码装入内核。
  - 用户可从/proc/filesystems文件读到已经在内核中注册的文件系统。每个文件系统有一个file_system_type数据结构
  - 内核调用register_filesystem()进行注册，通过unregister_filesystem()函数卸载。

10.3 文件系统的注册和挂装

- 已挂载文件系统描述符链表
  - 每个文件系统都有自己的挂载根目录。如果某个文件系统的根目录是文件系统文件树的根目录，那么该文件系统被称为根文件系统。
  - 其他文件系统挂载到系统的目录树上，则挂载的目录树称为挂载点（mount point）。
  - 系统中通过vfsmount类型的数据结构保存已挂载文件系统的信息，包括挂载点和挂载标志，以及其他已挂载文件系统之间关系。

10.3 文件系统的注册和挂装

- 挂装根文件系统
  - 系统从变量ROOT_DEV中查找包含根文件系统的磁盘主设备号，这个变量在编译内核时指定或引导程序向内核传递一个“root”参数。
  - 挂载根文件系统分2个阶段：
    - 安装特殊的rootfs文件系统，该文件系统仅提供一个作为初始安装点的空目录。
    - 内核在空目录上安装一个真正的根目录。

10.3 文件系统的注册和挂装

- 挂载一般文件系统
  - 挂载命令：mount
    - $ mount –t ext2 /dev/hda5 /mnt/hda5
      - 其中-t指出挂载系统类型，/dev/hda5为文件所在磁盘分区，/mnt/hda5为文件的挂载点。
  - 挂载过程：
    - 搜索文件类型注册表file_systems，从中找出相对应的节点。
    - 内核准备挂载点的inode索引节点，只能在目录下进行挂载。
    - 向super_blocks链表申请一个空闲的元素。
    - 安装文件系统的磁盘超级块，并把内容写入super_block中。
    - 调用add_vfsmnt()函数，申请vfsmount数据结构，填入相应数。据成员后，插入到已挂载文件描述符链表vfsmntlist的链尾。

10.3 文件系统的注册和挂装

- 卸载文件系统
  - 卸载命令unmount
    - unmount /mnt/hda5
  - 卸载过程
    - 先察看挂载点是否在使用中
    - 将对应的VFS的super_block超级块释放
    - 并把代表的vfsmount结构从文件系统链表vfsmntlist中删除。

10.4 进程与文件系统的联系

系统打开文件表

在访问文件之前，进程必须先打开文件，既通过系统调用open()得到一个文件描述符。通过这个描述符进程才能与相应的文件或物理设备相关联。

文件描述符中含有指向一个系统打开文件对象（struct file）的指针和其他参数。

由strcut file结构组成的链表叫做系统打开文件表。

10.4 进程与文件系统的联系

用户打开文件表

在进程描述符中，files域是进程打开文件表，用来记录和控制进程当前打开的文件。用来记录和控制进程当前打开的文件

```
struct files_struct{
    atomic_t count;  // 共享该表的进程数目
    spinlock_t file_lock; //用于表中字段的读写自旋锁
    int max_fds;  //文件对象的当前最大数目
    struct file ** fd; //指向文件对象指针数组的指针
    … …
}
```

10.4 进程与文件系统的联系

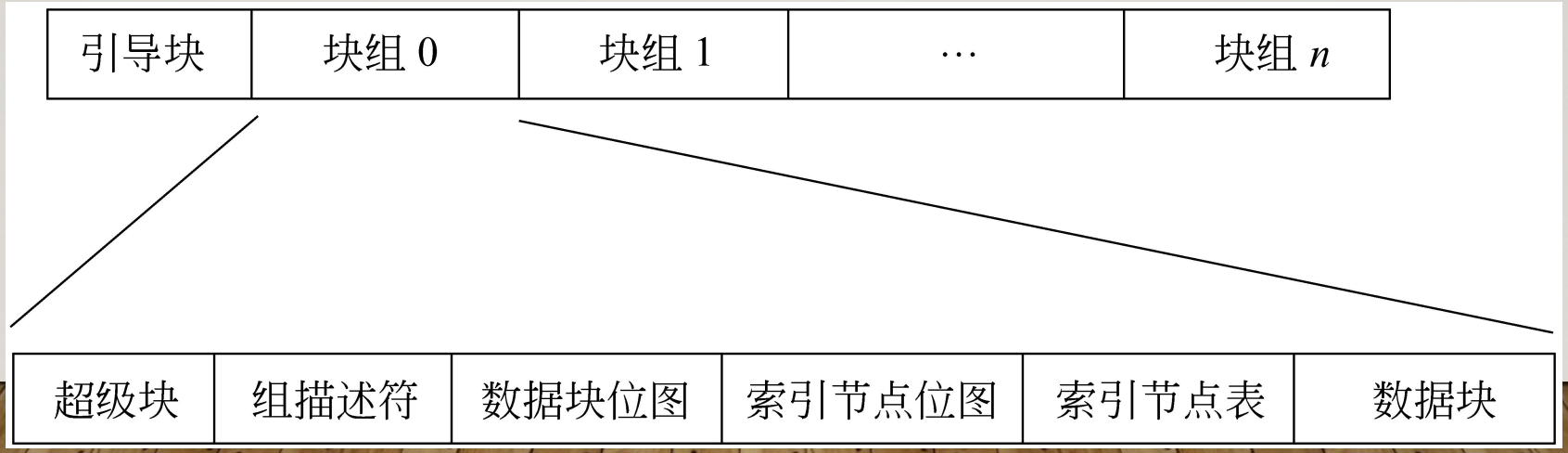
- 进程的当前目录和根目录
  - 在Linux系统中，每一个进程都有一个当前工作目录和它自己的根目录。这是进程状态的一部分。以便用户既可以使用相对路径名也可以使用绝对路径名来访问所需要文件。
  - 在进程描述符中的fs域就是用来维护这两个数据。这是一个指向struct fs_struct数据结构的指针。

10.5 EXT2 文件系统

ext2文件系统的存储结构

在ext2文件系统中，逻辑块的大小可以是1024,2048或者4096。

ext2文件系统将一个逻辑区分为若干个块组，每个块组都包含该文件系统的综合信息，即超级块和组描述符表。



10.5 EXT2 文件系统

- ext2文件系统主要磁盘数据结构
  - ext2超级块的数据结构是struct ext2_super_block
  - 该结构中：
    - s_inode存放索引节点的个数
    - s_blocks_count存放ext2文件系统的总块数
    - s_log_block_size用2的幂次方表示块的大小

    … …
  - 该结构定义于fs/ext2/ext2_fs.h

10.5 EXT2 文件系统

ext2的块组描述符

每个块组有自己的组描述符，用来描述每个块组的使用情况。

```
struct ext2 _group_desc
{
        __le32   bg_block_bitmap;          /* 块位图的块号 */
        __le32   bg_inode_bitmap;          /* 索引节点位图的块号 */
        __le32   bg_inode_table;           /* 第一个索引节点表块的块号 *
        __le16   bg_free_blocks_count;     /* 组中空闲块的个数 */
        __le16   bg_free_inodes_count;     /* 组中索引节点的个数 */
        __le16   bg_used_dirs_count;       /* 组中目录的个数 */
        __le16   bg_pad;                   /* 按字对齐 */
        __le32   bg_reserved[3];           /* 用 null 填充 */
};
```

10.5 EXT2 文件系统

- 块位图和索引节点位图
  - 每个块中使用了两个块来记录本组内各个数据块的使用情况和索引节点表的使用情况。
    - 数据块位图：每一位代表一个数据块。
    - 索引节点inode位图：纪录本组内索引节点表的使用情况。
  - 每个位图必须存放在一个单独的块中。

10.5 EXT2 文件系统

ext2文件系统的磁盘索引节点

ext2文件系统中的每个文件对应一个描述它的磁盘数据结构，即磁盘索引节点，inode节点。

每个块中的索引节点总数存放在磁盘超级块的成员s_inodes_per_group中。数据结构主要有：

```
strcut ext2_inode{
    __le16  i_mode; //文件类型及访问权限
    __le32  i_size; //以字节为单位的文件长度
    __le16 i_uid; //文件属主的uid
    __le16 i_links_count; //链接数
    __le32 i_blocks; //文件的数据块数
    __le32 i_block[EXT2_N_BLOCKS]; //数据块指针
    … …
}
```

10.5 EXT2 文件系统

- ext2文件系统的内存数据结构
  - ext2文件系统的磁盘组织和内核组织并不完全一样。磁盘上存储的数据结构主要考虑如何节省磁盘空间，而内存中的数据结构主要考虑系统运行的效率和性能。
  - 磁盘数据结构ext2_super_block和ext2_inode分别对应内存数据结构ext2_sb_info和ext2_inode_info
- ext2的内存超级块
  - 安装ext2文件系统时，首先从磁盘上读取磁盘超级块ext2_super_block结构体内容，填充到ext2_sb_info的内存超级块，并将后者的指针写入VFS超级块的s_fs_info域。

10.5 EXT2 文件系统

- 内存超级块ext2_sb_info包含：
  - 磁盘超级块的大部分内容
  - 块位图高速缓存
  - 索引节点位图高速缓存
  - s_sbh指针，指向磁盘超级块所在缓冲区的缓冲区首部
  - s_cs指针，指向磁盘超级块所在的缓冲区
  - 组描述符的个数
  - 其他与安装状态、安装选项等相关的数据

10.5 EXT2 文件系统

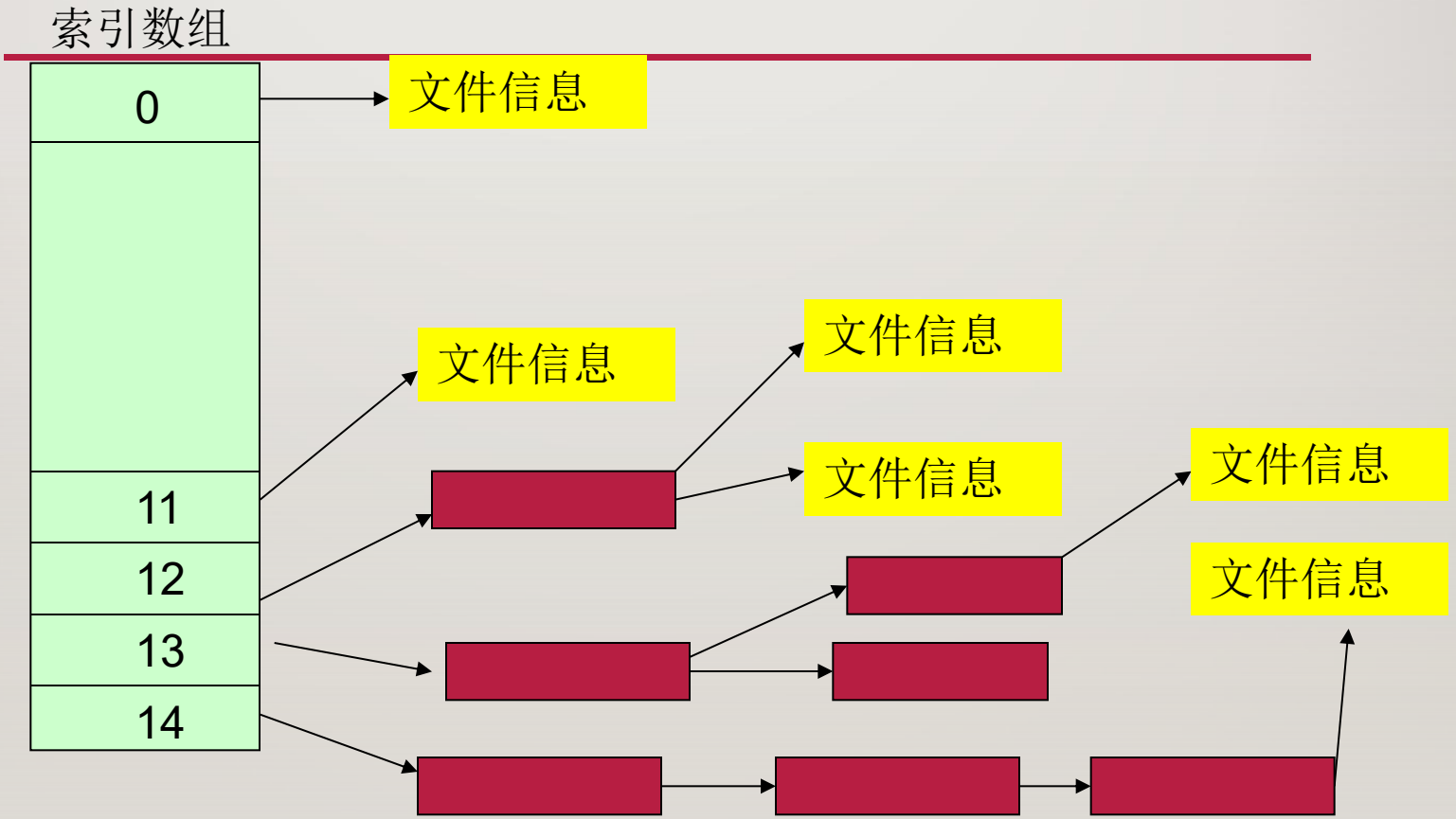
- ext2的内存索引节点
  - 当VFS处理ext2文件的索引节点时，从磁盘读取ext2_inode结构，填充到索引节点描述符ext2_inode_info数据结构中。主要包含：
    - 整个VFS索引节点
    - 片的大小和片数（未使用）
    - 索引节点所在块组的i_block_group块组索引
    - 为数据块进行预分配使用时的字段
    - i_osync，表示是否同步更新磁盘索引节点

10.5 EXT2 文件系统

- 数据块寻址
  - ext2采用了多重索引结构进行寻址。通过数组i_block实现文件块到磁盘逻辑块的转换。数组大小由EXT2_N_BLOCKS决定（默认为15）。数组中15个元素有4种类型
    - 下标0~11： 逻辑块号与文件最初的12个块对应
    - 下标12：一级间接索引
    - 下标13：二级间接索引
    - 下标14：三级间接索引

10.5 EXT2 文件系统

- 数据块寻址



10.6 块设备驱动

Linux系统中的设备可分为两类：块设备与字符设备。

管理这些设备的程序模块称为I/O子系统。I/O子系统完成进程与外设之间的通信任务。其中I/O子系统的核心部分是控制外设的设备驱动程序。

10.6 块设备驱动

- 设备配置
  - linux系统将设备也看作文件。
  - 在/dev目录下，一般存放常用的设备文件
  - 如果一个新设备和系统的连接，需要建立新的设备文件，要通过管理员使用相应的命令，如mknod来建立特殊的文件
    - mknod /dev/tty1 c 4 1
    - mknod要提供文件名，文件类型（字符或块设备），主设备号，次设备号

10.6 块设备驱动

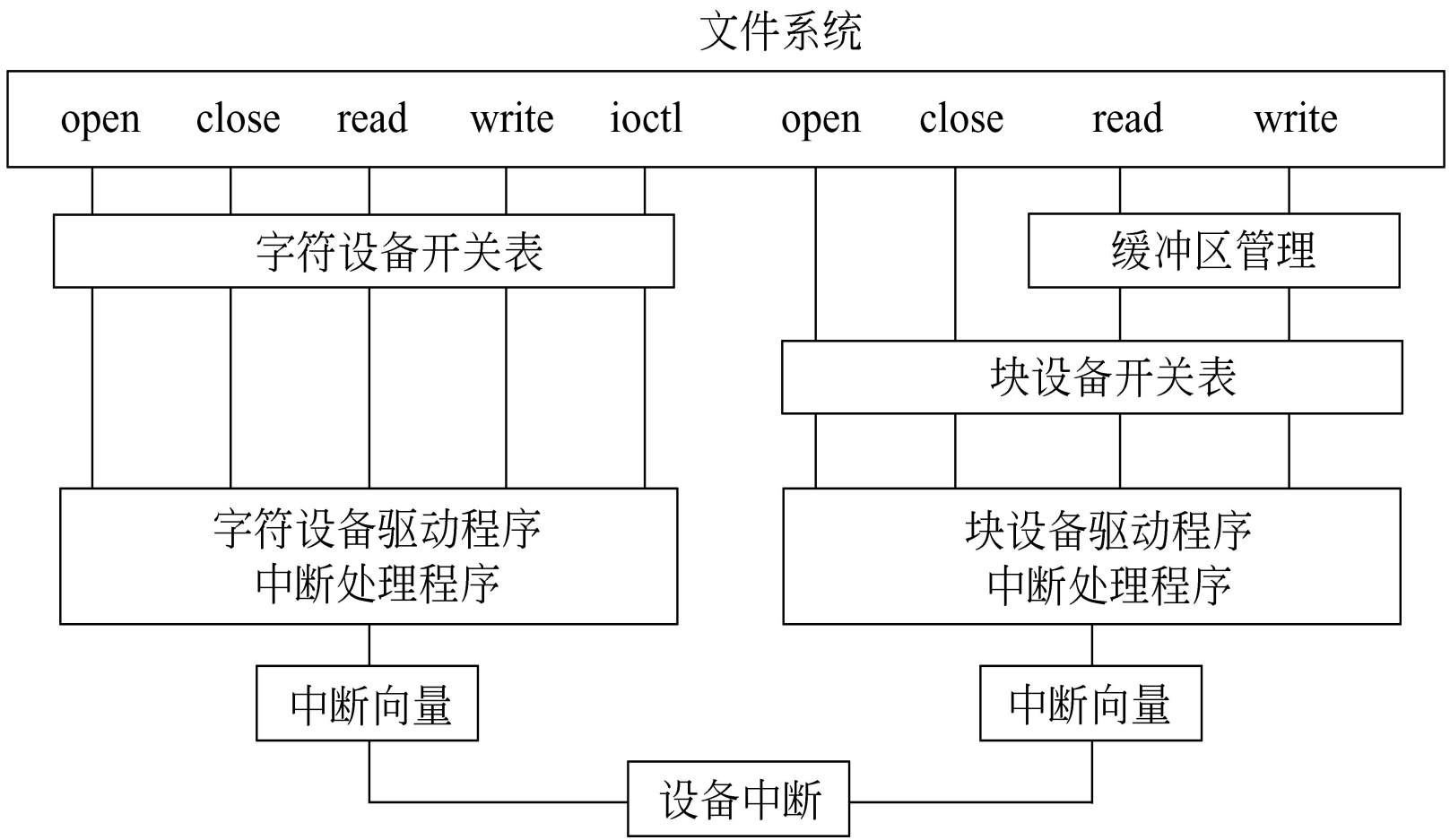
设备驱动程序的接口

用于块设备文件的缺省文件操作：

表 10-2　用于块设备文件的函数与缺省文件的操作方法对应表

方　法	用于块设备文件的函数	方　法	用于块设备文件的函数
open	blkdev_open( )	write	generic_file_write( )
release	blkdev_close( )	mmap	generic_file_mmap( )
llseek	block_llseek( )	fsync	block_fsync( )
read	generic_file_read( )	ioctl	blkdev_ioctl( )

10.6 块设备驱动


文件系统
open
close
read
write
ioctl
open
close
read
write
字符设备开关表
缓冲区管理
块设备开关表
字符设备驱动程序
中断处理程序
块设备驱动程序
中断处理程序
中断向量
中断向量
设备中断

10.7 字符设备驱动

- 字符设备：在I/O传输过程中以字符为单位进行传输的设备，如键盘，打印机等。
- 用于字符设备文件操作只有一个，即打开文件操作chrdev_open。
- 一但字符设备文件被打开，可以通过ioctl方法支持ioctl()调用。

The End