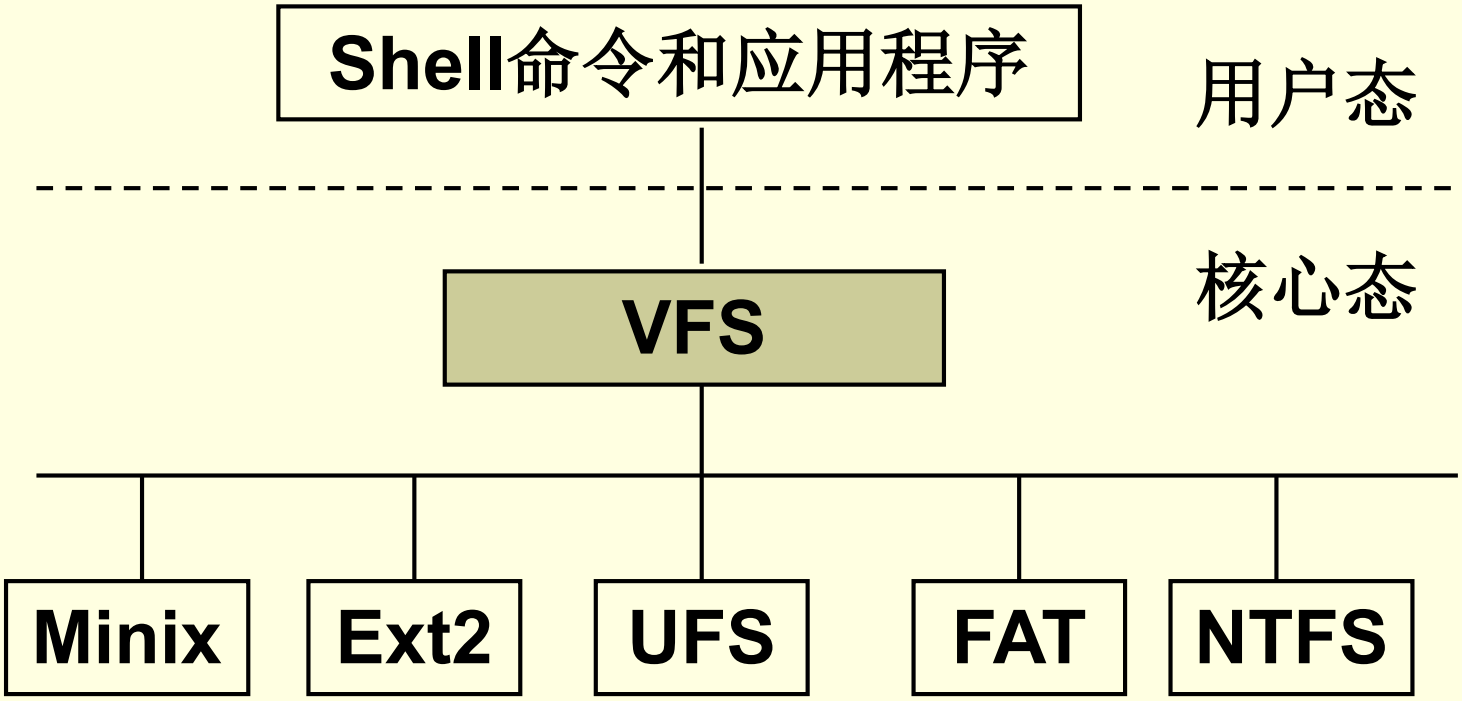
第10章 Linux 虚拟文件系统

北京理工大学计算机学院

VFS与具体文件系统的关系

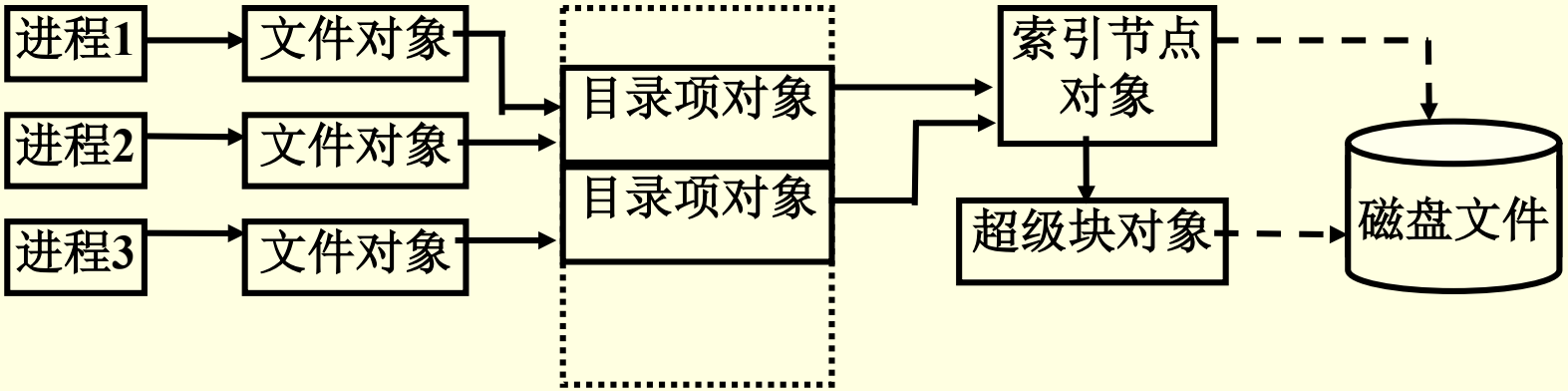


VFS

- VFS的主要思想在于引入一个通用的文件模型，该模型能够表示其支持的所有文件系统。
- VFS所涉及的所有数据结构在系统运行时才在内存建立，在磁盘上没有存储。

10.1 虚拟文件系统的数据结构

- 超级块对象：Linux为每个安装好的文件系统都建立一个超级块对象。
- 索引节点对象：打开的文件对应的…。
- 目录项对象：dentry (directory entry)
- 文件对象：记录了进程与打开的文件之间的交互信息。



10.1.1 超级块对象

```
struct super_block {
    struct list_head s_list; 系统超级块双向链
    struct file_system_type *s_type;
    struct super_operations *s_op;
    struct dentry *s_root 根目录的目录项对象
    struct list_head s_inode;  索引节点链表
    struct list_head s_files;   文件对象链表
    void *s_fs_info; 指向一个具体文件系统的超
    级块结构    ……}
```

- Ext2 在内存的超级块结构ext2_sb_info
- NTFS在内存的超级块结构ntfs_sb_info

10.1.2 索引节点对象

```
struct inode {
   struct list_head i_sb_list; 同一超级块的索引节点链表
   struct inode-operations *i_op;
   unsigned long i_ino;  磁盘索引节点号
   atomic_t i_count;   共享该对象的引用计数
   nlink_t  i_nlink;  硬链接计数
   struct file_operations *i_fop; 默认文件操作
……}
```

内嵌于ext2_inode_info结构。

10.1.3 文件对象

```
struct file {
   struct list_head f_list;    文件对象链表
   struct dentry *f_dentry;  指向目录项对象
   struct file_operation *f_op;
   atomic_t f_count;  共享该对象的进程数
   loff_t  f_pos;   文件的读写指针
   struct address_space *f_mapping; 映射
……}
```

没有对应的磁盘映像。

10.1.4 目录项对象

```
struct dentry {
    atomic_t  d_count;  引用计数
    struct inode *d_inode;  指向文件的inode
    struct dentry *d_parent; 指向父目录项对象
    struct list_head d_alias; 属于同一inode的
    dentry链表
    struct dentry_operations *d_op; 方法
……}
```

没有对应的磁盘映像。

10.1.5 与进程打开文件相关的数据结构

- Struct tast_struct{
  struct fs_struct *fs; 指向文件系统信息
  struct files_struct *files; 指向进程打开文件信息
  …}

fs_struct

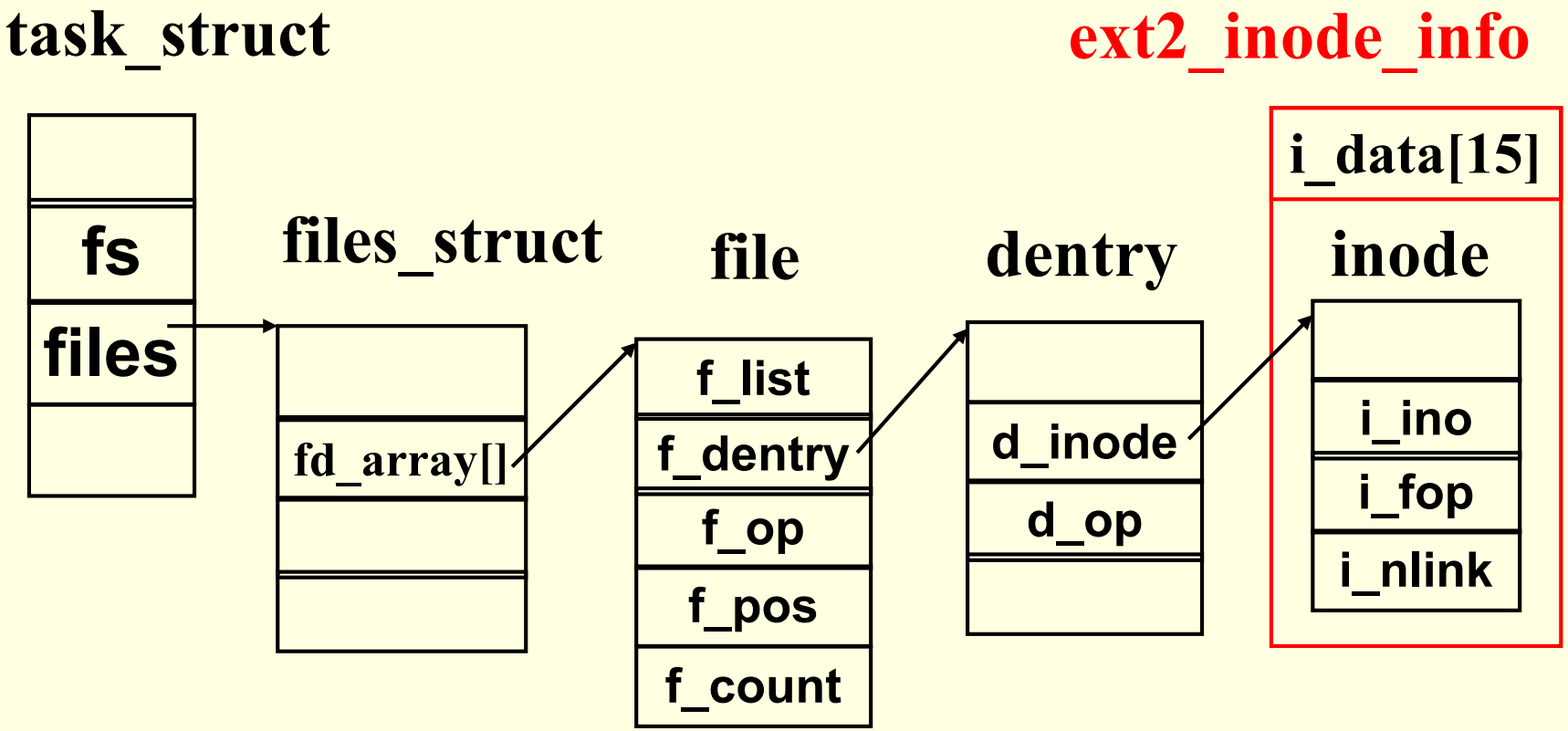
```
struct fs_struct {
    atomic_t count; 共享该结构的进程数
    struct dentry *root, *pwd;每个进程都有自己的当前工作目录和根目录，通过这两个目录与文件系统进行交互。
    struct vfsmount *rootmnt; 根目录下安装的文件系统对象
    struct vfsmount *pwdmnt; 当前目录下安装的文件系统对象
……}
```

files_struct

```
struct files_struct {
    struct file **fd;  指向文件对象指针数组的指针
    struct file *fd_array[ ];文件对象指针数组
    ……}
```

每个进程最多同时打开的文件数为1024个。

进程打开文件的过程



系统打开文件表—链表

10.2 文件系统的注册与安装

- 向Linux内核注册一个文件系统，是在通知内核可以支持的文件系统类型。
- 而要访问一个文件系统，必须将相应的文件系统安装在系统目录树的一个目录下。

10.2.1 文件系统注册

- 首先为被安装的文件系统生成一个类型为**file_system_type**的对象。
- 接着，调用**register_filesystem()**函数完成文件系统的注册。
- 内核将已注册的所有文件系统类型对象链接起来。

10.2.2 文件系统安装

内核将安装点与被安装的文件系统信息保存在vfsmount结构中。形成一个链式安装表。

```
struct vfsmount {
    struct dentry *mnt_mountpoint; 指向安装点的目录项对象
    struct dentry *mnt_root;指向被安装文件系统的根目录
    struct super_block *mnt_sb; 指向被安装文件系统的超级块
……}
```

mount

mount(要安装的文件系统类型，块特别文件路径名，要安装的目录路径名，文件系统的安装标志);

块特别文件路径名是被安装的文件系统所在的块设备文件路径名。

mount –t ntfs /dev/hda2 /mnt/ntfs

10.3 VFS系统调用的实现

- 文件打开与关闭：open(), close()
- 文件的读写：read(), write()