




操作系统


Operating System

北京理工大学计算机学院
马 锐
Email: mary@bit.edu.cn



版权声明

- 本内容版权归北京理工大学计算机学院操作系统课程组马锐所有
- 使用者可以将全部或部分本内容免费用于非商业用途
- 使用者在使用全部或部分本内容时请注明来源
 - 内容来自：北京理工大学计算机学院+马锐+材料名字
- 对于不准守此声明或其他违法使用本内容者，将依法保留追究权



第10章 Linux虚拟文件系统

- 10.1 虚拟文件系统涉及的数据结构
- 10.2 文件系统的注册与安装
- 10.3 VFS系统调用的实现

第10章 Linux虚拟文件系统(1)

- ▶ 借助VFS, Linux可以透明地安装具有其他操作系统的文件系统格式的磁盘或分区
- ▶ VFS与具体文件系统的关系

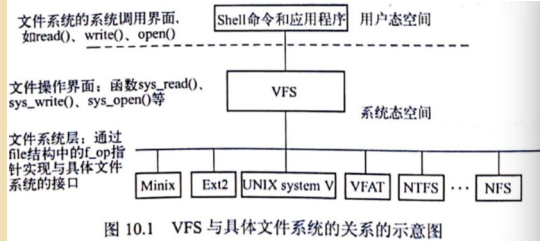
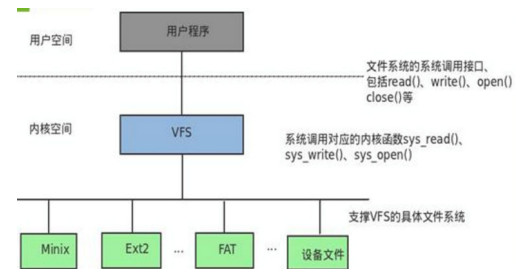


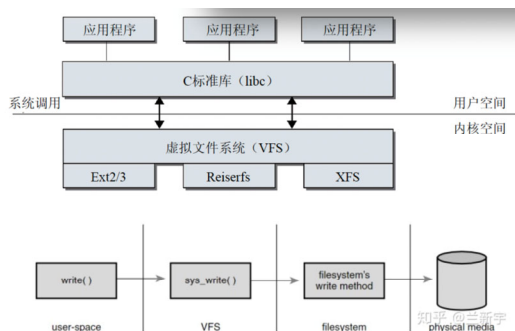
图 10.1 VFS 与具体文件系统的关系的示意图

第10章 Linux虚拟文件系统(2)



5

第10章 Linux虚拟文件系统(3)



6

第10章 Linux虚拟文件系统(4)

➤ VFS

- 主要思想：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
- 作用
 - 对各种文件系统的数据结构进行抽象，以统一的数据结构进行管理
 - 接收用户层的系统调用
 - 支持多种具体文件系统之间的相互访问
 - 接收内核其他子系统的操作请求

7

10.1 虚拟文件系统涉及的数据结构

- 10.1 超级块对象
- 10.2 索引节点对象
- 10.3 目录项对象
- 10.4 文件对象
- 10.5 与进程打开文件相关的数据结构

8

10.1 虚拟文件系统涉及的数据结构(1)

➤ 超级块对象

- 代表一个已安装的文件系统，存放该文件系统的管理和控制信息
- Linux为每个安装好的文件系统都在内存中建立一个超级块对象

➤ 索引节点对象

- 对于具体文件系统，代表一个文件，对应于存放在磁盘上的FCB

➤ 目录项对象

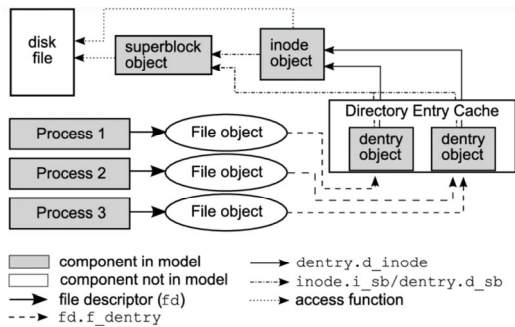
- 对应一个目录项，是文件路径的组成部分，存放目录项与对应文件进行链接的信息

➤ 文件对象

- 记录了进程与打开的文件之间的交互信息

9

10.1 虚拟文件系统涉及的数据结构(2)



10

10.1 超级块对象(1)

- 超级块用来描述整个文件系统的信息
- 每个具体的文件系统都有自己的超级块
- VFS超级块是各种文件系统在安装时建立的，并在卸载时被自动删除，其数据结构是super_block
- 所有超级块对象都以双向循环链表的形式链接在一起
- 与超级块关联的方法是超级块操作表，这些操作由struct super_operations描述

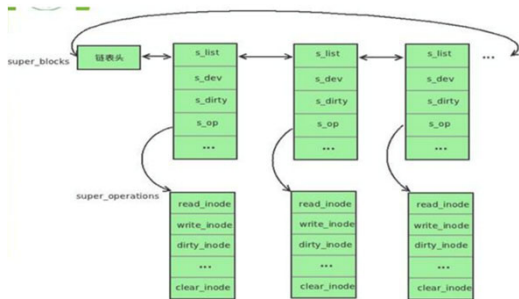
11

10.1 超级块对象(2)

```
struct super_block {
    struct list_head s_list; //系统超级块双向链
    struct file_system_type *s_type; //文件系统类型
    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在内存
                        //的超级块结构X_sb_info
    .....
}
```

12

10.1 超级块对象(3)



13

10.2 索引节点对象(1)

- 文件的管理和控制信息都包含在索引节点inode结构中
- 同一个文件系统中，每个文件的索引节点号都是唯一的
- 与索引节点关联的方法由struct inode_operations来描述

14

10.2 索引节点对象(2)

```
struct inode {
    struct list_head i_sb_list; //同一个超级块的索引
                                //节点链表的指针
    struct list_head i_dentry; //指向引用这个索引节点
                                //的目录项表的指针
    unsigned long i_ino;       //磁盘索引节点号
    atomic_t i_count;          //该对象的引用计数
    nlink_t i_nlink;           //硬链接计数
    loff_t i_size;             //文件的字节长度
    struct timespec i_atime;    //文件的最近访问时间
    struct inode_operations *i_op; //操作索引节点的方法的
                                //结构地址
    struct super_block *i_sb; //指向超级块对象的指针
    struct address_space *i_mapping; //指向地址空间对象的指针
    .....}

```

15

10.3 目录项对象(1)

- 每个文件除了一个inode结构以外，还对应一个目录项dentry结构
- 对于进程查找文件路径名中的每个分量，内核都为其建立一个目录项对象
- 目录项代表的是逻辑意义上的文件，描述的是文件逻辑上的属性，目录项对象在磁盘上并没有对应的映像
- inode代表的是物理意义上的文件，记录的是物理上的属性，对于一个具体的文件系统，其inode在磁盘上有对应的映像

16

10.3 目录项对象(2)

- 目录项对象在磁盘上没有映像，所有目录项对象存放在名为dentry_cache的slab分配器的高速缓冲区中
 - 相当于索引节点高速缓存控制器
 - 一旦目录项对象被使用，可以很快引用相应的索引节点对象
- 与目录项关联的方法由struct dentry_operations描述
- 一个索引节点可能对应多个目录项对象

17

10.3 目录项对象(3)

```
struct dentry {  
    atomic_t d_count;    //目录项对象引用计数  
    struct inode *d_inode; //指向文件的inode对象  
    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; //访问目录项  
                                   //的方法  
    .....  
}
```

18

10.4 文件对象(1)

➤ 文件对象

- 为了描述进程与其打开的一个文件进行交互而引入
- 文件对象在磁盘上没有对应的映像，在文件打开时创建，由file结构描述
- 一个文件对应一个目录项对象
- file结构形成一个双链表，称为系统打开文件表
- 与文件对象关联的方法由struct file_operations来描述

19

10.4 文件对象(2)

```
struct file {  
    struct list_head f_list;    //文件对象链表  
    struct dentry *f_dentry; //文件对应目录项对象指针  
    atomic_t f_count;    //文件对象的引用计数  
    loff_t f_pos;        //文件的当前读写位置  
    struct file_operations *f_op; //访问文件对象的方法  
    struct address_space *f_mapping; //指向该映射文件  
                                //的地址空间对象指针  
    .....  
}
```

20

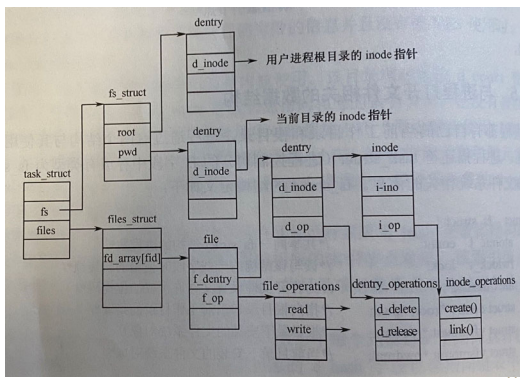
10.5 与进程打开文件相关的数据结构(1)

➤ 进程描述符task_struct

- fs字段存放的fs_struct结构的信息：描述安装的文件系统相关信息
- files字段存放的files_struct结构：描述当前打开文件的信息
- 用户打开文件表，是进程的私有数据

21

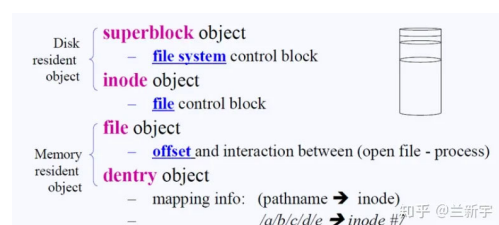
10.5 与进程打开文件相关的数据结构(2)



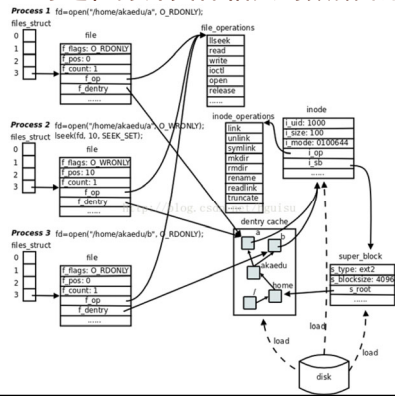
10.5 与进程打开文件相关的数据结构(3)

- 超级块是对一个文件系统的描述
- 索引节点是对一个文件物理属性的描述
- 目录项是对一个文件逻辑属性的描述
- 进程所在的文件系统由fs_struct描述
- 一个进程（或用户）打开的文件由files_struct描述
- 整个系统所打开的文件由file结构描述

10.5 与进程打开文件相关的数据结构(4)



10.5 与进程打开文件相关的数据结构(5)



25

10.5 与进程打开文件相关的数据结构(6)

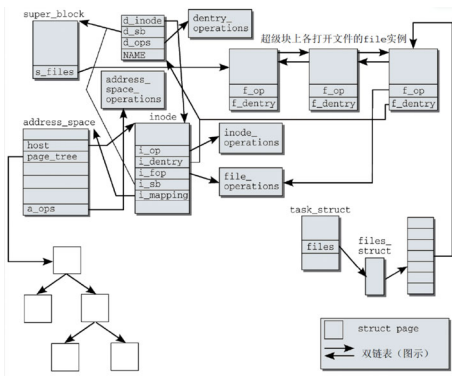
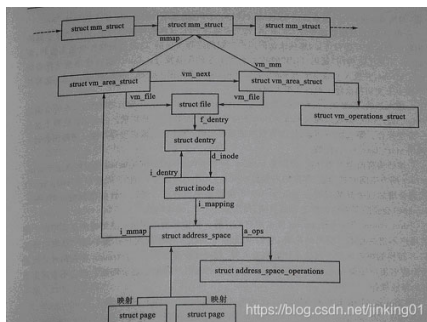


图8-3 各个VFS组件的相互关系

26

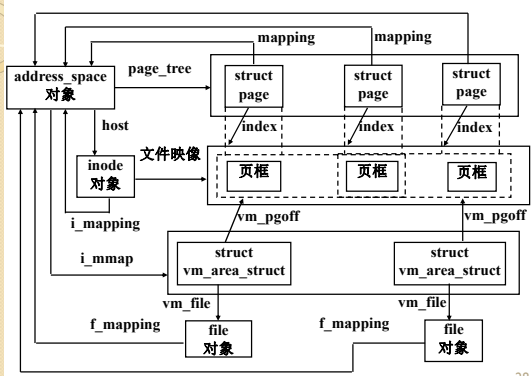
10.5 与进程打开文件相关的数据结构(7)



<https://blog.csdn.net/jinking01>

27

10.5 与进程打开文件相关的数据结构(8)



28

10.2 文件系统的注册与安装

- 10.2.1 文件系统注册
- 10.2.2 文件系统安装

29

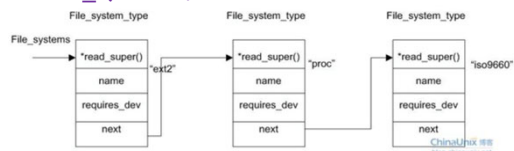
10.2 文件系统的注册与安装

- Linux系统支持的所有文件系统在使用前必须进行注册，完成注册后，VFS为内核提供一个调用相应文件系统的方法的接口
- 之后，再用mount命令将该文件系统安装到根文件系统的某个目录结点上

30

10.2.1 文件系统注册(1)

- 生成一个file_system_type类型的结构，填写相应的内容
- 所有已注册的各种文件系统类型的file_system_type结构形成一个文件系统类型链表，链头指针存放在全局变量file_systems中



10.2.1 文件系统注册(2)

- 调用register_filesystem() 完成注册
- 在配置内核时选择要支持的文件系统，之后其代码被静态链接到内核中。在系统初始化时，调用register_filesystem()，完成文件系统注册
- 采用内核可装载模块方式。当发现一个文件系统模块被装入时，在安装该文件系统模块时，模块初始化函数调用register_filesystem()完成注册，并在模块卸载时完成注销

32

10.2.2 文件系统安装(1)

- 向Linux内核注册一个文件系统只是通知内核可以支持的文件系统类型，若要访问一个文件系统，必须将该文件系统相应的硬盘分区安装在系统目录树的某一个目录（安装点）下
- 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，链头指针在全局变量vfsmnlist
- vfsmount是一次安装的实例，包含了操作该特定文件系统所必须的信息

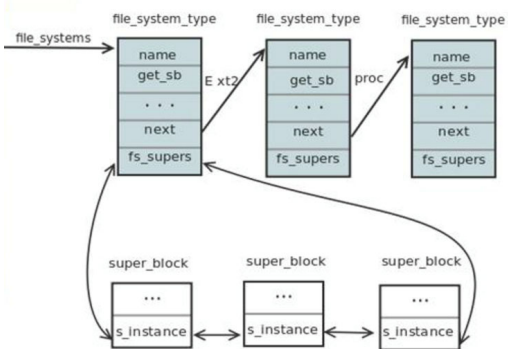
33

10.2.2 文件系统安装 (2)

- 一个超级块对应一个特定文件系统，超级块建立了VFS与特定文件系统的关联
- 同种文件系统类型的super_block挂在fs_supers链表上

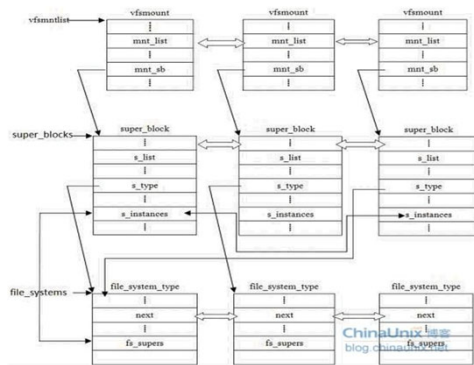
34

10.2.2 文件系统安装 (3)



35

10.2.2 文件系统安装 (4)



36

10.2.2 文件系统安装(5)

➤ 根文件系统

- 最顶层的文件系统叫做“根文件系统”
- Linux 在启动的时候，要求用户必须指定一个“根设备”，内核在初始化阶段，将“根设备”安装到“根安装点”上，从而有了根文件系统。这样，文件系统才算准备就绪

➤ 其他文件系统由用户通过 mount 命令来安装

37

10.2.2 文件系统安装(5)

➤ mount -t ntfs /dev/hda2 /mnt/ntfs

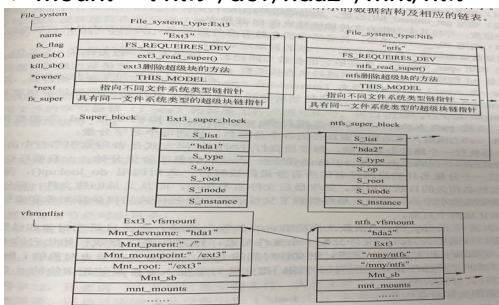


图 10.4 已安装的文件系统生成的各种数据结构之间的关系

38

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()

39