

**<<Linux Ext2文件系统>>**

**实现方案改进**



项目成员

SY1506418 丁贵强

SY1506417 王涵仲

SY1521101 曹卫青

SY1506407 王新晨

北京航空航天大学

2016-4-20

## 1 **项目当前进展**

## 1.1 VFS的文件模型

VFS的关键是，根据不同的文件系统抽象出了一个通用的文件模型。每个特定的文件系统都要把物理操作与通用文件模型对应起来。通用文件模型由4种数据对象组成。

### 1.1.1文件对象

文件对象存储一个打开的文件和一个进程的关联信息。只要文件一直打开，这个对象就一直存在。文件对象用struct file结构描述：

struct file{

struct list\_head f\_list;

struct dentry \* f\_dentry;/\* 指向与文件对象关联的dentry对象 \*/

struct vfsmount\* f\_vfsmnt;

struct file\_operations\* f\_op; /\* 文件对象的操作集合 \*/

atomic\_t f\_count; /\* 引用计数 \*/

unsigned int f\_flags; /\* 使用open()时设定的标志 \*/

mode\_t f\_mode; /\* 进程访问模式 \*/

int f\_error;

loff\_t f\_pos; /\* 对文件读写操作的当前位置　\*/

........

};

### 1.1.2 inode对象

inode对象存储某个文件的管理信息，通常对应磁盘文件系统的文件控制块。文件在文件系统内有唯一的inode号。struct inode结构的部分成员如下所述：

struct inode{

struct hlist\_node i\_hash; /\* 用于hasb表的指针 \*/

struct list\_head i\_list; /\* 根据inode状态链入相关链表 \*/

struct list\_head i\_sb\_list; /\* 链入超级快的inode链表 \*/

struct list\_head i\_dentry; /\* inode的dentry对象链表 \*/

unsigned long i\_ino; /\* inode号 \*/

atomic\_t i\_count; /\* 使用计数 \*/

umode\_t i\_mode; /\* 表示文件类型及权限 \*/

unsigned int i\_nlink; /\* 硬链接数 \*/

uid\_t i\_uid; /\* 文件拥有者用户id \*/

gid\_t i\_gid; /\* 用户所在组的id \*/

dev\_t i\_rdev; /\* 所在设备的设备号 \*/??? kdev\_t

loff\_t i\_size; /\* 以字节为单位的文件大小 \*/

struct timespec i\_atime; /\* 最后访问时间 \*/

struct timespec i\_mtime; /\* 最后修改(modify)时间 \*/

struct timespec i\_ctime; /\* 文件创建时间 \*/

.......

struct inode\_operations \*i\_op; / \* inode对象操作集合 \*/

struct file\_operations \*i\_fop; / \* 文件对象对象操作集合 \*/

struct super\_block \*i\_sb; / \* 所属的超级快 \*/

struct file\_lock \*i\_flock

struct address\_space \*i\_mapping;

.............

};

### 1.1.3 dentry对象

inode对象主要是描述目录项及相关联的inode信息。struct dentry 结构描述如下：

struct dentry {

.....

struct inode \* d\_inode; / \* 该dentry对象所属的inode \*/

struct dentry \*d\_parent; /\* 父目录 \*/

struct qstr d\_name; /\* 文件名 及附属信息 \*/

.....

struct dentry\_operations \*d\_op;

....

};

### 1.1.4超级块对象

struct super\_block{

struct list\_head s\_list; / \* 将所有的超级块链接起来 \*/

kdev\_t \*s\_dev; /\* 所在设备号 \*/

unsigned long s\_blocksize; /\* 该文件系统磁盘块的大小 \*/

.....

struct file\_system\_type \*s\_type; /\* 所属的文件系统类型 \*/

struct super\_operations \*s\_op;

....

};

图1描述了各种对象之间的关系。inode对象对应于一个文件， 该文件一般在磁盘上，inode对象与文件之间的关系是一对一的。而一个真正的文件可能有多个文件名，比如硬链接（Hard Link）,因此dentry对象与inode对象之间是多对一的关系。不同的进程可能打开同一个文件，但却不能用同一个文件对象来描述。因为操作标志、文件的操作位置等均可能不同。因而文件对象与dentry对象之间是多对一的关系。

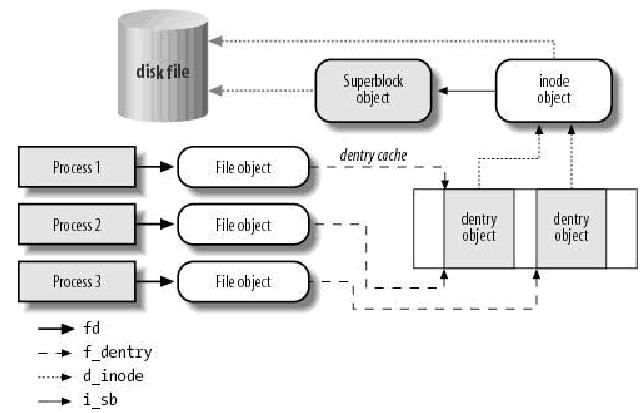


图1

为了加快文件的查找速度，内核引入了缓冲机制。最近使用的dentry对象放在dentry cache(简称dcache)中。最近使用的inode对象放在inode cache中。在cache中的对象又以Hash表的形式将它们组织起来。

### 1.2 文件系统的注册与安装

### 1.3 各种对象的操作接口

### 1.4 EXT2文件系统

未完