操作系统实验班大作业

VFS 研究报告

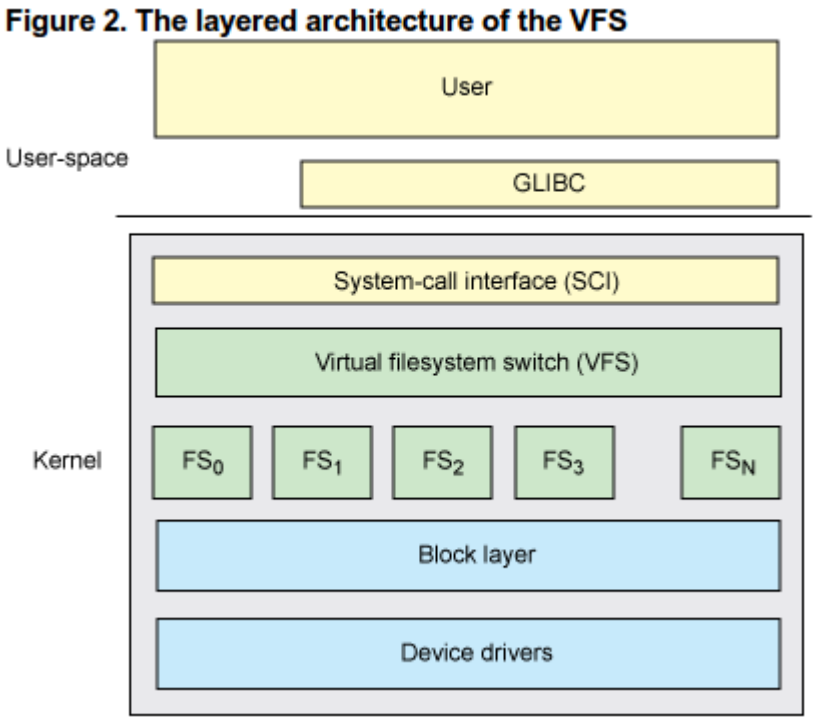
蒋捷 / 1200012708 & 兰兆千 / 1100012458 & 邢曜鹏 / 1200012835 &

赵万荣 / 1200012808 & 周昊宇 / 1200012823 （音序）

概述

VFS（Virtual File System）是 Linux 为文件读写提供的抽象层，Linux 通过这种方式增强了文件系统相关的可扩展性。下面从编写文件系统模块的角度考察 VFS：

Linux 中文件读写层次结构如下图所示：



这样，所有文件相关的系统调用都通过 VFS 抽象层提供的接口调用，而不需要知道具体的文件系统和设备。对于一个新的文件系统，只需要向 VFS 提供需要的操作，即注册过程之后即可使用。虽然 C 不是面向对象的语言，但 VFS 就是 Linux 内核中的一个抽象类，提供了一些接口需要具体的文件系统实现。

VFS 的抽象

在 VFS 中有四个重要的对象：

- Superblock：整个文件系统的抽象。
- Inode：文件系统中每一个文件的抽象。
- Dentry：文件在目录中的抽象。
- File：进程相关，打开文件的抽象。

1. Superblock

Superblock 提供了整个文件系统的元数据，VFS 中的 superblock 与 ext2 等文件系统实际存储的 superblock 不同，是内存中对文件系统的一个记录。VFS 维护了一个 superblock 的链表，用于记录当前系统中所有挂载的文件系统，其数据结构在 include/linux/fs.h 中定义，其中和实现相关最重要的域是 const struct super_operations *s_op，一个指向包含了高层次操作的对象的指针。

2. Inode

Inode 即 The index node，Linux 中所有的文件都通过 inode"直接"管理。同 superblock，这里的 inode 也是指内存中由 VFS 维护的数据结构，并非实际操作系统存储在设备上的 inode。除了文件的元数据（创建时间，修改时间等等）外，inode 中还保存了一个列表用来记录指向自己的 dentry。与实现相关的域是 const struct file_operations *i_fop 和 const stuct inode_operations *i_op，分别用来保存系统调用中文件的操作（open，read，write 等等）和 inode 相关的操作（如 create，lookup，link 等等）。

3. Dentry

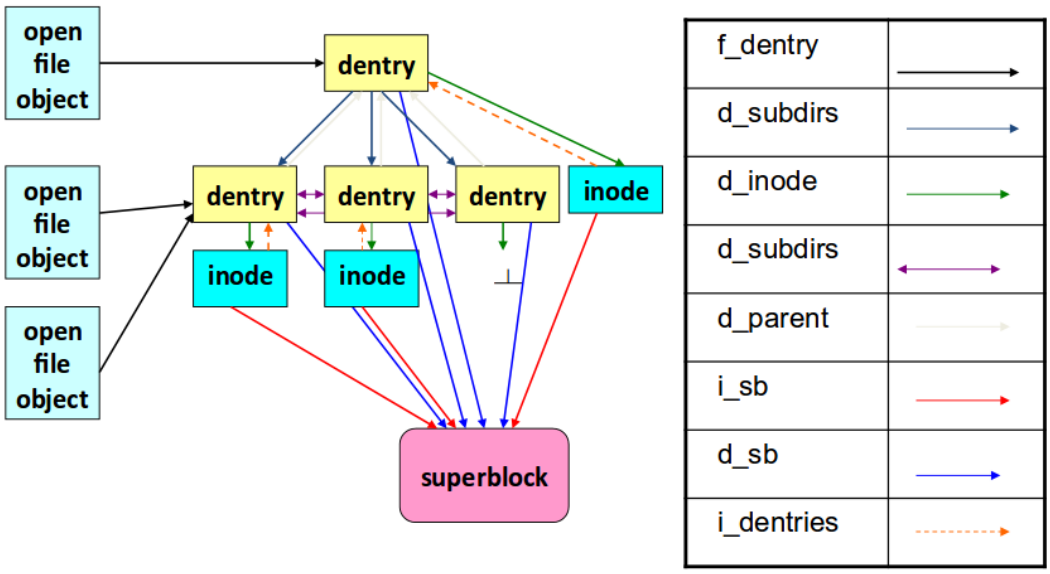
Dentry 即 Directory Entry，用于记录文件系统中目录层级（Inode 不能表示自己在哪一个目录）。唯一没有父目录的 dentry 对象是根目录，superblock 对象中包含对该 dentry 对象的引用。Dentry 结构体在 include/linux/dcache.h 中定义，其中包含了对父目录、目录包含对象列表的引用，与实现相关的域是 struct dentry_operations *d_op，一个指向 dentry 相关操作（hash，compare 等）对象的指针。另外，文件名也保存在 dentry 而非 inode 中。

4. File object

在 Linux 中每一个在进程中打开的文件都会对应一个 file 对象。这个对象记录了文件当前的位置，打开模式等等，除此之外包含了一个指向 dentry 的指针。与实现相关的域是 const struct file_operations，包含了文件相关的操作，这一结构与 inode 中相应的域相同，一般在 file 被创建时从相关的 inode 复制。

总结

这四个结构相互的引用关系可以用下图表示：



注意 dentry 和 inode 中都包含对 superblock 的引用

综上，对于创建一个新的文件系统，主要要实现以下四个数据结构

1. Super block 相关操作(struct super_operations)

read/write/clear/delete inode

write_super, put_super（释放 super）

write_super_locks, unlockfs, statfs

2. Inode 相关操作（struct inode_operations）

create: 为文件创建新的 inode

link/unlink/rename：实际上处理的是文件所在目录中的相关项（dentry）

symlink, readlink, follow_link: 软链接

mkdir/rmdir: 目录相关操作

mknod: 为设备文件创建 inode

truncate：修改文件大小

permission: 查看访问权限

3. File 相关操作（struct file_operations）

llseek, read, write, readdir, poll

mmap, open, flush, release, fsync 等等

4. **Dentry 相关操作**，这一部分大部分为默认操作，但对于一些特殊情况要特殊处理，如 DOS 文件系统中大小写不敏感，d_compare 函数就需要改动

文件系统的注册

文件系统模块通过 `int register_filesystem(struct file_system_type *);`函数进行注册，传入 `file_system_type` 结构，该结构中最重要的域是 `mount`，决定当系统调用 `mount` 方法时，如何挂载文件系统。此外还有 `kill_sb`，用于指定文件系统卸载时的内存清理工作和收尾工作。挂载的过程中要在内存中创建超级块，这个过程还会同时读取并在内存中创建根目录的 `inode`，在创建 `inode` 时根据 `inode` 类型将上面提到的这些结构体注册，最终完成整个注册过程。

参考资料

1. Anatomy of the Linux virtual file system switch
http://www.ibm.com/developerworks/library/l-virtual-filesystem-switch/

2. Design and Implementation of the Second Extended Filesystem
http://e2fsprogs.sourceforge.net/ext2intro.html

3. Linux VFS
http://www.cs.columbia.edu/~krj/os/lectures/L21-LinuxVFS.pdf

4. A tour of the Linux VFS
http://www.tldp.org/LDP/khg/HyperNews/get/fs/vfstour.html

5. The Linux Kernel's VFS Layer
https://www.usenix.org/legacy/event/usenix01/full_papers/kroeger/kroeger_html/node8.html

6. How to write a Linux VFS file system module
http://pages.cpsc.ucalgary.ca/~crwth/programming/VFS/VFS.php