

Virtual File System in Linux (Chap. 12 in Understanding the Linux Kernel)

J. H. Wang

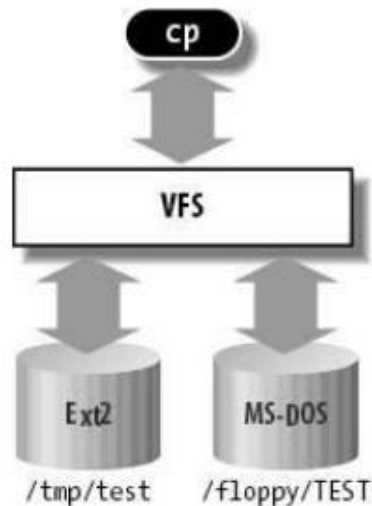
Nov. 22, 2011

Outline

- Role of VFS
- VFS Data Structures
- Filesystem Types
- Filesystem Handling
- Pathname Lookup
- Implementation of VFS System Calls
- File Locking

Role of VFS

- A common interface to several kinds of filesystems
 - Ex: `cp /floppy/TEST /tmp/test`



(a)

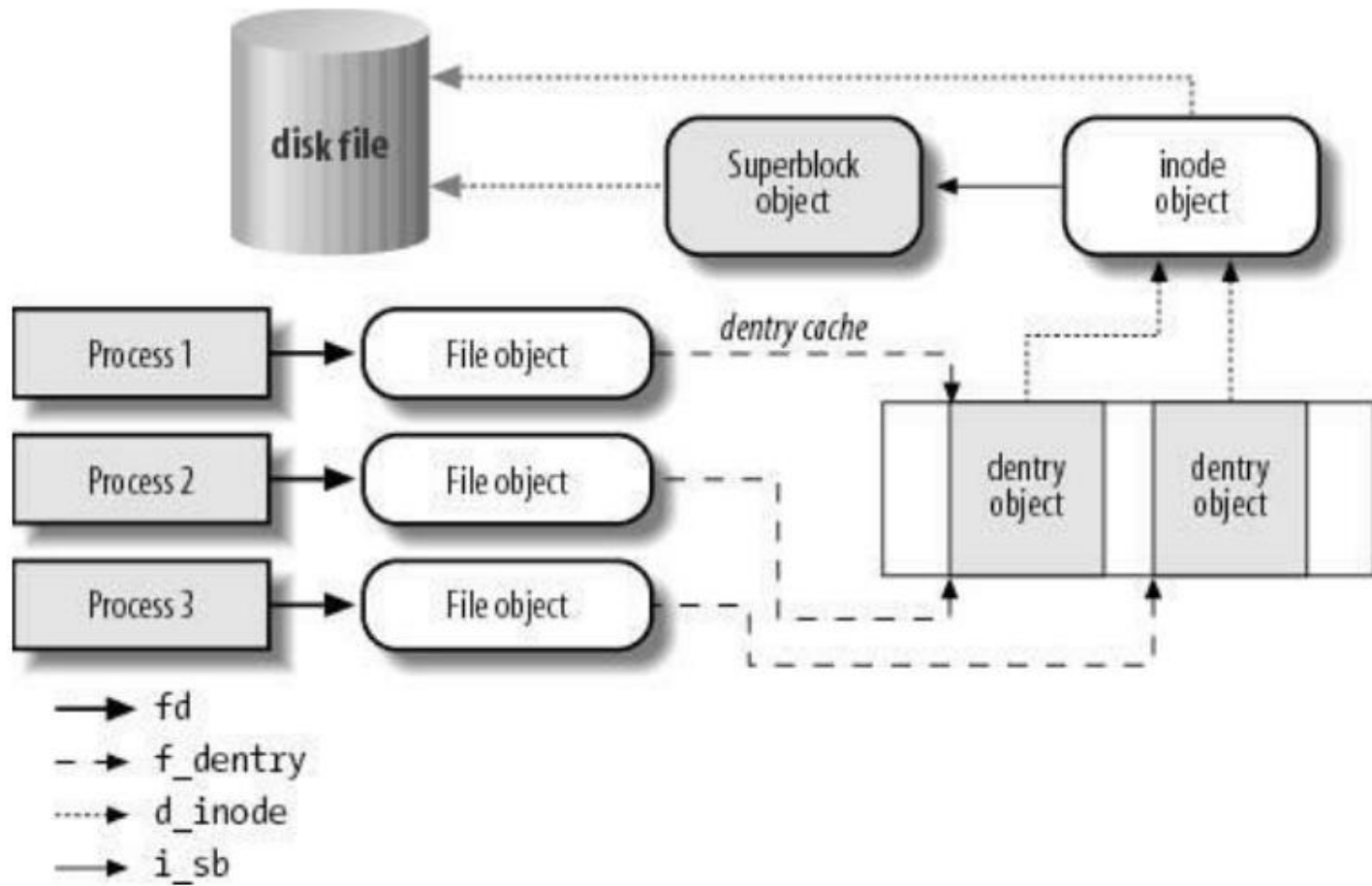
```
inf = open("/floppy/TEST", O_RDONLY, 0);
outf = open("/tmp/test",
            O_WRONLY|O_CREAT|O_TRUNC, 0600);
do {
    i = read(inf, buf, 4096);
    write(outf, buf, i);
} while (i);
close(outf);
close(inf);
```

(b)

- Filesystems supported by the VFS
 - Disk-based filesystems
 - Ext2, ext3, ReiserFS
 - Sysv, UFS, MINIX, VxFS
 - VFAT, NTFS
 - ISO9660 CD-ROM, UDF DVD
 - HPFS, HFS, ADFS, ADFS,
 - Network filesystems
 - NFS, Coda, AFS, CIFS, NCP
 - Special filesystems
 - E.g. /proc

The Common File Model

- Capable of representing all supported filesystems
 - Each specific filesystem implementation must translate its physical organization into VFS's common file model
 - E.g.: `read(...): file->f_op->read(...);`
- Object-oriented: data structures and associated operations
 - Superblock object: a mounted filesystem
 - Inode object: information about a file
 - File object: interaction between an open file and a process
 - Dentry object: directory entry



Some System Calls Handled by the VFS

- Filesystem
 - Mount(), umount(), umount2()
 - Sysfs()
 - Statfs(), fstatfs(), statfs64(), fstatfs64(), ustat()
- Directories
 - Chroot(), pivot_root()
 - Chdir(), fchdir(), getcwd()
 - Mkdir(), rmdir()
 - Getdents(), getdents64(), readdir(), link(), unlink(), rename(), lookup_dcookie()
- Links
 - Readlink(), symlink()

- Files

- Chown(), fchown(), lchown(), chown16(), fchown16(), lchown16()
- Chmod(), fchmod(), utime()
- Stat(), fstat(), lstat(), access(), oldstat(), oldfstat(), oldlstat(), stat64(), lstat64(), fstat64()
- Open(), close(), creat(), umask()
- Dup(), dup2(), fcntl(), fcntl64()
- Select(), poll()
- Truncate(), ftruncate(), truncate64(), ftruncate64()
- Lseek(), _llseek()
- Read(), write(), readv(), writev(), sendfile(), sendfile64(), readahead()

- Others

- Io_setup(), io_submit(), io_getevents(), io_cancel(), io_destroy()
- Pread64(), pwrite64()
- Mmap(), mmap2(), munmap(), madvise(), mincore(), remap_file_pages()
- Fdatasync(), fsync(), sync(), msync()
- Flock()
- Setxattr(), lsetxattr(), fsetxattr(), getxattr(), lgetxattr(), fgetxattr(), listxattr(), llistxattr(), flistxattr(), removexattr(), lremovexattr(), fremovexattr()

VFS Data Structures

- Superblock objects: super_block structure
 - (Table 12-2)
 - S_op: superblock operations – super_operations structure
 - Alloc_inode(), destroy_inode()
 - Read_inode(), dirty_inode(), write_inode(),
 - Put_inode(), drop_inode(), delete_inode()
 - Put_super(), write_super()
 - Sync_fs(), write_super_lockfs(), unlockfs(), statfs(), remount_fs()
 - Clear_inode(), umount_begin()
 - Show_options(), quota_read(), quota_write()

- Inode objects (Table 12-3): inode structure
 - i_op: inode operations – inode_operations structure
 - Create(), lookup(), link(), unlink(), symlink()
 - Mkdir(), rmdir(), mknod(), rename()
 - readlink(), follow_link(), put_link()
 - Truncate(), permission(),
 - Setattr(), getattr(), setxattr(), getxattr(), listxattr(), removexattr()

- File objects: file structure (Table 12-4)
 - File operations
 - Lseek(), read(), aio_read(), write(), aio_write()
 - Readdir(), poll(), ioctl(), unlocked_ioctl(), compat_ioctl()
 - Mmap(), open(), flush(), release()
 - Fsync(), aio_fsync(), fasync(), lock()
 - Readv(), writev(), sendfile(), sendpage()
 - Get_unmapped_area(), check_flags(), dir_notify(), flock()

- Dentry objects: (Table 12-5)
 - States: free, unused, in use, negative
 - Dentry operations
 - D_revalidate()
 - D_hash()
 - D_compare()
 - D_delete()
 - D_release()
 - D_input()
- Dentry cache
 - A set of dentry objects
 - A hash table

Files Associated with a Process

- fs field: fs_struct structure (Table 12-6)
- files field: files_struct structure (Table 12-7)
 - fd: file descriptors
 - fd[0]: stdin
 - fd[1]: stdout
 - fd[2]: stderr
 - NR_OPEN: max # of file descriptors for a process
 - Usually 1,048,576

Special Filesystems

- (Tabel 12-8)
 - /dev/pts: pseudo terminal support
 - /proc: general access point to kernel data structures
 - /sys: general access point to system data
 - /proc/bus/usb: USB devices
 - ...

Filesystem Type Registration

- File_system_type object: (Table 12-9)
- Fs_flags: (Table 12-10)

Filesystem Handling

- Root filesystem
- Mount point
- Namespaces: in Linux 2.6, each process might have its own tree of mounted filesystems
 - Namespace structure (Table 12-11)
- Filesystem mounting
 - It's possible in Linux to mount the same filesystem several times
 - Mounted filesystem descriptor: of type vfsmount (Table 12-12)
 - Mounting/unmounting the filesystem

- Pathname lookup
 - Pathname -> inode
 - Pathlookup(): return the nameidata structure (Table 12-15)
 - Standard pathname lookup
 - Parent pathname lookup
 - Lookup of symbolic links

Implementation of VFS System Calls

- Open()
- Read()
- Write()
- Close()

File Locking

- Advisory locks: by POSIX
 - Based on `fcntl()` system call
 - Possible to lock an arbitrary region of a file
- Mandatory locks: by System V Release 3
 - The kernel checks every invocation of `open()`, `read()`, `write()` system calls does not violate a mandatory lock
- Linux supports both + `fcntl()` and `flock()` system calls

Thanks for Your Attention!