```
/*
 * These are some of Linux’s base filesystem abstractions, as
 * taken from include/linux/fs.h
 */

/*
 * This is the inode structure.  Any filesystem object: Files in
 * filesystems, devices, sockets, pipes, etc. are all represented by
 * inodes.
 */
struct inode {
        kdev_t          i_dev;
        unsigned long   i_ino;
        umode_t         i_mode;
        nlink_t         i_nlink;
        uid_t           i_uid;
        gid_t           i_gid;
        kdev_t          i_rdev;
        off_t           i_size;
        time_t          i_atime;
        time_t          i_mtime;
        time_t          i_ctime;
        unsigned long   i_blksize;
        unsigned long   i_blocks;
        unsigned long   i_version;
        unsigned long   i_nrpages;
        struct semaphore i_sem;
        struct inode_operations *i_op;
        struct super_block *i_sb;
        struct wait_queue *i_wait;
        struct file_lock *i_flock;
        struct vm_area_struct *i_mmap;
        struct page *i_pages;
        struct dquot *i_dquot[MAXQUOTAS];
        struct inode *i_next, *i_prev;
        struct inode *i_hash_next, *i_hash_prev;
        struct inode *i_bound_to, *i_bound_by;
        struct inode *i_mount;
        unsigned short i_count;
        unsigned short i_flags;
        unsigned char i_lock;
        unsigned char i_dirt;
        unsigned char i_pipe;
        unsigned char i_sock;
        unsigned char i_seek;
        unsigned char i_update;
        unsigned short i_writecount;
        union {
                struct pipe_inode_info pipe_i;
                struct minix_inode_info minix_i;
                struct ext_inode_info ext_i;
                struct ext2_inode_info ext2_i;
                struct hpfs_inode_info hpfs_i;
                struct msdos_inode_info msdos_i;
                struct umsdos_inode_info umsdos_i;
                struct iso_inode_info isofs_i;
                struct nfs_inode_info nfs_i;
                struct xiafs_inode_info xiafs_i;
                struct sysv_inode_info sysv_i;
                struct affs_inode_info affs_i;
```

```
                struct ufs_inode_info ufs_i;
                struct socket socket_i;
                void * generic_ip;
        } u;
};

/*
 * ...and these are the operations you can perform on an inode.  Each
 * different kind of inode will have its own inode operations table.
 *
 * This structure is in effect the virtual function table for a
 * "sub-class" of the inode superclass, where examples of subclasses
 * of the inode superclass would include ext2 inodes, msdos inodes,
 * pipes, tty devices, etc.
 */
struct inode_operations {
        struct file_operations * default_file_ops;
        int (*create) (struct inode *,const char *,int,int,struct inode **);
        int (*lookup) (struct inode *,const char *,int,struct inode **);
        int (*link) (struct inode *,struct inode *,const char *,int);
        int (*unlink) (struct inode *,const char *,int);
        int (*symlink) (struct inode *,const char *,int,const char *);
        int (*mkdir) (struct inode *,const char *,int,int);
        int (*rmdir) (struct inode *,const char *,int);
        int (*mknod) (struct inode *,const char *,int,int,int);
        int (*rename) (struct inode *,const char *,int,
                       struct inode *,const char *,int, int);
        int (*readlink) (struct inode *,char *,int);
        int (*follow_link) (struct inode *,struct inode *,int,
                            int,struct inode **);
        int (*readpage) (struct inode *, struct page *);
        int (*writepage) (struct inode *, struct page *);
        int (*bmap) (struct inode *,int);
        void (*truncate) (struct inode *);
        int (*permission) (struct inode *, int);
        int (*smap) (struct inode *,int);
};

/*
 * Struct file represents an open file object.  Two file descriptors
 * can point to the same "struct file" if one was created using dup()
 * or dup2() of the first file descriptor.
 */
struct file {
        mode_t f_mode;
        loff_t f_pos;
        unsigned short f_flags;
        unsigned short f_count;
        unsigned long f_reada, f_ramax, f_raend, f_ralen, f_rawin;
        struct file *f_next, *f_prev;
        int f_owner;            /* pid or -pgrp where SIGIO should be sent */
        struct inode * f_inode;
        struct file_operations * f_op;
        unsigned long f_version;
        void *private_data;     /* needed for tty driver, and maybe others */
};

/*
 * These are the file operations.  Like the inode operations, each
 * different class of inode will tend to have a different file
```

```
 * operations pointer.  In some cases, the file operations may change
 * over time (for example, when a tty hangup happens and the tty
 * access revokes access to exsting file descriptors).
 */
struct file_operations {
        int (*lseek) (struct inode *, struct file *, off_t, int);
        int (*read) (struct inode *, struct file *, char *, int);
        int (*write) (struct inode *, struct file *, const char *, int);
        int (*readdir) (struct inode *, struct file *, void *, filldir_t);
        int (*select) (struct inode *, struct file *, int, select_table *);
        int (*ioctl) (struct inode *, struct file *,
                      unsigned int, unsigned long);
        int (*mmap) (struct inode *, struct file *, struct vm_area_struct *);
        int (*open) (struct inode *, struct file *);
        void (*release) (struct inode *, struct file *);
        int (*fsync) (struct inode *, struct file *);
        int (*fasync) (struct inode *, struct file *, int);
        int (*check_media_change) (kdev_t dev);
        int (*revalidate) (kdev_t dev);
};
```