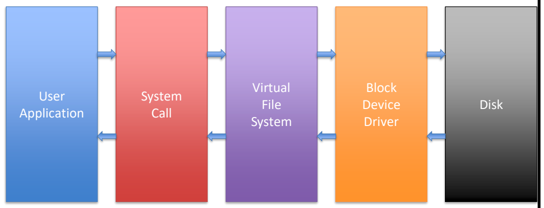
Filesystems

COSC-361
Stephen Marz

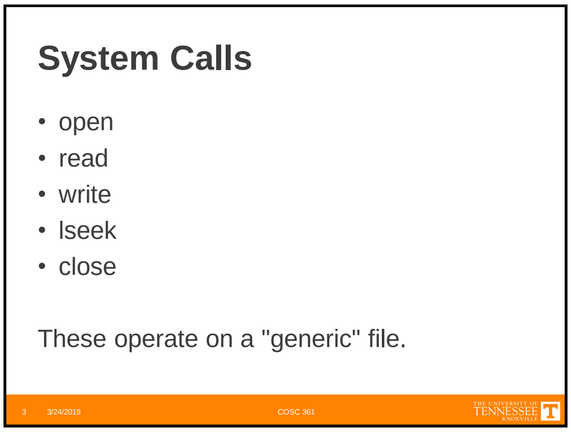
TICKLE COLLEGE OF ENGINEERING
MIN H. KAO DEPARTMENT OF ELECTRICAL ENGINEERING & COMPUTER SCIENCE

1

Data Route



2



System Calls

- open
- read
- write
- lseek
- close

These operate on a "generic" file.

3 3/24/2019 COSC 361

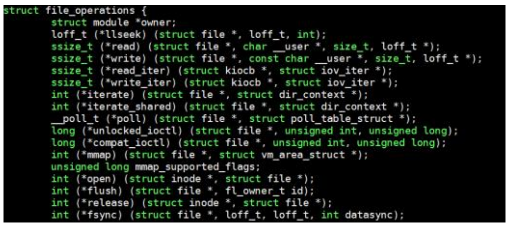
3

Virtual File System

- This system is in the kernel
  - It virtualizes the file system from the user application.
- It allows us to have a unified file tree that can support a multitude of file systems.
  - Example: /home on Hydra is an NFS drive
- We use the same system calls regardless of underlying file system type, thanks to VFS.

4

Block Device Drivers



```
struct file_operations {
        struct module *owner;
        loff_t (*llseek) (struct file *, loff_t, int);
        ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
        ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
        ssize_t (*read_iter) (struct kiocb *, struct iov_iter *);
        ssize_t (*write_iter) (struct kiocb *, struct iov_iter *);
        int (*iterate) (struct file *, struct dir_context *);
        int (*iterate_shared) (struct file *, struct dir_context *);
        __poll_t (*poll) (struct file *, struct poll_table_struct *);
        long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
        long (*compat_ioctl) (struct file *, unsigned int, unsigned long);
        int (*mmap) (struct file *, struct vm_area_struct *);
        unsigned long mmap_supported_flags;
        int (*open) (struct inode *, struct file *);
        int (*flush) (struct file *, fl_owner_t id);
        int (*release) (struct inode *, struct file *);
        int (*fsync) (struct file *, loff_t, loff_t, int datasync);
```

Function pointers are used for every file operation. This way, all I do is provide a function pointer that handles the given operation (open, close, etc).

5

Disk Protocols

- (S)ATA [S for Serial vs Parallel]
  - Advanced Technology Attachment
    - AT is for the IBM AT
- ATAPI
  - ATA Packet Interface (typically for CDROMs)
- SCSI
  - Small computer system interface
- SAS
  - Serial attached SCSI
- NVME
  - Non-volatile memory express
  - For PCI-express SSDs

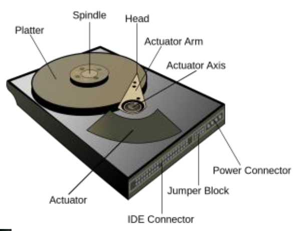
6

Input / Output

- PCI
  - Peripheral Component Interconnect
- IDE
  - Integrated Drive Electronics
- MMIO
  - Memory-mapped I/O

7

Disks

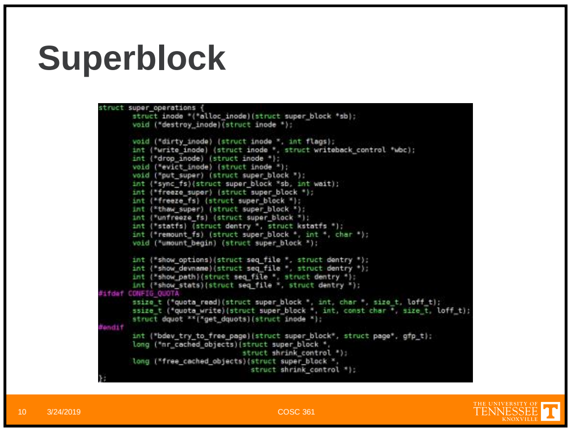


8

File Systems

- Unix-style
  - Superblock
  - Inodes
  - Blocks

9



Superblock

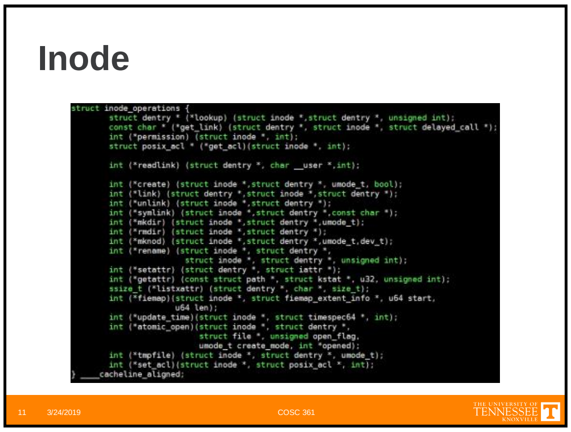
```
struct super_operations {
        struct inode *(*alloc_inode)(struct super_block *sb);
        void (*destroy_inode)(struct inode *);

        void (*dirty_inode) (struct inode *, int flags);
        int (*write_inode) (struct inode *, struct writeback_control *wbc);
        int (*drop_inode) (struct inode *);
        void (*evict_inode) (struct inode *);
        void (*put_super) (struct super_block *);
        int (*sync_fs)(struct super_block *sb, int wait);
        int (*freeze_super) (struct super_block *);
        int (*freeze_fs) (struct super_block *);
        int (*thaw_super) (struct super_block *);
        int (*unfreeze_fs) (struct super_block *);
        int (*statfs) (struct dentry *, struct kstatfs *);
        int (*remount_fs) (struct super_block *, int *, char *);
        void (*umount_begin) (struct super_block *);

        int (*show_options)(struct seq_file *, struct dentry *);
        int (*show_devname)(struct seq_file *, struct dentry *);
        int (*show_path)(struct seq_file *, struct dentry *);
        int (*show_stats)(struct seq_file *, struct dentry *);
#ifdef CONFIG_QUOTA
        ssize_t (*quota_read)(struct super_block *, int, char *, size_t, loff_t);
        ssize_t (*quota_write)(struct super_block *, int, const char *, size_t, loff_t);
        struct dquot **(*get_dquots)(struct inode *);
#endif
        int (*bdev_try_to_free_page)(struct super_block*, struct page*, gfp_t);
        long (*nr_cached_objects)(struct super_block *,
                                  struct shrink_control *);
        long (*free_cached_objects)(struct super_block *,
                                    struct shrink_control *);
};
```

10 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

10



Inode

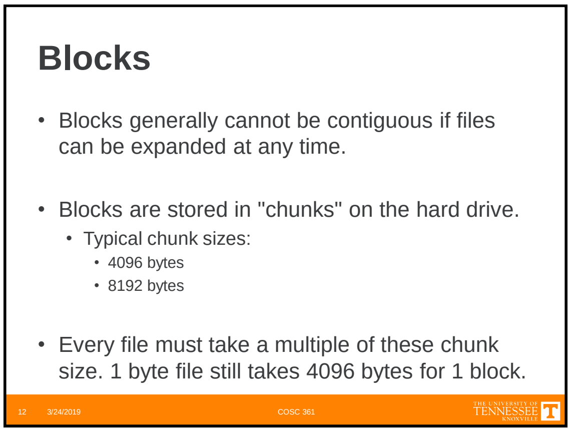
```
struct inode_operations {
        struct dentry * (*lookup) (struct inode *,struct dentry *, unsigned int);
        const char * (*get_link) (struct dentry *, struct inode *, struct delayed_call *);
        int (*permission) (struct inode *, int);
        struct posix_acl * (*get_acl)(struct inode *, int);

        int (*readlink) (struct dentry *, char __user *,int);

        int (*create) (struct inode *,struct dentry *, umode_t, bool);
        int (*link) (struct dentry *,struct inode *,struct dentry *);
        int (*unlink) (struct inode *,struct dentry *);
        int (*symlink) (struct inode *,struct dentry *,const char *);
        int (*mkdir) (struct inode *,struct dentry *,umode_t);
        int (*rmdir) (struct inode *,struct dentry *);
        int (*mknod) (struct inode *,struct dentry *,umode_t,dev_t);
        int (*rename) (struct inode *, struct dentry *,
                        struct inode *, struct dentry *, unsigned int);
        int (*setattr) (struct dentry *, struct iattr *);
        int (*getattr) (const struct path *, struct kstat *, u32, unsigned int);
        ssize_t (*listxattr) (struct dentry *, char *, size_t);
        int (*fiemap)(struct inode *, struct fiemap_extent_info *, u64 start,
                      u64 len);
        int (*update_time)(struct inode *, struct timespec64 *, int);
        int (*atomic_open)(struct inode *, struct dentry *,
                           struct file *, unsigned open_flag,
                           umode_t create_mode, int *opened);
        int (*tmpfile) (struct inode *, struct dentry *, umode_t);
        int (*set_acl)(struct inode *, struct posix_acl *, int);
} ____cacheline_aligned;
```

11 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

11



Blocks

- Blocks generally cannot be contiguous if files can be expanded at any time.
- Blocks are stored in "chunks" on the hard drive.
  - Typical chunk sizes:
    - 4096 bytes
    - 8192 bytes
- Every file must take a multiple of these chunk size. 1 byte file still takes 4096 bytes for 1 block.

12 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

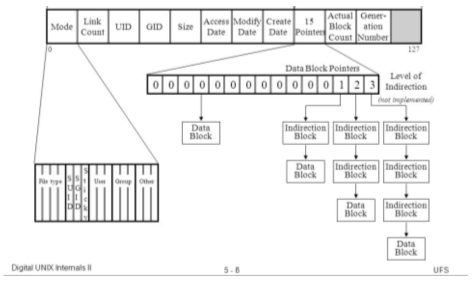
12

Locating Blocks

- The inode contains pointers that point to the blocks.
  - This process is known as ***indirection***.
  - This is necessary since inode is a fixed size and you can have a large number of blocks.

13

Block Indirection (UFS)

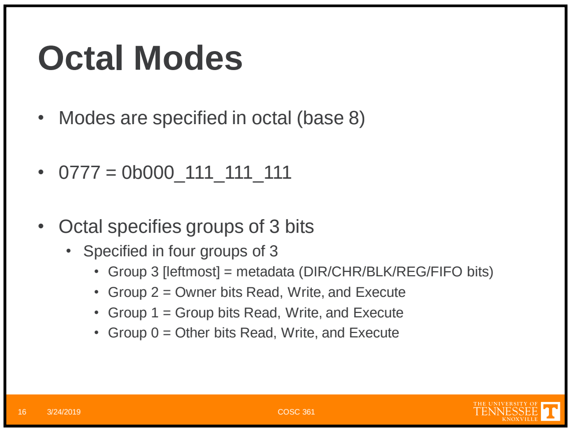


14

File Modes

- Describe permissions and other metadata for a file.
  - S_IFDIR - Directory bit
  - S_IFCHR - Char device bit
  - S_IFBLK - Block device bit
  - S_IFIFO - FIFO device bit

15

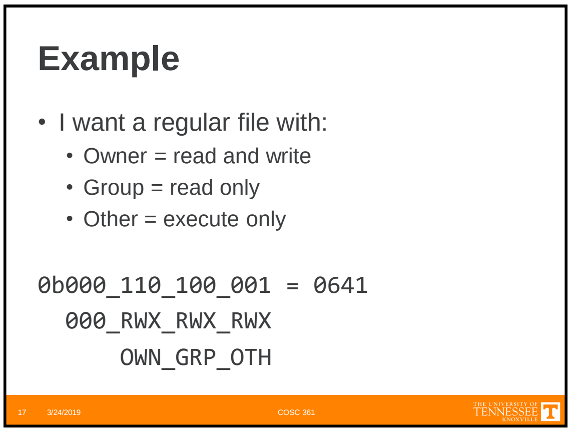


Octal Modes

- Modes are specified in octal (base 8)
- 0777 = 0b000_111_111_111
- Octal specifies groups of 3 bits
  - Specified in four groups of 3
    - Group 3 [leftmost] = metadata (DIR/CHR/BLK/REG/FIFO bits)
    - Group 2 = Owner bits Read, Write, and Execute
    - Group 1 = Group bits Read, Write, and Execute
    - Group 0 = Other bits Read, Write, and Execute

16 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

16



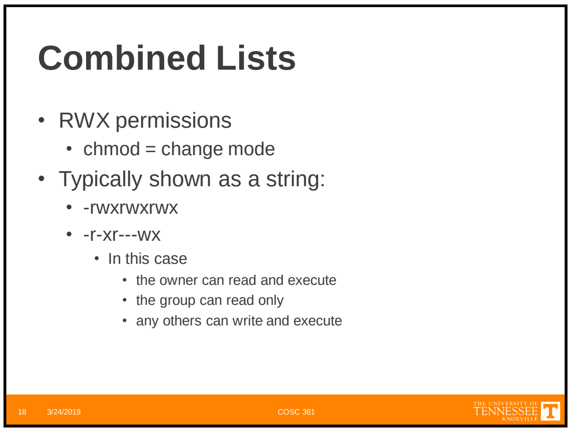
Example

- I want a regular file with:
  - Owner = read and write
  - Group = read only
  - Other = execute only

```
0b000_110_100_001 = 0641
  000_RWX_RWX_RWX
      OWN_GRP_OTH
```

17 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

17

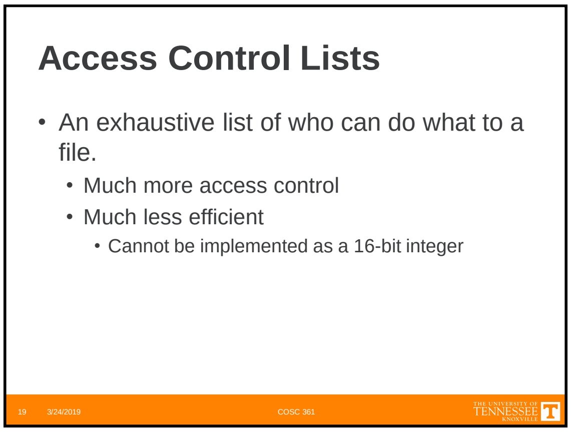


Combined Lists

- RWX permissions
  - chmod = change mode
- Typically shown as a string:
  - -rwxrwxrwx
  - -r-xr---wx
    - In this case
      - the owner can read and execute
      - the group can read only
      - any others can write and execute

18 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

18



Access Control Lists

- An exhaustive list of who can do what to a file.
  - Much more access control
  - Much less efficient
    - Cannot be implemented as a 16-bit integer

19 3/24/2019 COSC 361 THE UNIVERSITY OF TENNESSEE KNOXVILLE

19

Filesystems

COSC-361
Stephen Marz

THE UNIVERSITY OF TENNESSEE KNOXVILLE

20