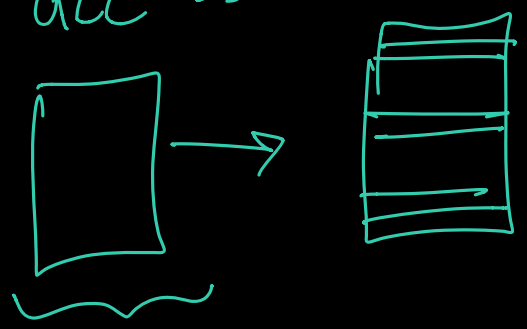
Operating Systems Design
12. File System Design

Paul Krzyzanowski
pxk@cs.rutgers.edu

Abstraction:

- Syscalls → Resource access
- Virtual memory →



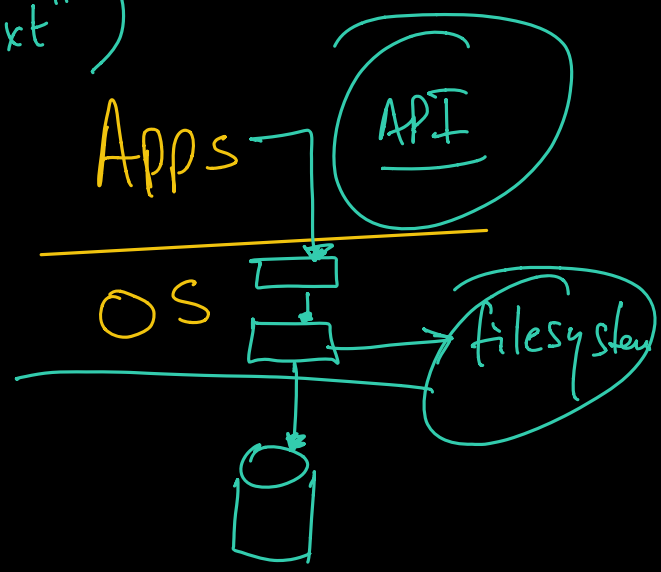
- Filesystems

a.txt →

Hardware

Persistent

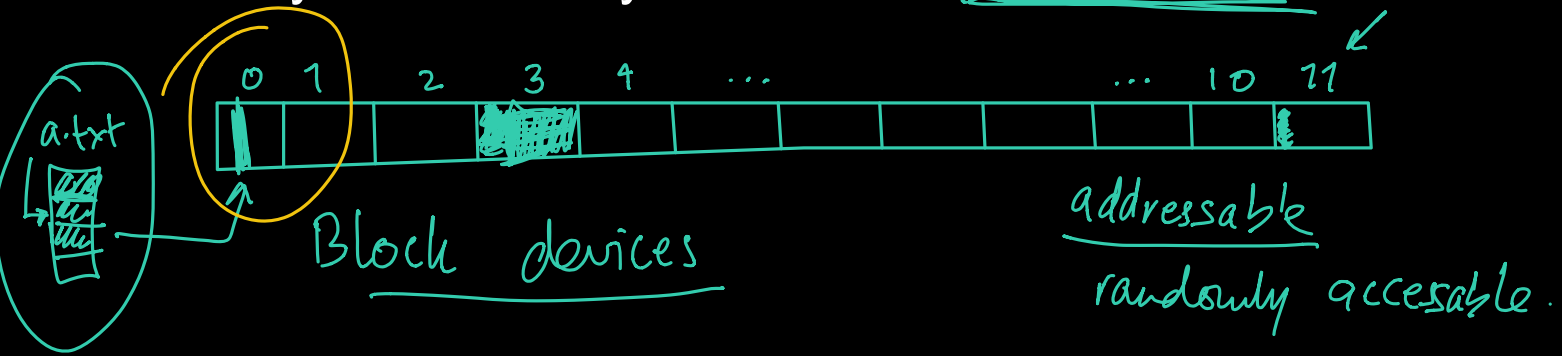
save("a.txt")



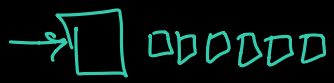
What's a file system

"This is dummy text".

- Organization of data and metadata
- What's metadata?
  - Attributes; things that describe the data
  - Name, length, type of file, creation/modification/access times, permissions, owner, location of data
- File systems usually interact with block devices



"Stream" devices



Design Choices

Namespace

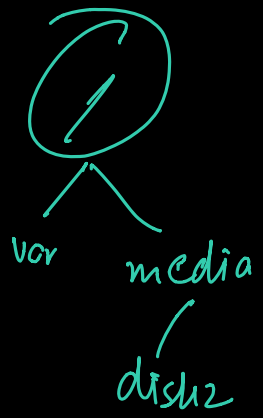
Flat, hierarchical, or other?

Amazon S3

Multiple volumes

Explicit device identification (A:, B:, C:, D:)

or integrate into one namespace?



Metadata

What kind of attributes should the file system have?

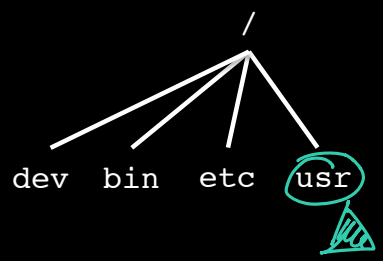
Implementation

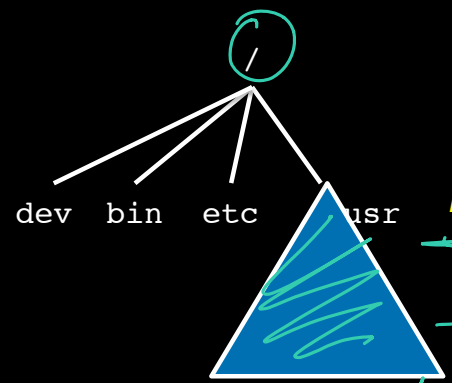
How is the data laid out on the disk?

Mounting

"mount"

- A file system must be *mounted* before it can be used by the operating system
- The *mount* system call is given the file system type, block device & mount point
- The mounted file system overlays anything under that mount point
- Looking up a pathname may involve traversing multiple mount points





mount point

→ filesystem

/media

/mnt/disk2/

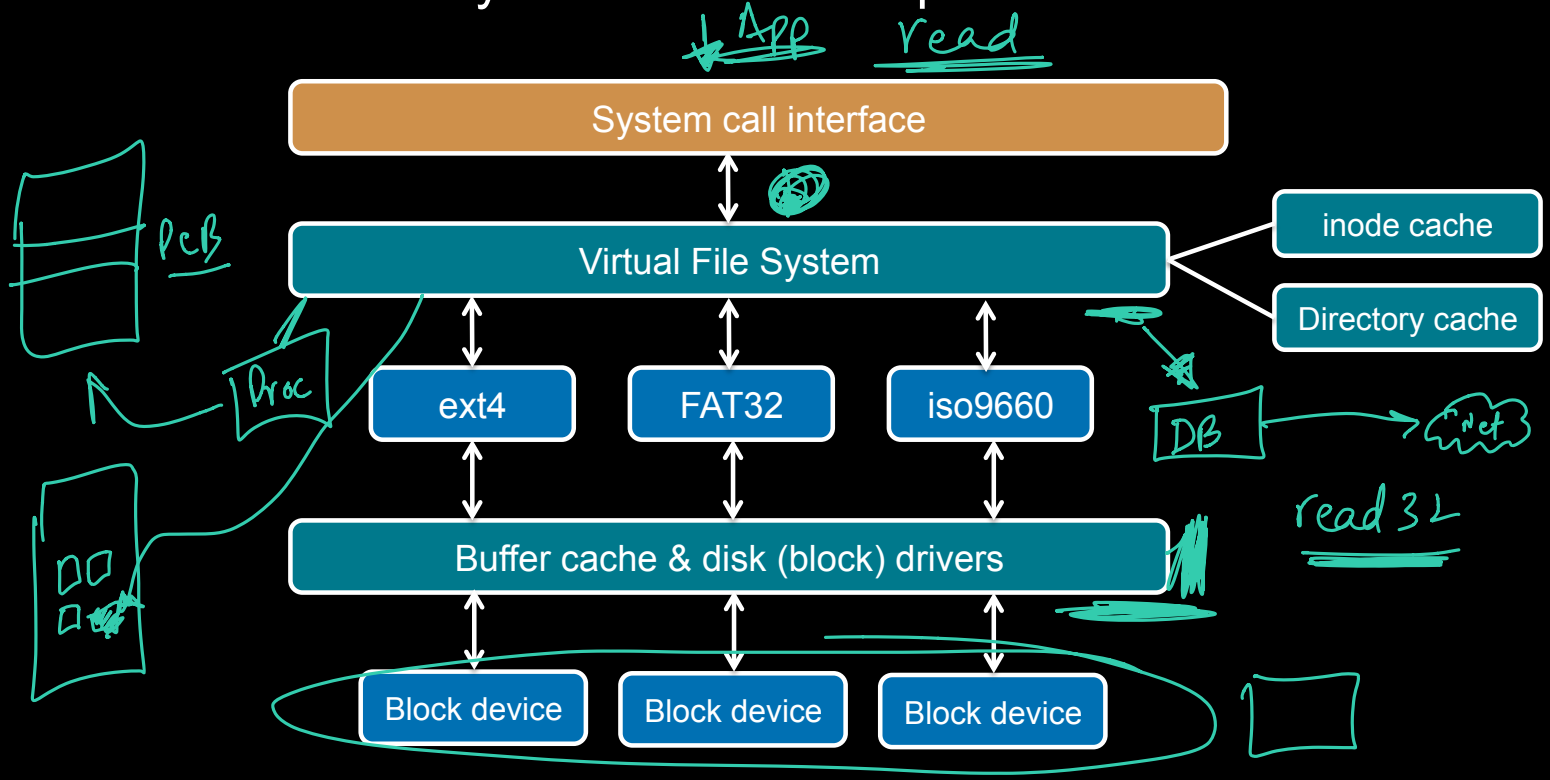
/usr/a/b.txt

/usr/a/c.txt

/usr/b/d.txt

Virtual File System (VFS) Interface

- Abstract interface for a file system object
- Each *real* file system interface exports a common interface



Keeping track of file system types

- Like drivers, file systems can be built into the kernel or compiled as loadable modules (loaded at mount)
- Each file system registers itself with VFS
- Kernel maintains a list of file systems

```
struct file_system_type {
        const char *name;
        struct super_block *(*get_sb)

};
```

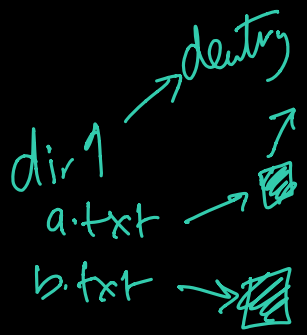
name of file system type

VFS superblock

- Structure that represents info about the file system
- Includes
  - File system name
  - Size
  - State
  - Reference to the block device
  - List of operations for managing inodes within the file system:
    - *alloc_inode*, *destroy_inode*, *read_inode*, *write_inode*, *sync_fs*, …

VFS: Common set of objects

- Superblock: Describes the file system
  - Block size, max file size, mount point
  - One per mounted file system
- inode: represents a single file
  - Unique identifier for every object (file) in a specific file system
  - File systems have methods to translate a name to an inode
  - VFS inode defines all the operations possible on it
- dentry: directory entries & contents
  - Name of file/directory, child dentries, parent
  - Directory entries: translations of names to inodes
- file: represents an open file
  - VFS keeps state: mode, read/write offset, etc.



inode

- Uniquely identifies a file in a file system
- Access metadata (attributes) of the file (except name)

```
struct inode {
        unsigned long i_ino;
        umode_t i_mode;
        uid_t i_uid;

        struct timespec i_atime;
        struct timespec i_ctime;
        struct timespec i_mtime;
        struct super_block *i_sb;

        struct list_head i_dentry;
        ...
}
```

inode operations

Functions that operate on file & directory *names and attributes*

```
struct inode_operations {
        int (*create)

        int (*link)

        int (*symlink)
        int (*mkdir)
        int (*rmdir)
        int (*mknod)
        int (*rename)


        int (*permission)
        int (*setattr)
        int (*getattr)



};
```

File operations

Functions that operate on file & directory *data*

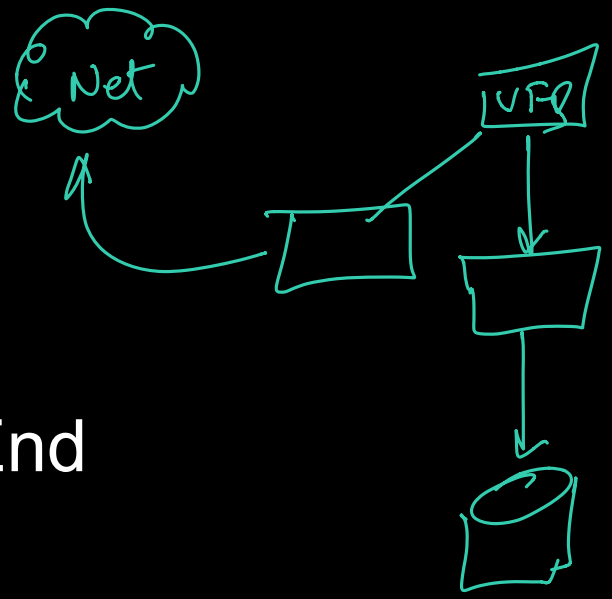
```
struct file_operations {
        struct module *owner;

        ssize_t (*read)

        ssize_t (*write)



        int (*mmap)
        int (*open)



        int (*lock)
```

Web hosting

- Space
- RAM
- CPU

- inodes

"inode count"



The End