File Systems

File System
- Persistant data storage
- Typically a core functionality of the system, OS managed
- Organize data into natural coherent units called files (self contained entity)
- Provides simple organization principle for a collection of files

Persistant Data Options
- Use raw storage blocks to store the data
    - Ona hard disk, flash drive
    - Does Not work for users or devs
- Database to store data
    - More structure and overhead than we can afford
- File system
    - More organized way of structuring persistent data
    - Makes sense to both users and devs

Flash Drives
- Solid sate persistent storage
- Reads and writes are fast
- A block can only be written once
    - Writing again requires an erase, which is slow
    - Slower when erasing large sections of data

Data and Metadata
- Data: information that the file stores
- Metadata: Information about the information the file stores
    - Ex. Number of bytes and when it was created
    - Attributes
- Data and metadata all stored persistently

Desirable File System Properties
- Persistent
- Easy to use
- Flexible
- Portale across hardware device types
- High performance
- Reliable
- Secure

    Performance Issue
    - Storage read and writes are slower than CPU instructions and memory

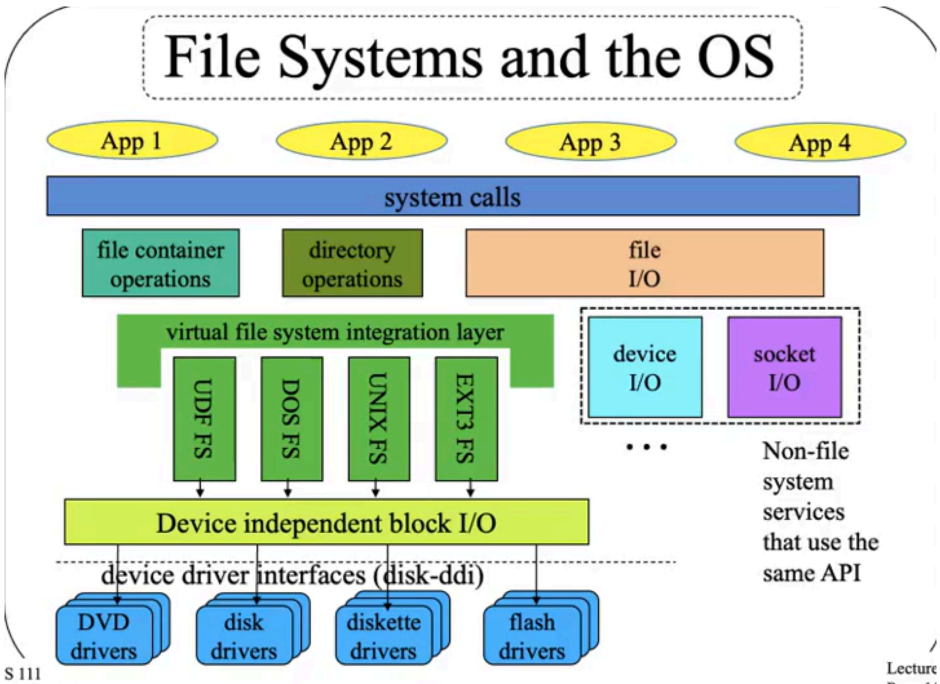
    Reliability
    - Every file that is written must be exactly the same

- Must be completely free of eros in both the hardware and software
- Race condition issues

Security

- Well denied access control model and mechanism
- Strong guarantees that the system enforces the controls

File System Design



- Socket: I/O

File System API

- Single API for programmers and users to use files (regardless of implementation)
- Requirement for program portability (so program can interact with the file system)

File Container Operations

- Standard file management sys calls
  - Manipulate files as objects
  - Operations ignore the contents of the file
- Implemented with standard file system methods
  - Get/set, ownership, protection
  - Create/destroy files, directories, and links

Directory Operations

- Directories provide hierarchical file organization
- Directories are file pointers

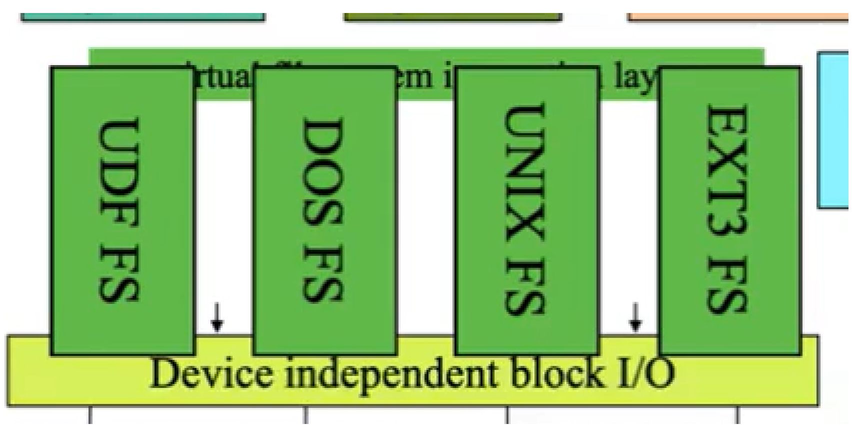
- Related to finding file by name, creating file mapping, and listing a set of known names in the directory

File I/O

- Open: use name to set up an open instance
- Read data and write data to a files
    - Implement logical block fetches
    - Copy data between user space and file buffer
- Seek
    - Change logical offset associated with open instance
- Map file into address space
    -

Virtual File Systems Layer

- Virtual File System Application Layer performs all the operations on the file system
    - Acts as an API between the app and the operations done on the file system
    - Each file system implemented a plug in module
- All implemented same basic methods
    - Create, delete, open, link, etc



File System Layer

- Desirable to support multiple file systems, where all are implemented on top of block I/O (4K blocks common)
- All systems perform same basic functions
    - Map names to files
    - Manage free space and allocate to files
    - Create/destroy files

Multiple File Systems

- Multiple storage devices
- Different file systems provide different services and are used for different purposes
    - Reliability, performance, read only vs write only

File Systems and Block I/O Devices

- Files systems typically sit on a general block I/O layer
- Generalizing abstraction to make hardware look identical
- Implements standard operations on each block device
    - Asynchronous reads and writes (physical block number, buffer, bytecount)
- Map logical block numbers ot device addresses

Device Independent Block I/O

- Better abstraction than generic drives
- Allows unified LRU buffer cache for drive data
    - Hold frequently used data until its needed again
    - Hold pre fetched read ahead data until its requested
- Provides buffers for data re-blocking
    - Adapting file system block size to device block size
    - Adapting file system block size to user request sizes
- Handles automatic buffer management
    - Allocation and deallocation
    - Automatic write-back of changed buffers

Caches

- File access exhibits a high degree of locality of references
    - Uses often read and write parts of a single block
- Common cache eliminates the need for slow disk accesses

Single Block I/O cache

- More efficient than multiple users access the same file
- Single cache provides higher hit ratio and thus better performance

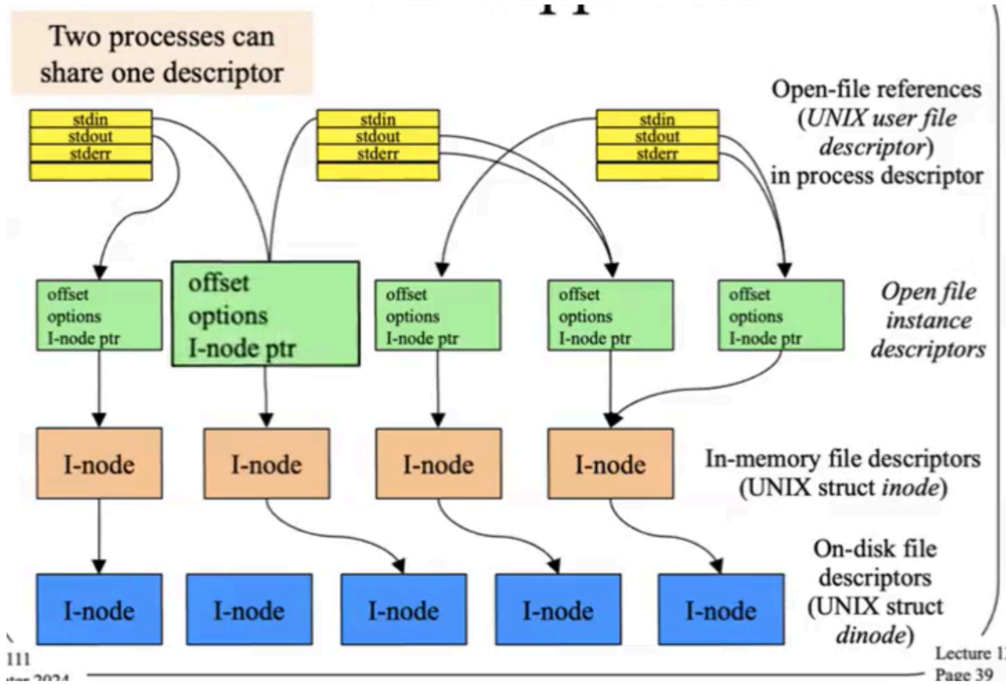
In Memory File System

- File system needs to store and retrieve data as well as manage the space the data is stored

In Memory Representation

- On disk structure pointing to device blocks
- When file is opened, in memory structure is created
- Not an exact copy of the device version
    - Device version points to device blocks

Unix In Memory File System



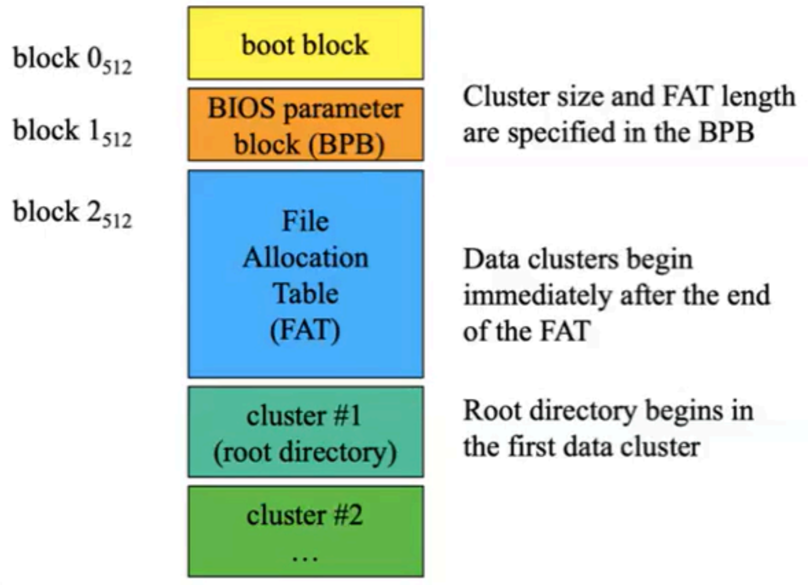
Boot Block

- 0th block of a device is reserved for the boot block
  - Code allowing the machine to boot an OS
- Not in control of a file system

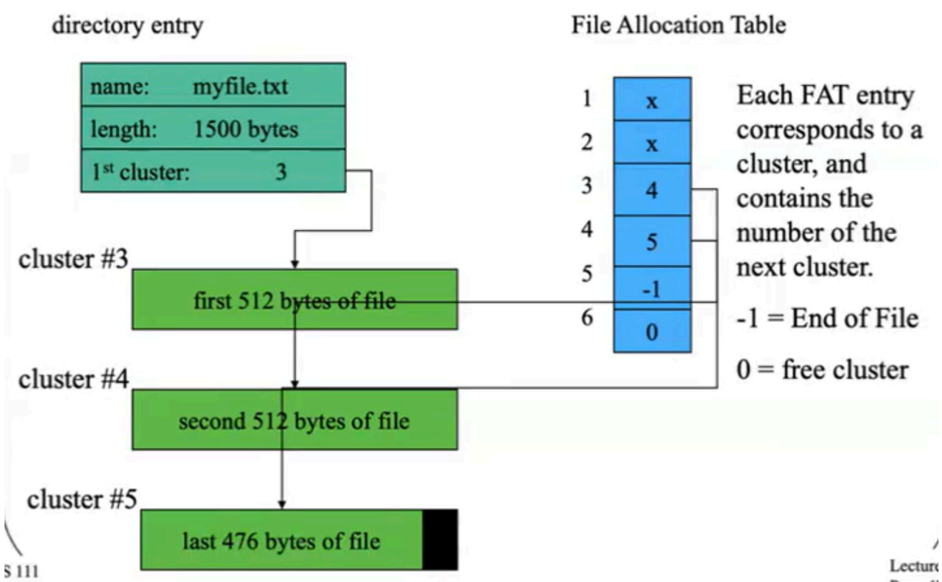
Linked Extents

- File control block contains exactly one pointer
  - To the first chunk of the file
  - Each chunk has a pointer to the next chunk
  - Allows us to add as many chunks to each file
- Pointers can be in chunks themselves
- Pointers can be in auxiliary chunk tables
  - Results in faster searches, especially if the table is kept in memory

DOS File System



- File System divides space into clusters
    - Cluster size fixed for each file system
    - Clusters numbered 1 through n
- File control structure points to first cluster of a file
- File Allocation Table (FAT)
    - One entry per cluster
    - Contains the number of the next cluster in file
    - 0 entry means that the cluster is not allocated
    - -1 means end of file



- Go to the current chunks FAT entry to get the next block

DOS Characteristics

- To find a particular block of a file
    - Get the number of the first cluster from directory entry
    - Follow chain of pointer through PAT
- Entire table is kept in memory
    - No disk I/O used to find a cluster
    - For very large files the in memory search can still be long
- Width of FAT determines max file system size
    - Number of bits which describe a cluster address

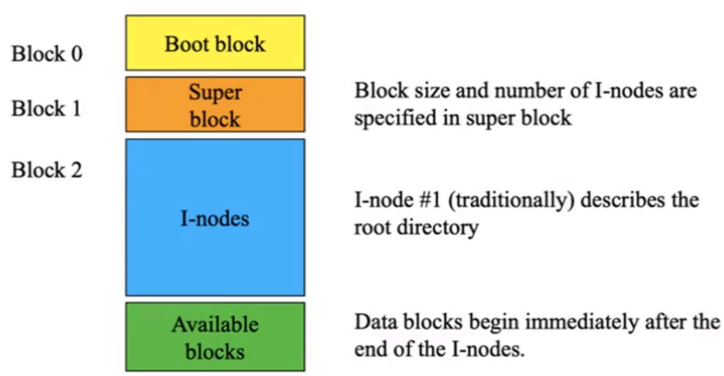
File Index Blocks

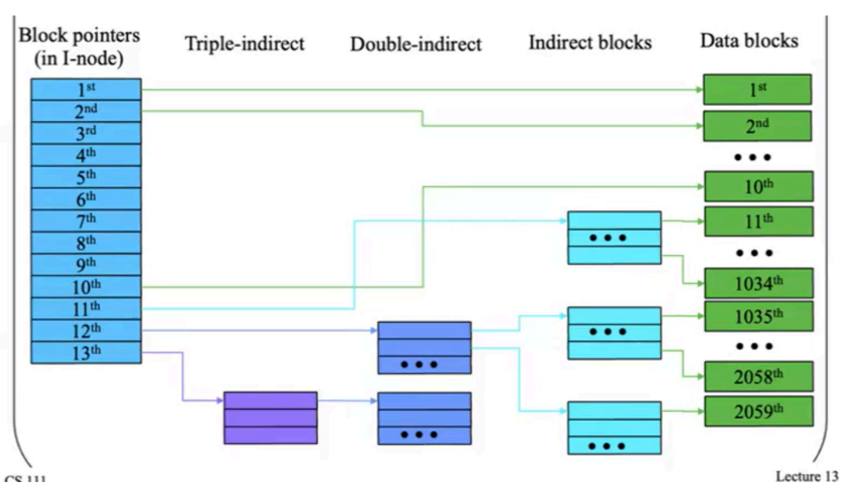
- File control block points to all blocks in file
- Very fast access to any desired block
- File control block could point at extent descriptors

Hierarchically Structured File Index Blocks

- Solves problem of file size being limited by entries in file index block
- Basic file index block points to blocks, which contain pointers to other blocks
- Can point to many extents

Unix V File System





Unix Inode Performance Issues

- Inode is in memory when file is open
- First 10 blocks can be found with no extra I/O
- After that read indirect blocks
    - Real pointers are in indirect blocks
- 1-3 extra I/O operations per thousand blocks