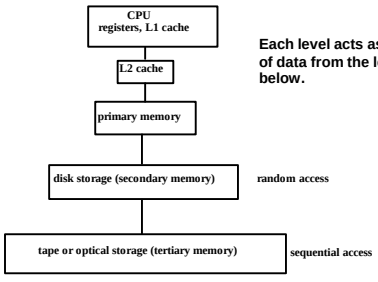
Secondary Storage Management

Secondary Storage

- **Secondary Storage is usually:**
  - anything outside of "primary memory"
  - storage that does not permit direct instruction execution or data fetch by load/store instructions
  - it's large
  - it's cheap
  - it's non-volatile
  - it's slow

The Memory Hierarchy

? **Memory is arranged as a hierarchy**

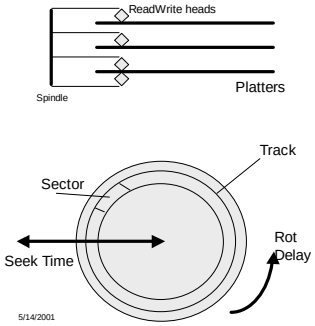


Physical Disks

- **The OS must deal with the mess of physical devices:**
  - errors
  - bad blocks
  - missed seeks
- **The job of the OS is to hide this mess from higher levels by:**
  - providing low-level device control
  - providing higher-level abstractions: files, database, ...
- **The OS may provide different levels of disk access**
  - physical disk block (surface, cylinder, sector)
  - disk logical block (disk block #)
  - file logical (file block, record, or byte #)

Physical Disk Structure

- **Disk are made of metal platters with a read/write head flying over it.**
- **To read from disk, we must specify:**
  - cylinder #
  - surface #
  - sector #
  - transfer size
  - memory address
- **Transfer time includes: seek, latency, and transfer time**



Some Typical Numbers

- Sector size: 512 bytes
- Cylinders per disk (tracks per platter): 6962
- Platters: 3 - 12
- Rotational speed: 10000 RPM
- Storage size: 4 - 80 GB
- Seek time: 5 - 12 ms
- Latency: 3 ms
- Transfer rate: 14 - 20 MB/sec

Disk Structure

- **There is no structure to a disk except cylinders and sectors, anything else is up to the OS.**
- **The OS imposes some structure on disks.**
- **Each disk contains:**
  - 1. data: e.g., user files
  - 2. meta-data: OS info describing the disk structure
- **For example, the free list is a data structure indicating which disk blocks are free. It is stored on disk (usually) as a bit map: each bit corresponds to one disk block.**
- **The OS may keep the free list bit map in memory and write it back to disk from time to time.**

Dealing with Mechanical Latencies

- Caches
  - locality in file access
- RAM disk
  - cheap, slow, big memory on the disk.
- RAID
  - parallelism
- Clever layouts and scheduling algorithms
  - head scheduling
  - meta-information layout

Bad Blocks

- With increasing densities, all disks have some bad blocks, and some go bad as time goes on.
- The OS can remove that block from its allocation map.
- On some disks, each cylinder contains a set of replacement blocks that the device can remap to replace other "logical" blocks on the same cylinder that are bad.

The File System

- The file system supports the abstraction of file objects. It supports creation, deletion, access, naming, sharing, and protection.
- A file is simply a named collection of data.
- The structure and interpretation of that data is typically defined by its creator and unknown to the file system.
- In some systems, though, the file type is known to the system, to prevent improper file manipulation.

Directories

- Directories support file naming and location.
- Most systems (like unix) support multi-level directories, where a file name describes its path from a root through the directories to the file at the leaf.
- Most systems have a *current* directory, from which names can be specified relatively, as opposed to absolutely from the root of the directory tree.

Directories

? **Conceptually, a directory describes the logical information about a file, e.g.:**

- file name
- file type
- file size
- location on disk
- current position of open file
- protection
- creation and last access time
-.... other stuff

(this info may or may not be actually stored in the directory)

Protection

- **Files can be protected in different ways:**
  - not at all (open system, single-user system)
  - protected access: read, write, execute, append, delete
  - complete access control list: list of users who have (or are denied) access, along with access allowed/denied
  - simple group schemes: owner, group, everyone

Access Methods

- **Some file systems provide different access methods that specify the data to read in different ways:**
  - sequential access: read bytes one at a time, in order
  - direct access: random access, given block/byte number
  - record access: file is array of fixed- or variable-length records, read/written sequentially or randomly by record number
  - indexed access: file system contains an index to a particular field of each record in a file. reads specify a value for that field, and the system finds the record through the index.

Meta-Data

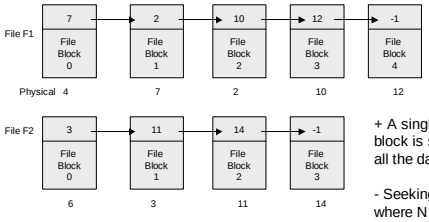
- **How the meta-data is represented is an OS issue, e.g., the free list could be a bit map, or a linked list (each free block points to next one), or something else.**
- **Disk storage (files) can be allocated in different ways:**
  - contiguously on disk
    - it's fast and simplifies directory access
    - it's inflexible, causes fragmentation, needs compaction
  - linked structures
    - each block contains a pointer to the next
    - good only for sequential access
  - indexed structures
    - store index to all blocks in 1 index block
    - good for random access.

Storing Files

- **Files can be allocated on disk in different ways, e.g.:**
  - 1. contiguous allocation
    - like memory
    - fast and simplifies directory access
    - inflexible, causes fragmentation, needs compaction
  - 2. linked structure
    - each block points to next block, directory points to first
    - good for sequential access (bad otherwise)
  - 3. indexed structure
    - an "index block" contains pointers to many other blocks
    - better for random access
    - may need multiple index blocks (linked together)

Linked-list Allocation

- Every data block in a file contains a pointer to the next data block

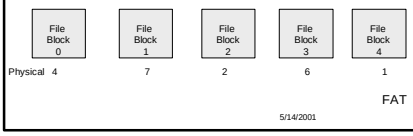


+ A single pointer to the first block is sufficient to locate all the data blocks in a file

- Seeking takes O(N) time where N is the size of the file

MS-DOS Filesystem

- MSDOS uses a file allocation table (FAT)
- Just like a linked-structure, except all the pointer information is stored in a separate table
  - For every block, the FAT keeps track of whether or not it is allocated, and if so, which block it points to
  - There are two copies of the FAT on disk



FAT

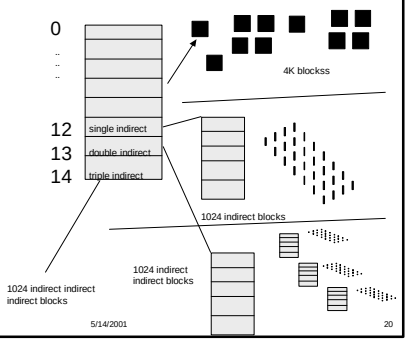
Physical	InUse	Next
0	0	-1
1	1	-1
2	1	6
3	0	-1
4	1	7
5	0	-1
6	1	1
7	1	2

MS-DOS Filesystem

- Advantage:
  - Can seek to any location in a file with a single disk read
- Primary disadvantage: space for FAT
  - Want to keep FAT in memory
  - 20GB disk, 1 KB block size = 20 million entries
  - Each disk location specifier is 3 or 4 bytes
  - Need 60 – 80MB of memory for the FAT
- Secondary disadvantage: does not deal with failures well
  - Two errors will take out the whole filesystem

UNIX Inodes

- A UNIX Inode is the metainformation for UNIX files.
- Contains control and allocation information.
- Each inode contains 15 block pointers.
  - first 12 are direct blocks
  - then single, double, and triple indirect

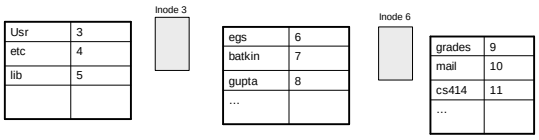


UNIX Inodes

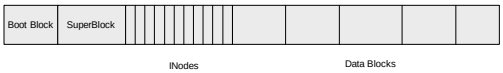
- Data blocks are 4K
- First 48K are directly reachable from the inode
- A single-indirect block containing 1024 entries addresses 4K times 1024 = 4M of data
- A double-indirect block addresses 1024 x 1024 x 4K = 4G
- A triple-indirect block addresses 1024 x 1024 x 1024 x 4K = 4T
- Any block can be found with at most 3 disk accesses

UNIX Directories

- Unix directories are just like regular files
  - They contain <filename, inode number> tuples

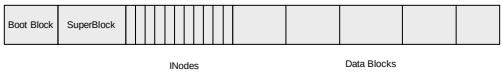


Unix Disk Layout



- Boot block provides information on how to boot the computer
- Superblock contains the filesystem layout: # of inodes, block size, start of the list of free blocks
  - System V:
    - Free blocks are kept in a list
    - Superblock contains the list of first 100 superblocks
    - When the 100th block is allocated, move its list into superblock
  - FFS:
    - Bitmap per cylinder
    - Each cylinder has its own inodes and data blocks.

Unix Disk Layout



- How do you look up /usr/egs/mail
  - Look up the inode of the "/" directory in the superblock, say 2
  - Read inode 2, go to all the data blocks and read the directory
  - See if "usr" appears in the "/" directory, if so, find its inode number
  - Check "usr" directory for subdirectory "egs"
  - Check "egs" directory for file "mail"
  - Start reading the first data block of "mail"

Unix Filesystem and Faults

- Problem 1: Disks used to be one of the most unreliable components in a computer system
  - Prone to developing “bad blocks”
  - Modern hardware often detects such faults and has spare blocks that it can transparently remap in place of the bad blocks
  - The filesystems still need to track bad blocks and avoid using them
  - Inode 1 is a special inode that keeps track of where all the bad blocks are
- Problem 2: System crashes or power failures can occur at any time
  - Any disk operation can be interrupted at any time

Unix file updates

- Need to ensure that the filesystem is consistent throughout updates
  - Data that is being modified may be lost, but the entire filesystem should not be jepordized
- A write in Unix involves
  - Writing the new data
  - Updating the inode
  - Updating the free list
- Is there a correct order ? What can go wrong if the FS does not respect the ordering requirements ?

Unix file updates

- Ordering requirements
  - Writing the new data MUST HAPPEN BEFORE
  - Updating the inode
  - Updating the free list
- Is there a correct order ? What can go wrong if the FS does not respect the ordering requirements ?

Disk Scheduling

- **Because disks are slow and seeks are long and depend on distance, we can schedule disk accesses, e.g.:**
  - **FCFS (do nothing)**
    - ok when load is low
    - long waiting times for long request queue
  - SSTF (shortest seek time first)
    - always minimize arm movement. maximize throughput.
    - favors middle blocks
  - SCAN (elevator) -- continue in same direction until done, then reverse direction and service in that order
  - C-SCAN -- like scan, but go back to 0 at end
- **In general, unless there are request queues, it doesn’t matter**
- **The OS (or database system) may locate files strategically for performance reasons.**