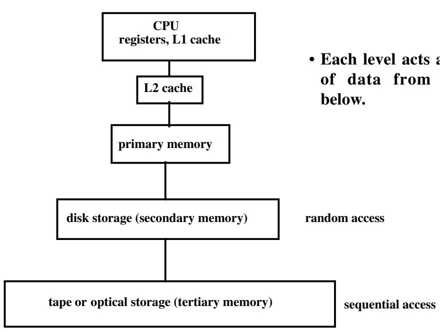
Secondary Storage Management

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**



- **Each level acts as a cache of data from the level below.**

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**

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. <u>data</u>: e.g., user files
  2. <u>meta-data</u>: OS info describing the disk structure
- **For example, the <u>free list</u> is a data structure indicating which disk blocks are free. It is stored on disk (usually) as a <u>bit map</u>: 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.**

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.

Storing Files

- **Files can be allocated on disk in different ways, e.g.:**
  1. contiguous allocation
     - 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)

Disk Scheduling

- **Because disks are slow and seeks are long and depend on distance, we can schedule disk accesses, e.g.:**
  - SSTF (shortest seek time first)
  - 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.**

Bad 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 <u>file</u> 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 <u>type</u> 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 <u>relatively</u>, as opposed to <u>absolutely</u> 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 <u>access methods</u> 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 <u>index</u> to a particular <u>field</u> of each record in a file. reads specify a value for that field, and the system finds the record through the index.