**File System**

### File

* Contiguous logical address space
* Types:
	+ Data
		- numeric
		- character
		- binary
	+ Program

### File Structure

* None - sequence of words, bytes
* Simple record structure
	+ Lines
	+ Fixed length
	+ Variable length
* Complex Structures
	+ Formatted document
	+ Relocatable load file
* Can simulate last two with first method by inserting appropriate control characters
* Who decides:
	+ Operating system

l Program

### File Attribute

* **Name** – only information kept in human-readable form
* **Identifier** – unique tag (number) identifies file within file system
* **Type** – needed for systems that support different types
* **Location** – pointer to file location on device
* **Size** – current file size
* **Protection** – controls who can do reading, writing, executing
* **Time, date, and user identification** – data for protection, security, and usage monitoring
* Information about files are kept in the directory structure, which is maintained on the disk

### File Types

**File Operations**

### Create, Write, Read, Reposition within file, Delete, Truncate

* *Open(Fi)* – search the directory structure on disk for entry *Fi*, and move the content of entry to memory
* *Close (Fi)* – move the content of entry *Fi* in memory to directory structure on disk

### File Access Methods

n **Sequential Access**

read next write next reset

no read after last write (rewrite)

n **Direct Access**

read *n*

write *n*

position to *n*

read next write next

rewrite *n*

*n* = relative block number



**Directory Structure**

### Single Level Directory

* A single directory for all users
* Naming problem
* Grouping problem



### Two Level Directory

* Separate directory for each user
* Path name
* Can have the same file name for different user
* Efficient searching
* No grouping capability



### Tree Structure Directory

* Efficient searching
* Grouping Capability



### Acyclic Graph Directories



* + - Have shared subdirectories and files

### File Sharing

* Sharing of files on multi-user systems is desirable
* Sharing may be done through a **protection** scheme
* On distributed systems, files may be shared across a network
* Network File System (NFS) is a common distributed file-sharing method
* **User IDs** identify users, allowing permissions and protections to be per-user
* **Group IDs** allow users to be in groups, permitting group access rights
* Uses networking to allow file system access between systems
	+ Manually via programs like FTP
	+ Automatically, seamlessly using **distributed file systems**
	+ Semi automatically via the **world wide web**
* **Client-server** model allows clients to mount remote file systems from servers
	+ Server can serve multiple clients
	+ Client and user-on-client identification is insecure or complicated
	+ **NFS** is standard UNIX client-server file sharing protocol
	+ **CIFS** is standard Windows protocol
	+ Standard operating system file calls are translated into remote calls
* Distributed Information Systems **(distributed naming services)** such as LDAP, DNS, NIS, Active Directory implement unified access to information needed for remote computing
* Remote file systems add new failure modes, due to network failure, server failure
* Recovery from failure can involve state information about status of each remote request
* Stateless protocols such as NFS include all information in each request, allowing easy recovery but less security
* **Consistency semantics** specify how multiple users are to access a shared file simultaneously
	+ Similar to Ch 7 process synchronization algorithms
		- Tend to be less complex due to disk I/O and network latency (for remote file systems
	+ Andrew File System (AFS) implemented complex remote file sharing semantics
	+ Unix file system (UFS) implements:
		- Writes to an open file visible immediately to other users of the same open file
		- Sharing file pointer to allow multiple users to read and write concurrently
	+ AFS has session semantics
		- Writes only visible to sessions starting after the file is closed

### File System Structure

* File structure
	+ Logical storage unit
	+ Collection of related information

n File system resides on secondary storage (disks)

n File system organized into layers

n **File control block** – storage structure consisting of information about a file

### Layered File System



**File Control Block**



### File Allocation Methods

An allocation method refers to how disk blocks are allocated for files:

### Contiguous Allocation

n Each file occupies a set of contiguous blocks on the disk

n Simple – only starting location (block #) and length (number of blocks) are required

n Random access

n Wasteful of space (dynamic storage-allocation problem)

n Files cannot grow



### Linked Allocation

n Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.

n Simple – need only starting address

n Free-space management system – no waste of space

n No random access

n Mapping



### Indexed Allocation



n Brings all pointers together into the *index block.*

n Need index table

n Random access

n Dynamic access without external fragmentation, but have overhead of index block.

# Secondary Storage Structure

### Magnetic Disk

* + Magnetic disks provide bulk of secondary storage of modern computers
		- Drives rotate at 60 to 200 times per second
		- **Transfer rate** is rate at which data flow between drive and computer
		- **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**)
		- **Head crash** results from disk head making contact with the disk surface
			* That’s bad
	+ Disks can be removable
	+ Drive attached to computer via **I/O bus**
		- Busses vary, including **EIDE, ATA, SATA, USB, Fibre Channel, SCSI**
		- **Host controller** in computer uses bus to talk to **disk controller** built into drive or storage array

### Magnetic Tap

* Was early secondary-storage medium
* Relatively permanent and holds large quantities of data
* Access time slow
* Random access ~1000 times slower than disk
* Mainly used for backup, storage of infrequently-used data, transfer medium between systems
* Kept in spool and wound or rewound past read-write head
* Once data under head, transfer rates comparable to disk
* 20-200GB typical storage

### Disk Structure

* Disk drives are addressed as large 1-dimensional arrays of *logical blocks*, where the logical block is the smallest unit of transfer.
* The 1-dimensional array of logical blocks is mapped into the sectors of the disk sequentially.
	+ Sector 0 is the first sector of the first track on the outermost cylinder.
	+ Mapping proceeds in order through that track, then the rest of the tracks in that cylinder, and then through the rest of the cylinders from outermost to innermost.

### Disk Scheduling

* The operating system is responsible for using hardware efficiently — for the disk drives, this means having a fast access time and disk bandwidth.
* Access time has two major components
	+ *Seek time* is the time for the disk are to move the heads to the cylinder containing the desired sector.
	+ *Rotational latency* is the additional time waiting for the disk to rotate the desired sector to the disk head.
* Minimize seek time
* Seek time  seek distance
* Disk bandwidth is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

### Disk Scheduling Algorithms FCFS

* This algorithm is intrinsically fair, but it generally does not provide the fastest service.

### SSTF (Shortest Seek Time First)

n Selects the request with the minimum seek time from the current head position.

n SSTF scheduling is a form of SJF scheduling; may cause starvation of some requests.



### SCAN

* The disk arm starts at one end of the disk, and moves toward the other end, servicing requests until it gets to the other end of the disk, where the head movement is reversed and servicing continues.
* Sometimes called the *elevator algorithm*.



### C-SCAN

* Provides a more uniform wait time than SCAN.
* The head moves from one end of the disk to the other. servicing requests as it goes. When it reaches the other end, however, it immediately returns to the beginning of the disk, without servicing any requests on the return trip.
* Treats the cylinders as a circular list that wraps around from the last cylinder to the first one

C-LOOK

* Version of C-SCAN
* Arm only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.



### Disk Management

* *Low-level formatting*, or *physical formatting* — Dividing a disk into sectors that the disk controller can read and write.
* To use a disk to hold files, the operating system still needs to record its own data structures on the disk.
	+ *Partition* the disk into one or more groups of cylinders.
	+ *Logical formatting* or “making a file system”.
* Boot block initializes system.
	+ The bootstrap is stored in ROM.
	+ *Bootstrap loader* program.
* Methods such as *sector sparing* used to handle bad blocks.
* The controller can be told to replace each bad sector logically with one of the spare sectors. This scheme is known as **sector sparing** or **forwarding**.

### Swap Space Management

* Swap-space — Virtual memory uses disk space as an extension of main memory.
* Swap-space can be carved out of the normal file system or, more commonly, it can be in a separate disk partition.
* A swap space can reside in one of two places: it can be carved out of the normal file system, or it can be in a separate disk partition.
* If the swap space is simply a large file within the file system, normal file-system routines can be used to create it, name it, and allocate its space.
* Alternatively, swap space can be created in a separate **raw partition**. No file system or directory structure is placed in this space.
* A separate swap-space storage manager is used to allocate and deallocate the blocks from the raw partition.

# I/O Systems

### I/O Hardware

* A device communicates with a computer system by sending signals over a cable or even through the air. The device communicates with the machine via a connection point, or **port**—for example, a serial port.
* If devices share a common set of wires, the connection is called a bus.
* A **bus** is a set of wires and a rigidly defined protocol that specifies a set of messages that can be sent on the wires.
* When device *A* has a cable that plugs into device *B,* and device *B* has a cable that plugs into device *C,* and device *C* plugs into a port on the computer, this arrangement is called a **daisy chain**. A daisy chain usually operates as a bus.
* A **PCI bus** (the common PC system bus) connects the processor–memory subsystem to fast devices, and an **expansion bus** connects relatively slow devices, such as the keyboard and serial and USB ports.
* Disks are connected together on a **Small Computer System Interface (SCSI)**

bus plugged into a SCSI controller.

* A **controller** is a collection of electronics that can operate a port, a bus, or a device.
* A serial-port controller is a simple device controller. It is a single chip (or portion of a chip) in the computer that controls the signals on the wires of a serial port.
* But the SCSI protocol is complex, the SCSI bus controller is often implemented as a separate circuit board (or a **host adapter**) that plugs into the computer. It typically contains a processor, microcode, and some private memory to enable it to process the SCSI protocol messages.

### Polling

* Determines state of device
	+ command-ready
	+ busy
	+ Error
* **Busy-wait** cycle to wait for I/O from device

### Interrupt

* CPU **Interrupt-request line** triggered by I/O device
* **Interrupt handler** receives interrupts
* **Maskable** to ignore or delay some interrupts
* Interrupt vector to dispatch interrupt to correct handler
* Most CPUs have two interrupt request lines. One is the **nonmaskable interrupt**, which is reserved for events such as unrecoverable memory errors.
* The second interrupt line is **maskable**: it can be turned off by the CPU before the execution of critical instruction sequences that must not be interrupted.
* The maskable interrupt is used by device controllers to request service.
* Interrupt mechanism also used for exceptions



### Direct Memory Access

* Used to avoid **programmed I/O** for large data movement
* Requires **DMA** controller
* Bypasses CPU to transfer data directly between I/O device and memory

### Application I/O Interface

* I/O system calls encapsulate device behaviors in generic classes
* Device-driver layer hides differences among I/O controllers from kernel
* Devices vary in many dimensions

### Character-stream or block

* + **Sequential or random-access**

### Sharable or dedicated

* + **Speed of operation**

### read-write, read only, or write only Kernel I/O Structure

**Characteristics of I/O Devices**

### Block and Character Devices

* Block devices include disk drives
	+ Commands include read, write, seek
	+ Raw I/O or file-system access
	+ Memory-mapped file access possible
* Character devices include keyboards, mice, serial ports
	+ Commands include get, put
	+ Libraries layered on top allow line editing

### Network Devices

* Varying enough from block and character to have own interface
* Unix and Windows NT/9*x*/2000 include socket interface
	+ Separates network protocol from network operation
	+ Includes select functionality
* Approaches vary widely (pipes, FIFOs, streams, queues, mailboxes)

### Clock and Timers

* Provide current time, elapsed time, timer
* **Programmable interval timer** used for timings, periodic interrupts

### Blocking and Non-blocking I/O

* **Blocking** - process suspended until I/O completed
	+ Easy to use and understand
	+ Insufficient for some needs
* **Nonblocking** - I/O call returns as much as available
	+ User interface, data copy (buffered I/O)
	+ Implemented via multi-threading
	+ Returns quickly with count of bytes read or written
* **Asynchronous** - process runs while I/O executes
	+ Difficult to use
	+ I/O subsystem signals process when I/O completed

### Kernel I/O Subsystem

* **Scheduling**
	+ Some I/O request ordering via per-device queue
	+ Some OSs try fairness
* **Buffering -** store data in memory while transferring between devices
	+ To cope with device speed mismatch
	+ To cope with device transfer size mismatch
	+ To maintain “copy semantics”
* **Caching** - fast memory holding copy of data
	+ Always just a copy
	+ Key to performance
* **Spooling** - hold output for a device
	+ If device can serve only one request at a time
	+ i.e., Printing
* **Device reservation** - provides exclusive access to a device
	+ System calls for allocation and deallocation

l Watch out for deadlock