UNIT : V : Storage management

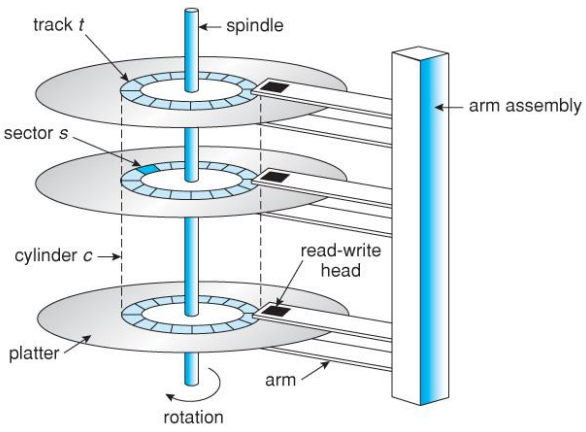
Overview of Mass-Storage

Overview of Mass-Storage Structure

Magnetic Disks

Traditional magnetic disks have the following basic structure:

- One or more platters in the form of disks covered with magnetic media. Hard disk platters are made of rigid metal, while "floppy" disks are made of more flexible plastic.
- Each platter has two working surfaces. Older hard disk drives would sometimes not use the very top or bottom surface of a stack of platters, as these surfaces were more susceptible to potential damage.
- Each working surface is divided into a number of concentric rings called tracks. The collection of all tracks that are the same distance from the edge of the platter, (i.e. all tracks immediately above one another in the following diagram) is called a cylinder.
- The data on a hard drive is read by read-write heads. The standard configuration uses one head per surface, each on a separate arm, and controlled by a common arm assembly which moves all heads simultaneously from one cylinder to another. (Other configurations, including independent read-write heads, may speed up disk access, but involve serious technical difficulties.)



- In operation the disk rotates at high speed, such as 7200 rpm (120 revolutions per second.) The rate at which data can be transferred from the disk to the computer is composed of several steps:

- The positioning time, a.k.a. the seek time or random access time is the time required to move the heads from one cylinder to another, and for the heads to settle down after the move. This is typically the slowest step in the process and the predominant bottleneck to overall transfer rates.
- The rotational latency is the amount of time required for the desired sector to rotate around and come under the read-write head.This can range anywhere from zero to one full revolution, and on the average will equal one-half revolution. This is another physical step and is usually the second slowest step behind seek time. (For a disk rotating at 7200 rpm, the average rotational latency would be 1/2 revolution / 120 revolutions per second, or just over 4 milliseconds, a long time by computer standards.

- The host controller is at the computer end of the I/O bus, and the disk controller is built into the disk itself. The CPU issues commands to the host controller via I/O ports. Data is transferred between the magnetic surface and onboard cache by the disk controller, and then the data is transferred from that cache to the host controller and the motherboard memory at electronic speeds.

Solid-State Disks - New

- As technologies improve and economics change, old technologies are often used in different ways. One example of this is the increasing used of solid state disks, or SSDs.
- SSDs use memory technology as a small fast hard disk. Specific implementations may use either flash memory or DRAM chips protected by a battery to sustain the information through power cycles.
- Because SSDs have no moving parts they are much faster than traditional hard drives, and certain problems such as the scheduling of disk accesses simply do not apply.
- However SSDs also have their weaknesses: They are more expensive than hard drives, generally not as large, and may have shorter life spans.
- SSDs are especially useful as a high-speed cache of hard-disk information that must be accessed quickly. One example is to store filesystem meta-data, e.g. directory and inode information, that must be accessed quickly and often. Another variation is a boot disk containing the OS and some application executables, but no vital user data. SSDs are also used in laptops to make them smaller, faster, and lighter.
- Because SSDs are so much faster than traditional hard disks, the throughput of the bus can become a limiting factor, causing some SSDs to be connected directly to the system PCI bus for example.

Magnetic Tapes

- Magnetic tapes were once used for common secondary storage before the days of hard disk drives, but today are used primarily for backups.
- Accessing a particular spot on a magnetic tape can be slow, but once reading or writing commences, access speeds are comparable to disk drives.
- Capacities of tape drives can range from 20 to 200 GB, and compression can double that capacity.

Disk Structure

- The traditional head-sector-cylinder, HSC numbers are mapped to linear block addresses by numbering the first sector on the first head on the outermost track as sector 0. Numbering proceeds with the rest of the sectors on that same track, and then the rest of the tracks on the same cylinder before proceeding through the rest of the cylinders to the center of the disk. In modern practice these linear block addresses are used in place of the HSC numbers for a variety of reasons:

1. The linear length of tracks near the outer edge of the disk is much longer than for those tracks located near the center, and therefore it is possible to squeeze many more sectors onto outer tracks than onto inner ones.
2. All disks have some bad sectors, and therefore disks maintain a few spare sectors that can be used in place of the bad ones. The mapping of spare sectors to bad sectors in managed internally to the disk controller.
3. Modern hard drives can have thousands of cylinders, and hundreds of sectors per track on their outermost tracks. These numbers exceed the range of HSC numbers for many (older) operating systems, and therefore disks can be configured for any convenient combination of HSC values that falls within the total number of sectors physically on the drive.

- There is a limit to how closely packed individual bits can be placed on a physical media, but that limit is growing increasingly more packed as technological advances are made.
- Modern disks pack many more sectors into outer cylinders than inner ones, using one of two approaches:
  - With Constant Linear Velocity, CLV, the density of bits is uniform from cylinder to cylinder. Because there are more sectors in outer cylinders, the disk spins slower when reading those cylinders, causing the rate of bits passing under the read-write head to remain constant. This is the approach used by modern CDs and DVDs.
  - With Constant Angular Velocity, CAV, the disk rotates at a constant angular speed, with the bit density decreasing on outer cylinders. (These disks would have a constant number of sectors per track on all cylinders.)

Disk Attachment

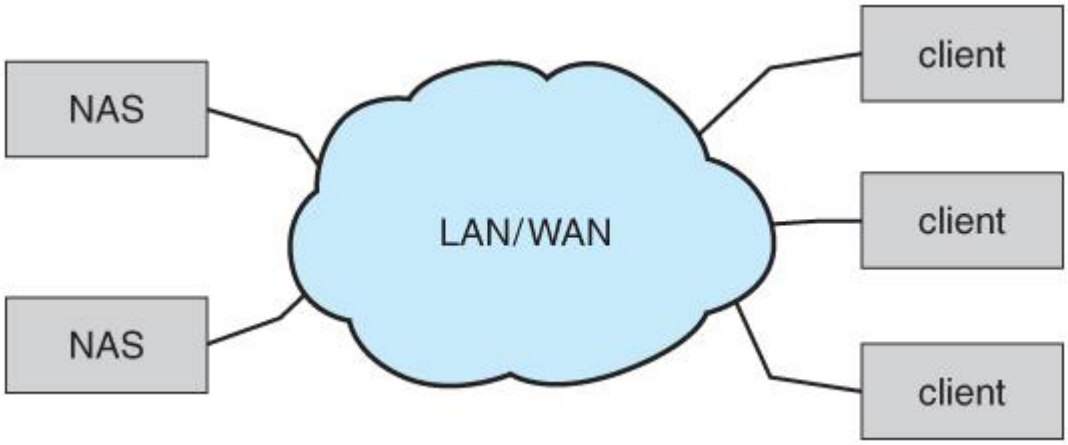
Disk drives can be attached either directly to a particular host (a local disk) or to a network.

Host-Attached Storage

- Local disks are accessed through I/O Ports as described earlier.
- The most common interfaces are IDE or ATA, each of which allow up to two drives per host controller.
- SATA is similar with simpler cabling.
- High end workstations or other systems in need of larger number of disks typically use SCSI disks:
  - The SCSI standard supports up to 16 targets on each SCSI bus, one of which is generally the host adapter and the other 15 of which can be disk or tape drives.
  - A SCSI target is usually a single drive, but the standard also supports up to 8 units within each target. These would generally be used for accessing individual disks within a RAID array. (See below.)
  - The SCSI standard also supports multiple host adapters in a single computer, i.e. multiple SCSI busses.
  - Modern advancements in SCSI include "fast" and "wide" versions, as well as SCSI-2.
  - A large switched fabric having a 24-bit address space. This variant allows for multiple devices and multiple hosts to interconnect, forming the basis for the storage-area networks, SANs, to be discussed in a future section.

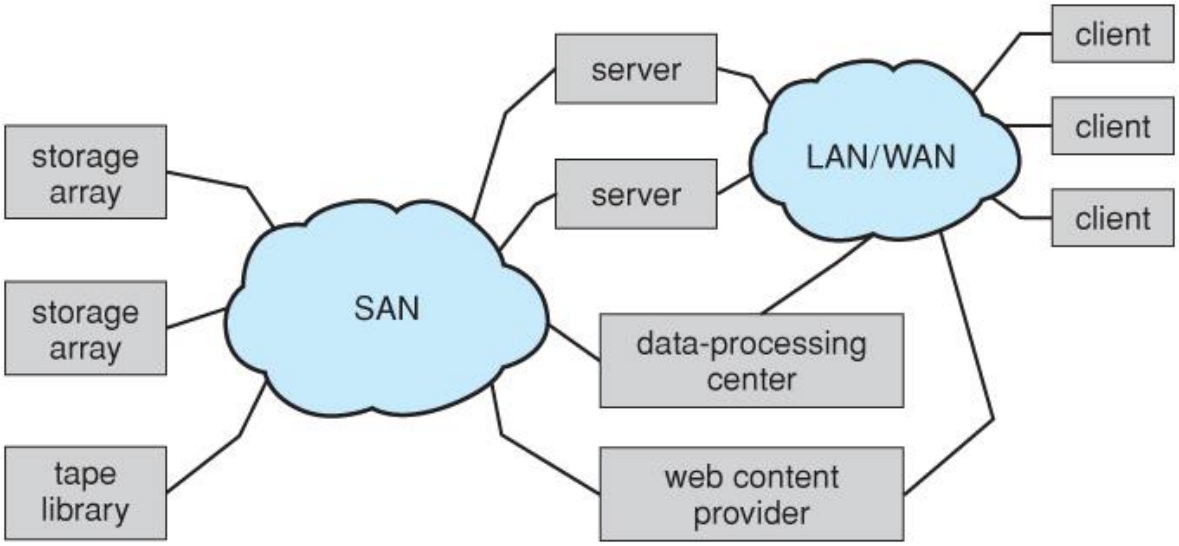
Network-Attached Storage

- Network attached storage connects storage devices to computers using a remote procedure call, RPC, interface, typically with something like NFS filesystem mounts. This is convenient for allowing several computers in a group common access and naming conventions for shared storage.
- NAS can be implemented using SCSI cabling, or ISCSI uses Internet protocols and standard network connections, allowing long-distance remote access to shared files.
- NAS allows computers to easily share data storage, but tends to be less efficient than standard host-attached storage.



Storage-Area Network

- A Storage-Area Network, SAN, connects computers and storage devices in a network, using storage protocols instead of network protocols.
- One advantage of this is that storage access does not tie up regular networking bandwidth.
- SAN is very flexible and dynamic, allowing hosts and devices to attach and detach on the fly.
- SAN is also controllable, allowing restricted access to certain hosts and devices.



Storage-area network.

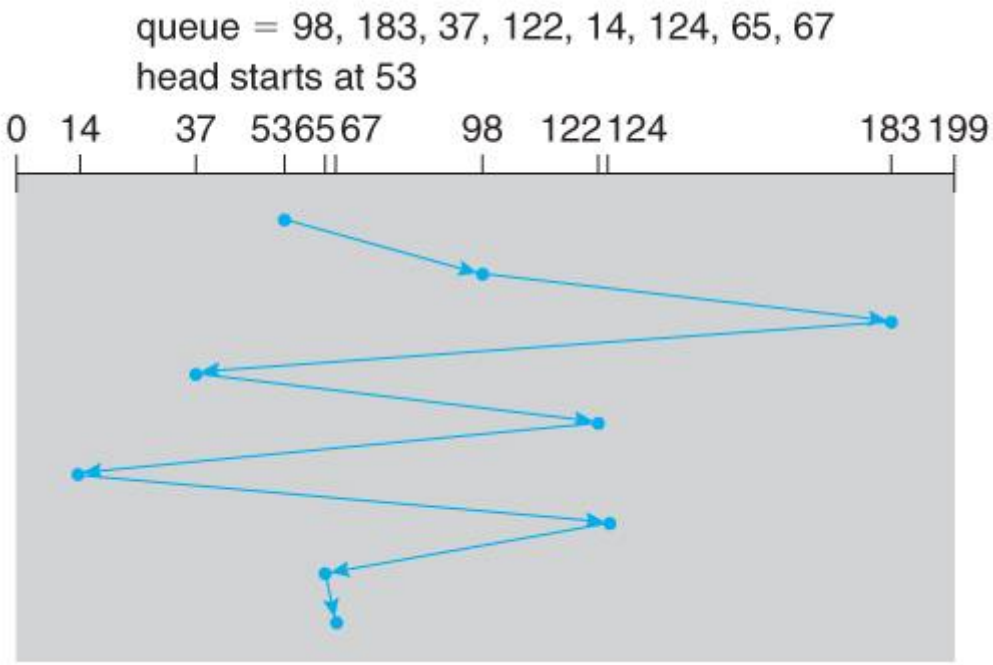
Disk Scheduling

- As mentioned earlier, disk transfer speeds are limited primarily by seek times and rotational latency. When multiple requests are to be processed there is also some inherent delay in waiting for other requests to be processed.

- Bandwidth is measured by the amount of data transferred divided by the total amount of time from the first request being made to the last transfer being completed, (for a series of disk requests.)
- Both bandwidth and access time can be improved by processing requests in a good order.
- Disk requests include the disk address, memory address, number of sectors to transfer, and whether the request is for reading or writing.

FCFS Scheduling

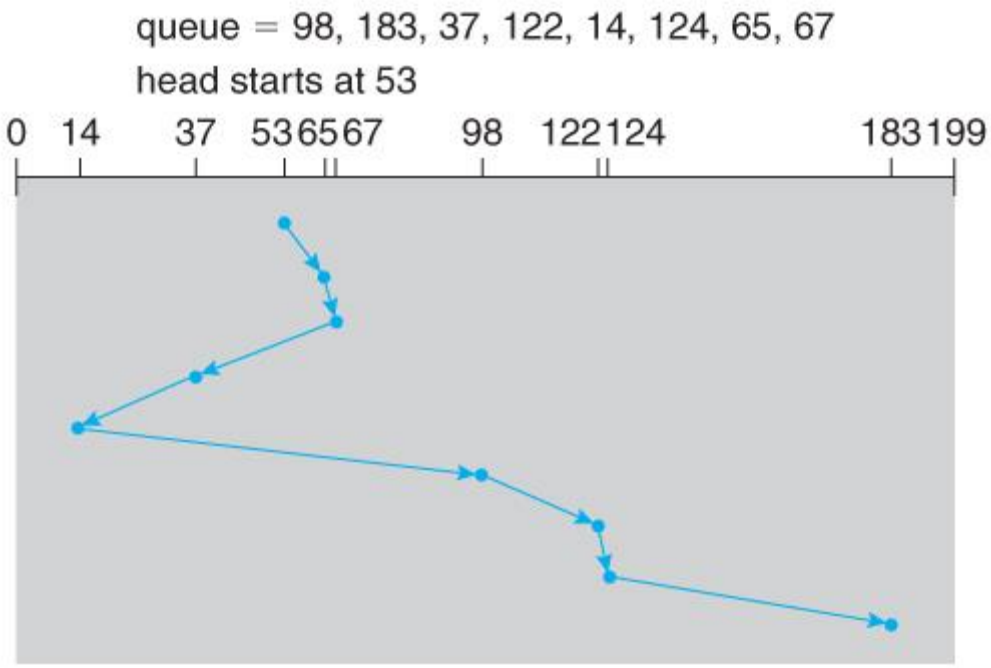
- First-Come First-Serve is simple and intrinsically fair, but not very efficient. Consider in the following sequence the wild swing from cylinder 122 to 14 and then back to 124:



FCFS disk scheduling.

SSTF Scheduling

- Shortest Seek Time First scheduling is more efficient, but may lead to starvation if a constant stream of requests arrives for the same general area of the disk.
- SSTF reduces the total head movement to 236 cylinders, down from 640 required for the same set of requests under FCFS. Note, however that the distance could be reduced still further to 208 by starting with 37 and then 14 first before processing the rest of the requests.



SSTF disk scheduling.

SCAN Scheduling

- The SCAN algorithm, a.k.a. the elevator algorithm moves back and forth from one end of the disk to the other, similarly to an elevator processing requests in a tall building.

SCAN disk scheduling.

- Under the SCAN algorithm, If a request arrives just ahead of the moving head then it will be processed right away, but if it arrives just after the head has passed, then it will have to wait for the head to pass going the other way on the return trip. This leads to a fairly wide variation in access times which can be improved upon.

- The Circular-SCAN algorithm improves upon SCAN by treating all requests in a circular queue fashion - Once the head reaches the end of the disk, it returns to the other end without processing any requests, and then starts again from the beginning of the disk:

C-SCAN disk scheduling.

LOOK Scheduling

- LOOK scheduling improves upon SCAN by looking ahead at the queue of pending requests, and not moving the heads any farther towards the end of the disk than is necessary. The following diagram illustrates the circular form of LOOK:

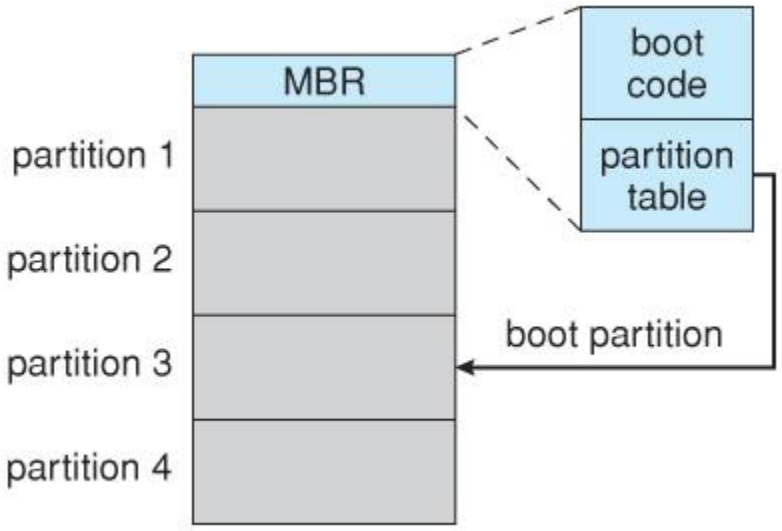
Disk Formatting

- Before a disk can be used, it has to be low-level formatted, which means laying down all of the headers and trailers marking the beginning and ends of each sector. Included in the header and trailer are the linear sector numbers, and error-correcting codes, ECC, which allow damaged sectors to not only be detected, but in many cases for the damaged data to be recovered (depending on the extent of the damage.) Sector sizes are traditionally 512 bytes, but may be larger, particularly in larger drives.
- ECC calculation is performed with every disk read or write, and if damage is detected but the data is recoverable, then a soft error has occurred. Soft errors are generally handled by the on-board disk controller, and never seen by the OS.

Boot Block

- Computer ROM contains a bootstrap program (OS independent) with just enough code to find the first sector on the first hard drive on the first controller, load that sector into memory, and transfer control over to it. (The ROM bootstrap program may look in floppy and/or CD drives before accessing the hard drive, and is smart enough to recognize whether it has found valid boot code or not.)
- The first sector on the hard drive is known as the Master Boot Record, MBR, and contains a very small amount of code in addition to the partition table. The partition table documents how the disk is partitioned into logical disks, and indicates specifically which partition is the active or boot partition.
- The boot program then looks to the active partition to find an operating system, possibly loading up a slightly larger / more advanced boot program along the way.
- In a dual-boot (or larger multi-boot) system, the user may be given a choice of which operating system to boot, with a default action to be taken in the event of no response within some time frame.
- Once the kernel is found by the boot program, it is loaded into memory and then control is transferred over to the OS. The kernel will normally continue the boot process by initializing all important kernel data structures, launching important system services (e.g. network daemons,

sched, init, etc.), and finally providing one or more login prompts. Boot options at this stage may include single-user a.k.a. maintenance or safe modes, in which very few system services are started - These modes are designed for system administrators to repair problems or otherwise maintain the system.



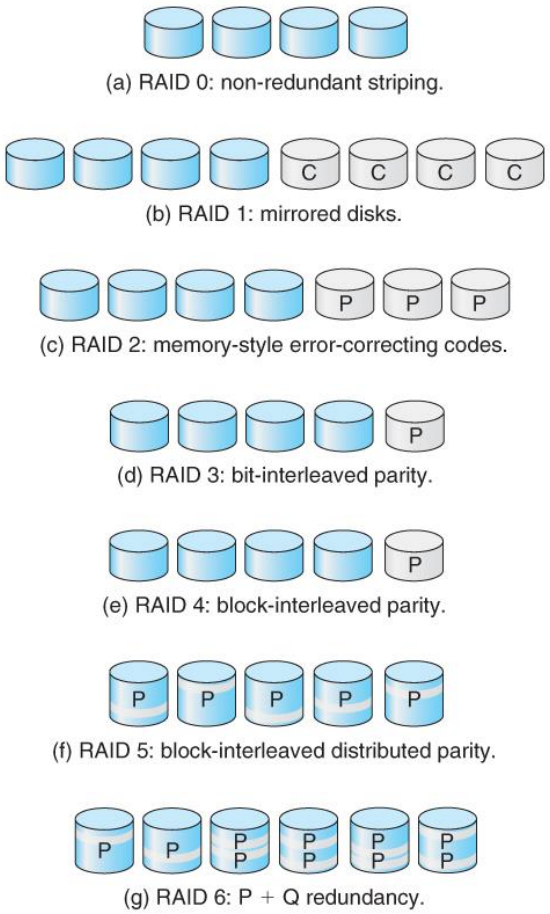
Booting from disk in Windows 2000.

Bad Blocks

- No disk can be manufactured to 100% perfection, and all physical objects wear out over time. For these reasons all disks are shipped with a few bad blocks, and additional blocks can be expected to go bad slowly over time. If a large number of blocks go bad then the entire disk will need to be replaced, but a few here and there can be handled through other means.
- Modern disk controllers make much better use of the error-correcting codes, so that bad blocks can be detected earlier and the data usually recovered. (Recall that blocks are tested with every write as well as with every read, so often errors can be detected before the write operation is complete, and the data simply written to a different sector instead.)
- If the data on a bad block cannot be recovered, then a hard error has occurred., which requires replacing the file(s) from backups, or rebuilding them from scratch.

Swap-Space Management

- Modern systems typically swap out pages as needed, rather than swapping out entire processes. Hence the swapping system is part of the virtual memory management system.
- Managing swap space is obviously an important task for modern OSes.



(a) RAID 0: non-redundant striping.

(b) RAID 1: mirrored disks.

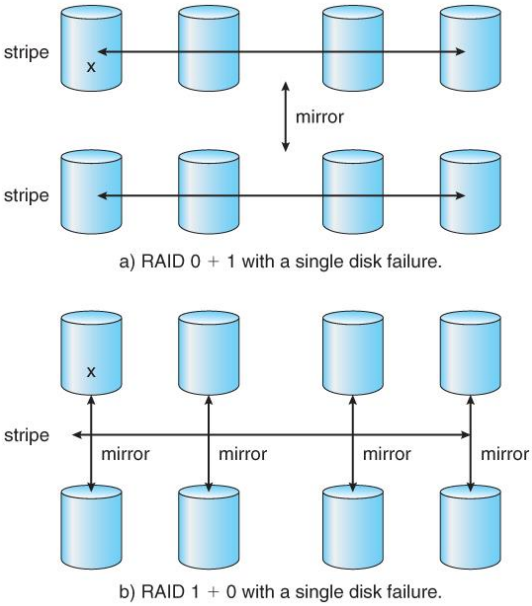
(c) RAID 2: memory-style error-correcting codes.

(d) RAID 3: bit-interleaved parity.

(e) RAID 4: block-interleaved parity.

(f) RAID 5: block-interleaved distributed parity.

(g) RAID 6: P + Q redundancy.



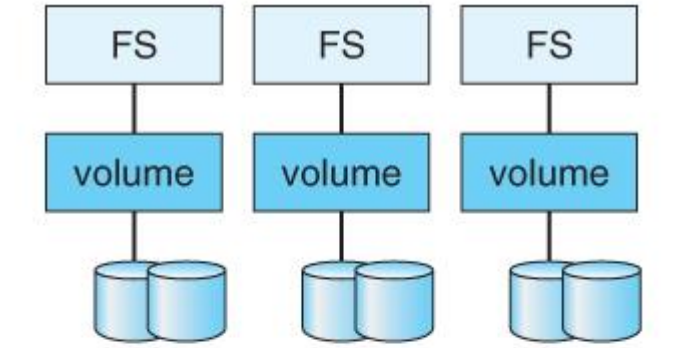
a) RAID 0 + 1 with a single disk failure.

b) RAID 1 + 0 with a single disk failure.

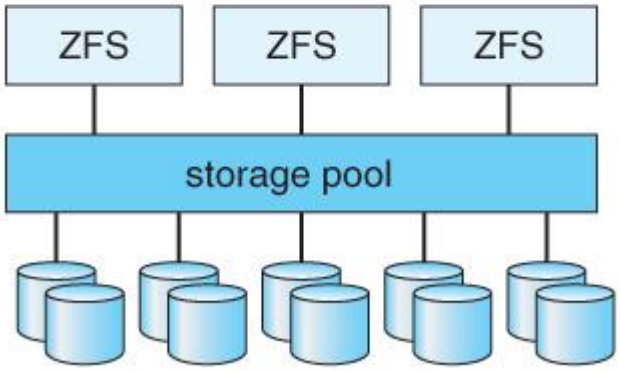
RAID 0 + 1 and 1 + 0

Extensions

- RAID concepts have been extended to tape drives (e.g. striping tapes for faster backups or parity checking tapes for reliability), and for broadcasting of data.



(a) Traditional volumes and file systems.



(b) ZFS and pooled storage.

Traditional volumes and file systems. (b) a ZFS pool and file systems.

Stable-Storage Implementation

The concept of stable storage (first presented in chapter 6) involves a storage medium in which data is never lost, even in the face of equipment failure in the middle of a write operation.

- To implement this requires two (or more) copies of the data, with separate failure modes.
- An attempted disk write results in one of three possible outcomes:

1. The data is successfully and completely written.
2. The data is partially written, but not completely. The last block written may be garbled.
3. No writing takes place at all.

12.9.1.1 Removable Disks

- Removable magnetic disks (e.g. floppies) can be nearly as fast as hard drives, but are at greater risk for damage due to scratches. Variations of removable magnetic disks up to a GB or more in capacity have been developed. (Hot-swappable hard drives?)
- A magneto-optical disk uses a magnetic disk covered in a clear plastic coating that protects the surface.
  - The heads sit a considerable distance away from the magnetic surface, and as a result do not have enough magnetic strength to switch bits at normal room temperature.

Tapes

- Tape drives typically cost more than disk drives, but the cost per MB of the tapes themselves is lower.
- Tapes are typically used today for backups, and for enormous volumes of data stored by certain scientific establishments. (E.g. NASA's archive of space probe and satellite imagery, which is currently being downloaded from numerous sources faster than anyone can actually look at it.)
- Robotic tape changers move tapes from drives to archival tape libraries upon demand.
- (Never underestimate the bandwidth of a station wagon full of tapes rolling down the highway!)

Future Technology

- Solid State Disks, SSDs, are becoming more and more popular.
- Holographic storage uses laser light to store images in a 3-D structure, and the entire data structure can be transferred in a single flash of laser light.
- Micro-Electronic Mechanical Systems, MEMS, employs the technology used for computer chip fabrication to create VERY tiny little machines. One example packs 10,000 read-write heads within a square centimeter of space, and as media are passed over it, all 10,000 heads can read data in parallel.

Operating-System Support

- The OS must provide support for tertiary storage as removable media, including the support to transfer data between different systems.

Application Interface

- File systems are typically not stored on tapes. (It might be technically possible, but it is impractical.)
- Tapes are also not low-level formatted, and do not use fixed-length blocks. Rather data is written to tapes in variable length blocks as needed.
- Tapes are normally accessed as raw devices, requiring each application to determine how the data is to be stored and read back. Issues such as header contents and ASCII versus binary encoding (and byte-ordering) are generally application specific.

File Naming

- File naming conventions for removable media are not entirely uniquely specific, nor are they necessarily consistent between different systems. (Two removable disks may contain files with the same name, and there is no clear way for the naming system to distinguish between them.)
- Fortunately music CDs have a common format, readable by all systems. Data CDs and DVDs have only a few format choices, making it easy for a system to support all known formats.

Hierarchical Storage Management

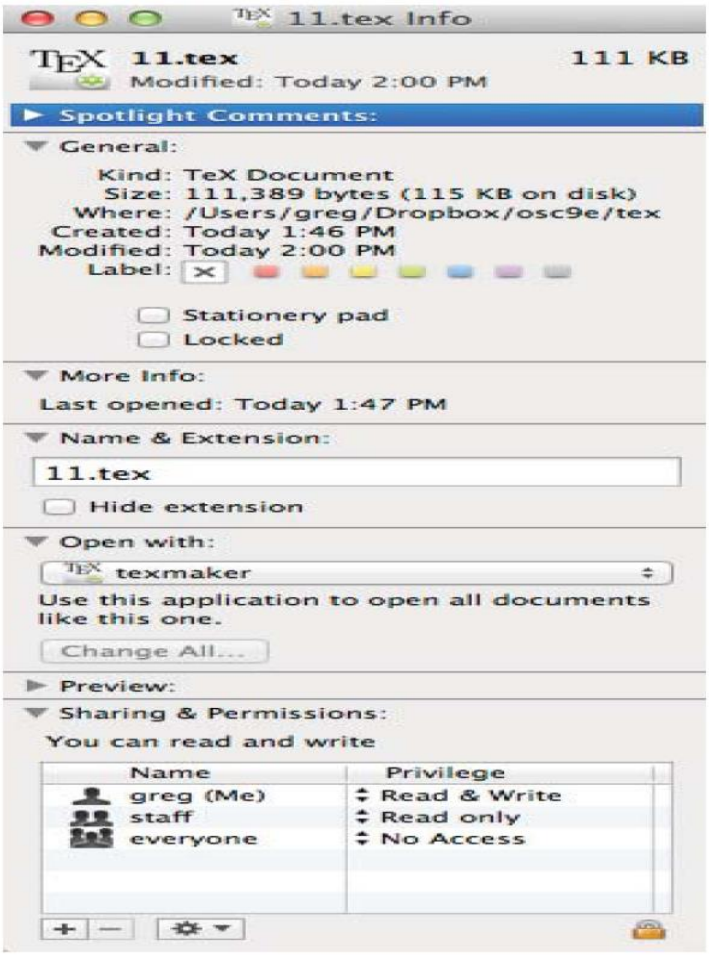
- Hierarchical storage involves extending file systems out onto tertiary storage, swapping files from hard drives to tapes in much the same manner as data blocks are swapped from memory to hard drives.
- A placeholder is generally left on the hard drive, storing information about the particular tape (or other removable media) on which the file has been swapped out to.
- A robotic system transfers data to and from tertiary storage as needed, generally automatically upon demand of the file(s) involved.

File Concept

File Attributes

- Different OS keep track of different file attributes, including:
  - **Name** - Some systems give special significance to names, and particularly extensions (.exe, .txt, etc.), and some do not. Some extensions may be of significance to the OS (.exe), and others only to certain applications (.jpg)
  - **Identifier** (e.g. inode number)
  - **Type** - Text, executable, other binary, etc.

- **Location** - on the hard drive.
- **Size**
- **Protection**
- **Time & Date**
- **User ID**



File Operations

- The file ADT supports many common operations:
  - Creating a file
  - Writing a file
  - Reading a file
  - Repositioning within a file
  - Deleting a file
  - Truncating a file.
- Most OSes require that files be ***opened*** before access and ***closed*** after all access is complete. Normally the programmer must open and close files explicitly, but some rare systems open the file automatically at first access. Information about currently open files is stored in an ***open file table***, containing for example:
  - **File pointer** - records the current position in the file, for the next read or write access.

- **File-open count** - How many times has the current file been opened (simultaneously by different processes) and not yet closed? When this counter reaches zero the file can be removed from the table.
- **Disk location of the file.**
- **Access rights**
- Some systems provide support for ***file locking.***
- A ***shared lock*** is for reading only.
- A ***exclusive lock*** is for writing as well as reading.
- An ***advisory lock*** is informational only, and not enforced. (A "Keep Out" sign, which may be ignored.)
- A ***mandatory lock*** is enforced. (A truly locked door.)
- UNIX used advisory locks, and Windows uses mandatory locks.

File-locking example in Java.

File Types

- Windows (and some other systems) use special file extensions to indicate the type of each file:

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

Common file types.

- Macintosh stores a creator attribute for each file, according to the program that first created it with the create( ) system call.
- UNIX stores magic numbers at the beginning of certain files. (Experiment with the "file" command, especially in directories such as /bin and /dev)

File Structure

- Some files contain an internal structure, which may or may not be known to the OS.
- For the OS to support particular file formats increases the size and complexity of the OS.
- UNIX treats all files as sequences of bytes, with no further consideration of the internal structure. (With the exception of executable binary programs, which it must know how to load and find the first executable statement, etc.)
- Macintosh files have two ***forks*** - a ***resource fork***, and a ***data fork***. The resource fork contains information relating to the UI, such as icons and button images, and can be modified independently of the data fork, which contains the code or data as appropriate.

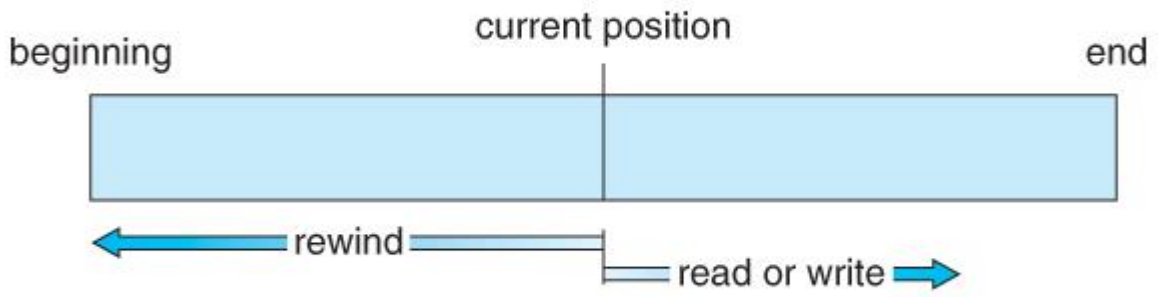
Internal File Structure

- Disk files are accessed in units of physical blocks, typically 512 bytes or some power-of-two multiple thereof. (Larger physical disks use larger block sizes, to keep the range of block numbers within the range of a 32-bit integer.)
- Internally files are organized in units of logical units, which may be as small as a single byte, or may be a larger size corresponding to some data record or structure size.
- The number of logical units which fit into one physical block determines its ***packing***, and has an impact on the amount of internal fragmentation (wasted space) that occurs.
- As a general rule, half a physical block is wasted for each file, and the larger the block sizes the more space is lost to internal fragmentation.

Access Methods

Sequential Access

- A sequential access file emulates magnetic tape operation, and generally supports a few operations:
  - read next - read a record and advance the tape to the next position.
  - write next - write a record and advance the tape to the next position.
  - rewind
  - skip n records - May or may not be supported. N may be limited to positive numbers, or may be limited to +/- 1.



Sequential-access file.

Direct Access

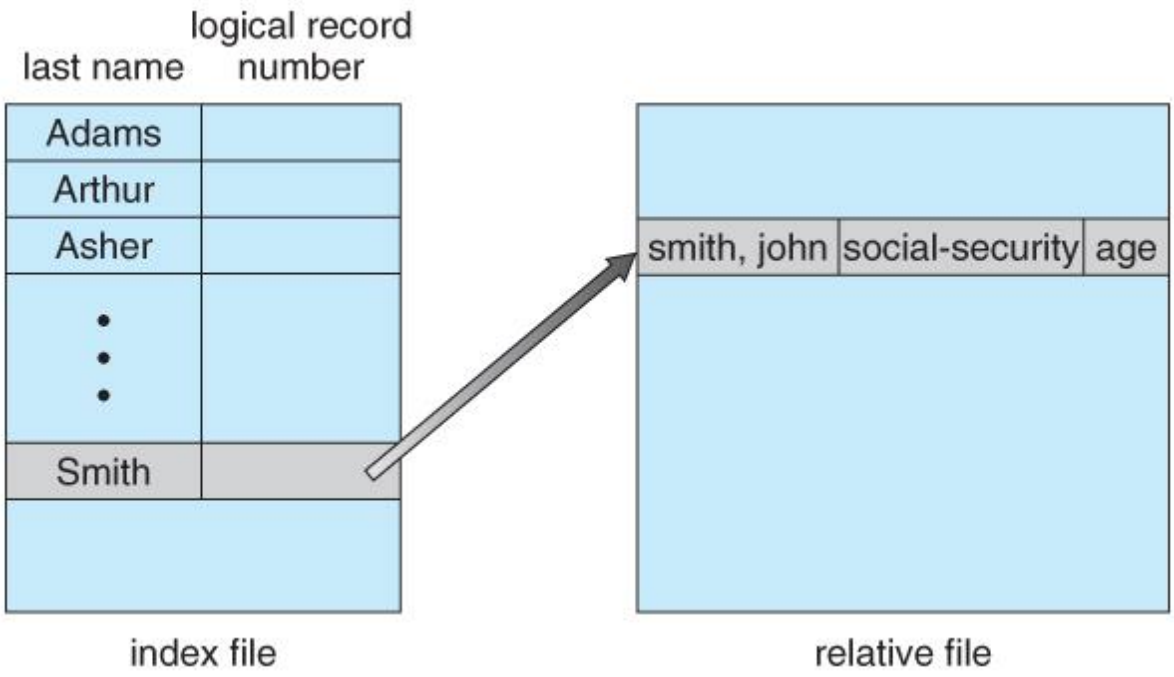
- Jump to any record and read that record. Operations supported include:
  - read n - read record number n. (Note an argument is now required.)
  - write n - write record number n. (Note an argument is now required.)
  - jump to record n - could be 0 or the end of file.
  - Query current record - used to return back to this record later.
  - Sequential access can be easily emulated using direct access. The inverse is complicated and inefficient.

sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;
write_next	write cp; cp = cp + 1;

Simulation of sequential access on a direct-access file.

Other Access Methods

- An indexed access scheme can be easily built on top of a direct access system. Very large files may require a multi-tiered indexing scheme, i.e. indexes of indexes.

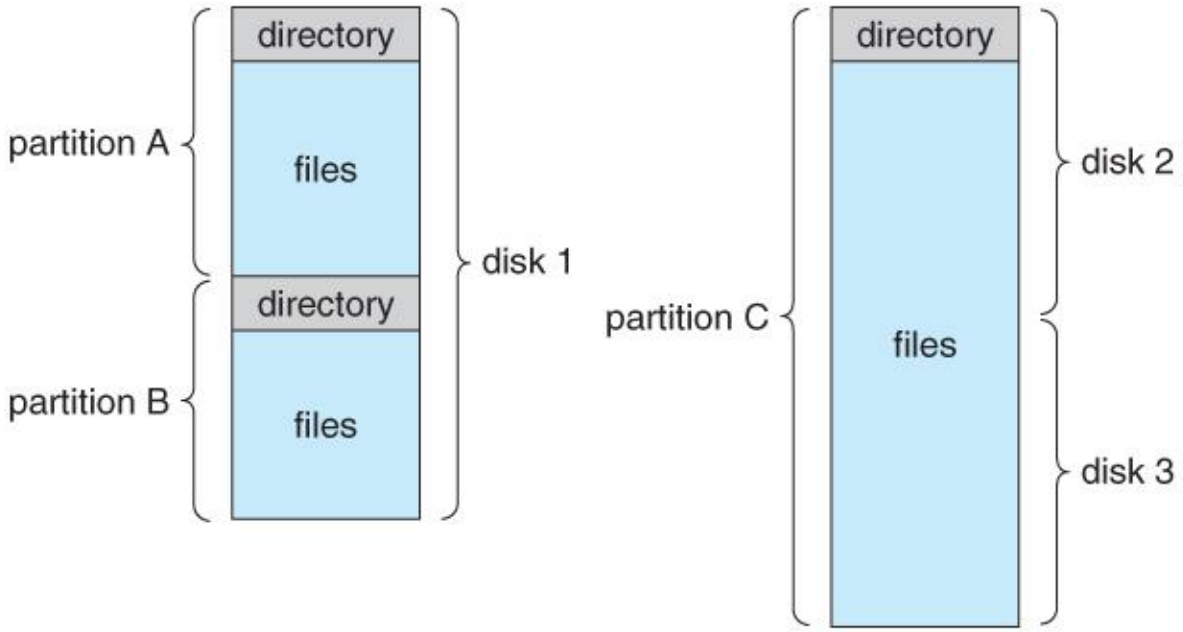


Example of index and relative files.

Directory Structure

Storage Structure

- A disk can be used in its entirety for a file system.
- Alternatively a physical disk can be broken up into multiple ***partitions, slices, or mini-disks***, each of which becomes a virtual disk and can have its own filesystem. (or be used for raw storage, swap space, etc.)
- Or, multiple physical disks can be combined into one ***volume***, i.e. a larger virtual disk, with its own filesystem spanning the physical disks.

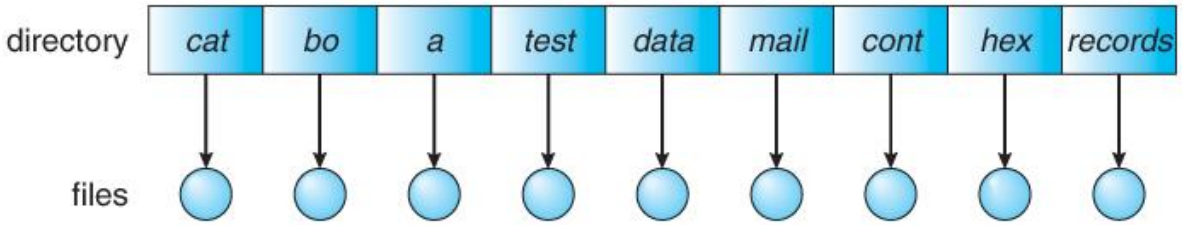


Directory Overview

- Directory operations to be supported include:
  - Search for a file
  - Create a file - add to the directory
  - Delete a file - erase from the directory
  - List a directory - possibly ordered in different ways.
  - Rename a file - may change sorting order
  - Traverse the file system.

Single-Level Directory

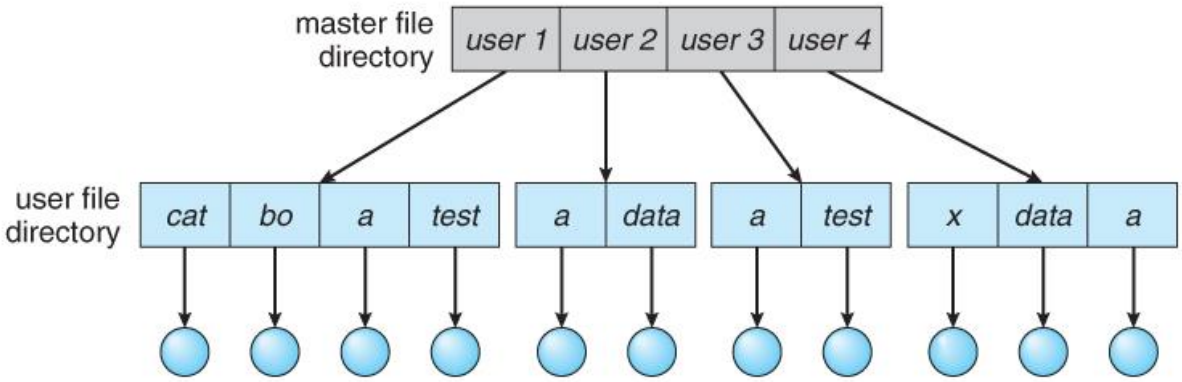
- Simple to implement, but each file must have a unique name.



Single-level directory.

Two-Level Directory

- Each user gets their own directory space.
- File names only need to be unique within a given user's directory.
- A master file directory is used to keep track of each users directory, and must be maintained when users are added to or removed from the system.
- A separate directory is generally needed for system (executable) files.
- Systems may or may not allow users to access other directories besides their own
  - If access to other directories is allowed, then provision must be made to specify the directory being accessed.
  - If access is denied, then special consideration must be made for users to run programs located in system directories. A ***search path*** is the list of directories in which to search for executable programs, and can be set uniquely for each user.

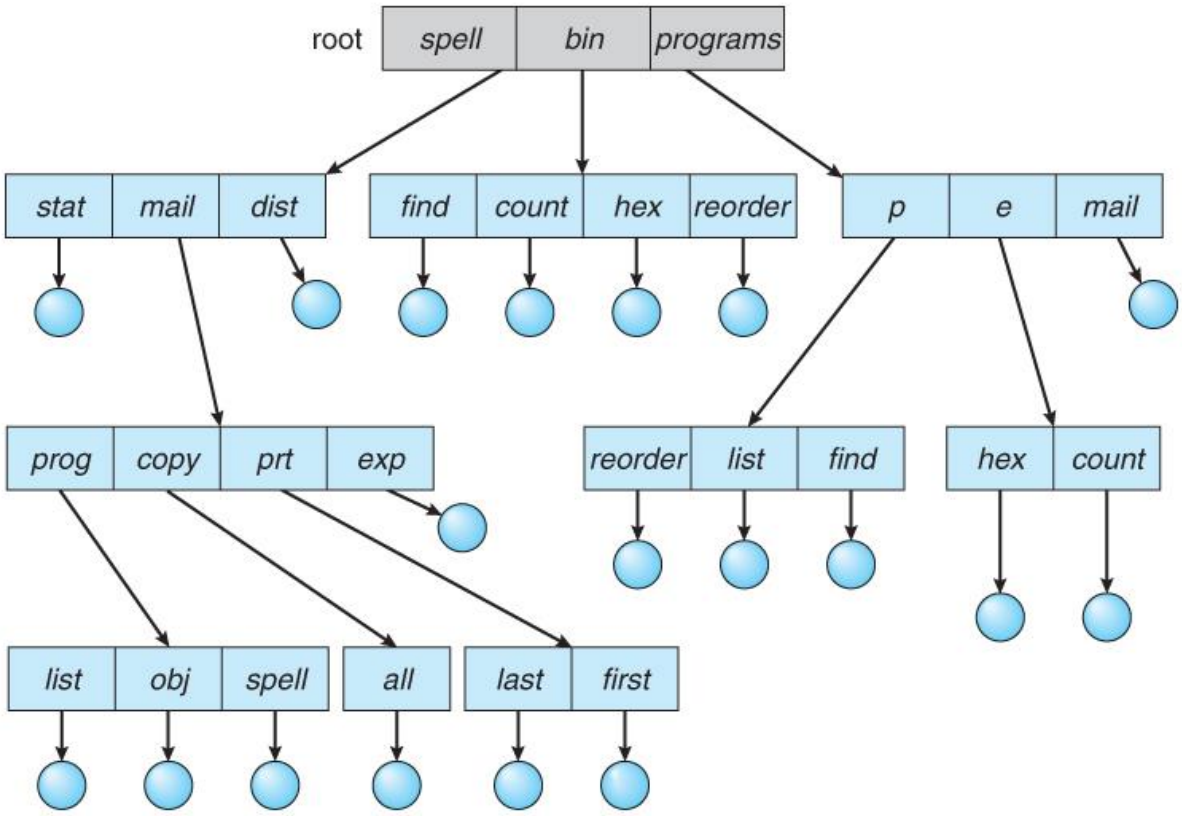


Two-level directory structure.

Tree-Structured Directories

- An obvious extension to the two-tiered directory structure, and the one with which we are all most familiar.
- Each user / process has the concept of a ***current directory*** from which all (relative) searches take place.
- Files may be accessed using either absolute pathnames (relative to the root of the tree) or relative pathnames (relative to the current directory.)
- Directories are stored the same as any other file in the system, except there is a bit that identifies them as directories, and they have some special structure that the OS understands.

- One question for consideration is whether or not to allow the removal of directories that are not empty - Windows requires that directories be emptied first, and UNIX provides an option for deleting entire sub-trees.

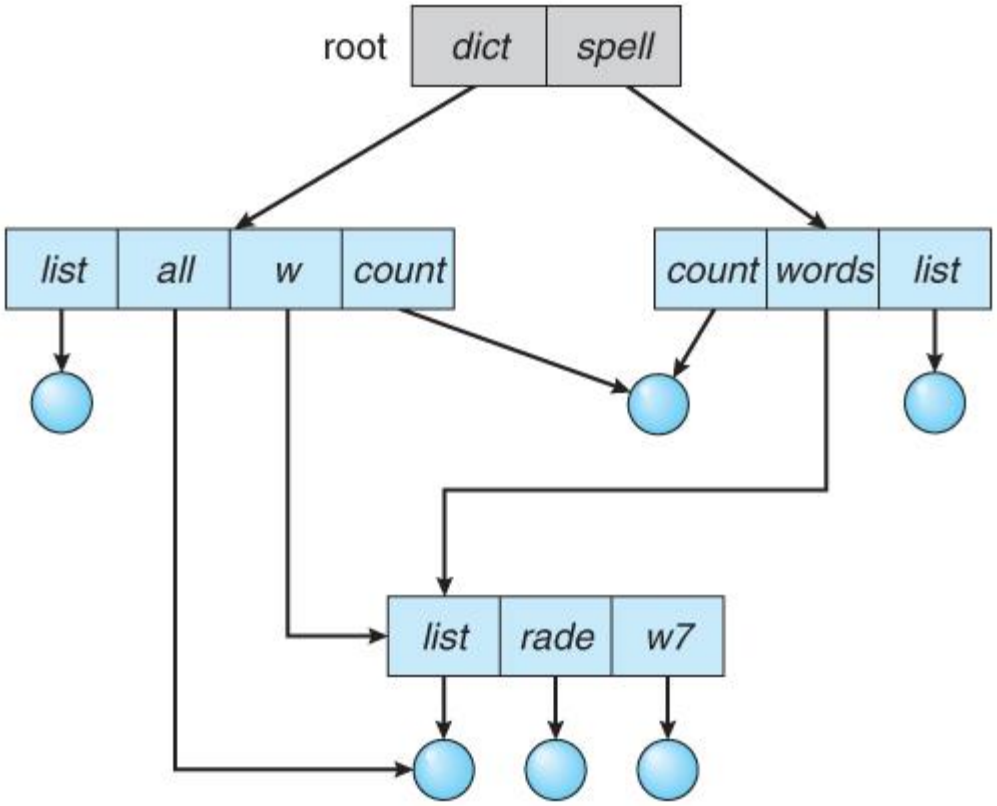


Tree-structured directory structure.

Acyclic-Graph Directories

- When the same files need to be accessed in more than one place in the directory structure (e.g. because they are being shared by more than one user / process), it can be useful to provide an acyclic-graph structure. (Note the ***directed*** arcs from parent to child.)
- UNIX provides two types of ***links*** for implementing the acyclic-graph structure. (See "man ln" for more details.)
  - A ***hard link*** (usually just called a link) involves multiple directory entries that both refer to the same file. Hard links are only valid for ordinary files in the same filesystem.
  - A ***symbolic link***, that involves a special file, containing information about where to find the linked file. Symbolic links may be used to link directories and/or files in other filesystems, as well as ordinary files in the current filesystem.
- Windows only supports symbolic links, termed ***shortcuts.***
- Hard links require a ***reference count***, or ***link count*** for each file, keeping track of how many directory entries are currently referring to this file. Whenever one of the references is removed the link count is reduced, and when it reaches zero, the disk space can be reclaimed.

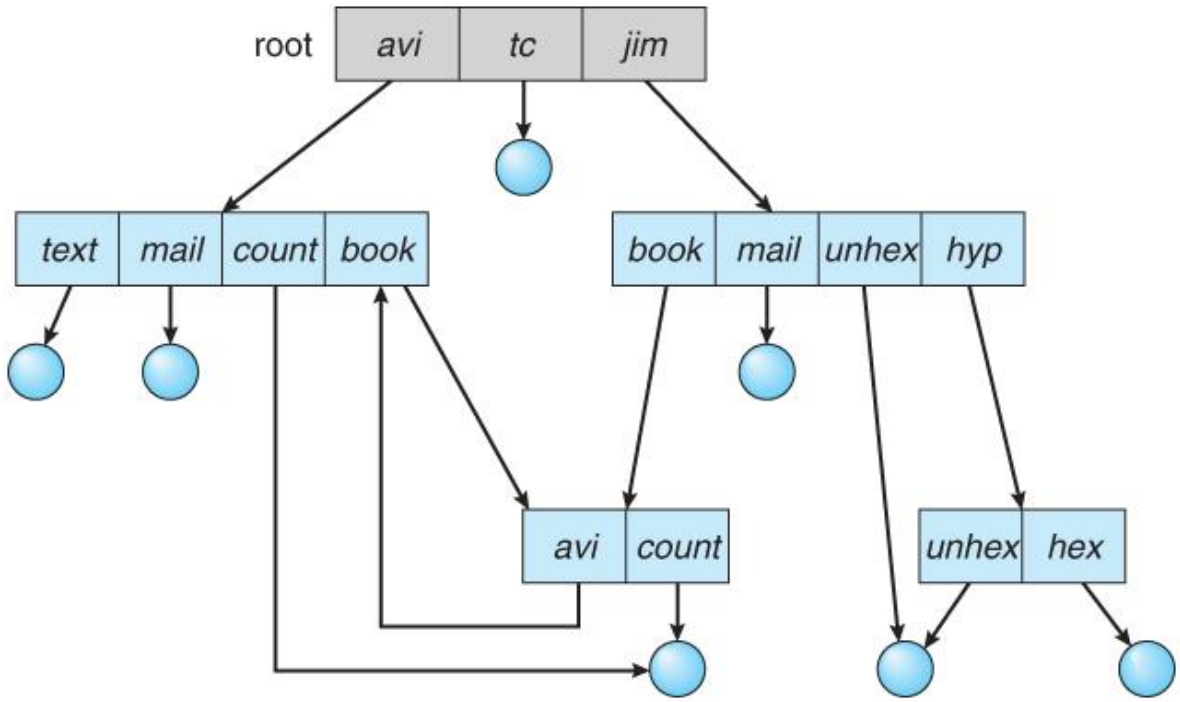
- For symbolic links there is some question as to what to do with the symbolic links when the original file is moved or deleted:
  - One option is to find all the symbolic links and adjust them also.
  - Another is to leave the symbolic links dangling, and discover that they are no longer valid the next time they are used.
  - What if the original file is removed, and replaced with another file having the same name before the symbolic link is next used?



Acyclic-graph directory structure.

General Graph Directory

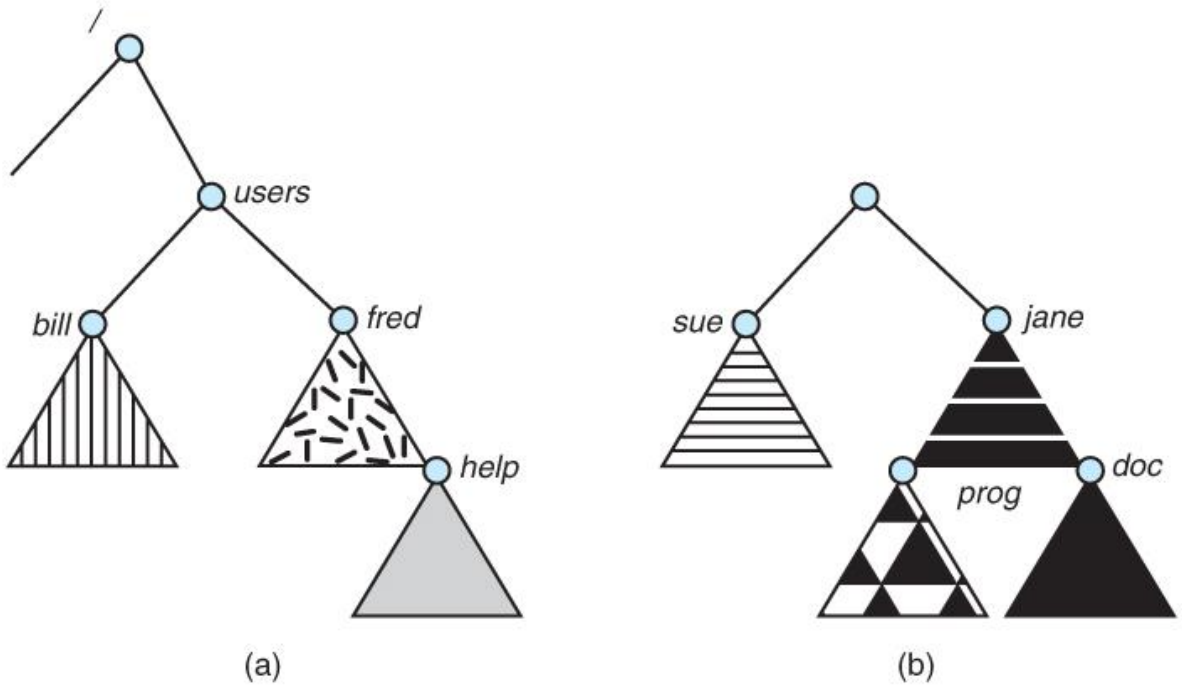
- If cycles are allowed in the graphs, then several problems can arise:
  - Search algorithms can go into infinite loops. One solution is to not follow links in search algorithms. (Or not to follow symbolic links, and to only allow symbolic links to refer to directories.)
  - Sub-trees can become disconnected from the rest of the tree and still not have their reference counts reduced to zero. Periodic garbage collection is required to detect and resolve this problem. (chkdsk in DOS and fsck in UNIX search for these problems, among others, even though cycles are not supposed to be allowed in either system. Disconnected disk blocks that are not marked as free are added back to the file systems with made-up file names, and can usually be safely deleted.)



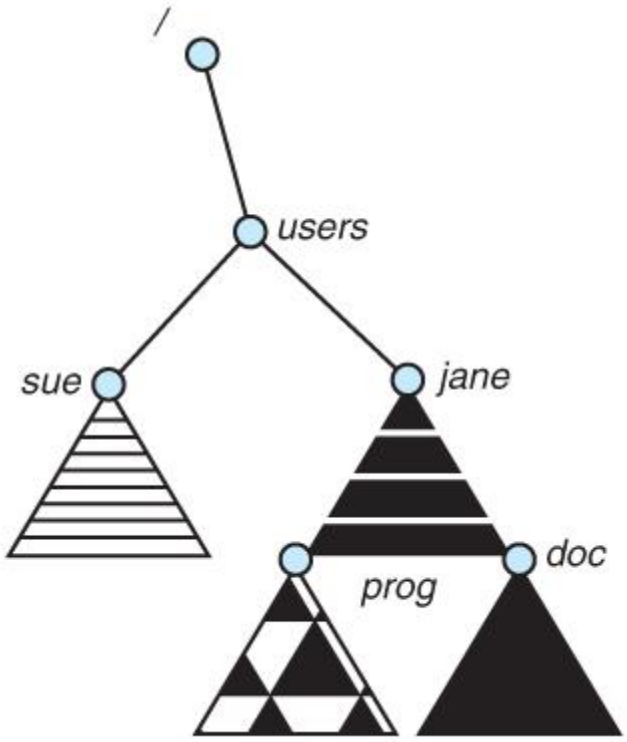
- General graph directory.

File-System Mounting

- The basic idea behind mounting file systems is to combine multiple file systems into one large tree structure.
- The mount command is given a filesystem to mount and a ***mount point*** (directory) on which to attach it.
- Once a file system is mounted onto a mount point, any further references to that directory actually refer to the root of the mounted file system.
- Any files (or sub-directories) that had been stored in the mount point directory prior to mounting the new filesystem are now hidden by the mounted filesystem, and are no longer available. For this reason some systems only allow mounting onto empty directories.
- Filesystems can only be mounted by root, unless root has previously configured certain filesystems to be mountable onto certain pre-determined mount points. (E.g. root may allow users to mount floppy filesystems to /mnt or something like it.) Anyone can run the mount command to see what filesystems are currently mounted.
- Filesystems may be mounted read-only, or have other restrictions imposed.



File system. (a) Existing system. (b) Unmounted volume.



Mount point.

- The traditional Windows OS runs an extended two-tier directory structure, where the first tier of the structure separates volumes by drive letters, and a tree structure is implemented below that level.
- Macintosh runs a similar system, where each new volume that is found is automatically mounted and added to the desktop when it is found.

- More recent Windows systems allow filesystems to be mounted to any directory in the filesystem, much like UNIX.

File Sharing

Multiple Users

- On a multi-user system, more information needs to be stored for each file:
  - The owner (user) who owns the file, and who can control its access.
  - The group of other user IDs that may have some special access to the file.
  - What access rights are afforded to the owner (**U**ser), the **G**roup, and to the rest of the world (the universe, a.k.a. **O**thers.)
  - Some systems have more complicated access control, allowing or denying specific accesses to specifically named users or groups.

Remote File Systems

- The advent of the Internet introduces issues for accessing files stored on remote computers
  - The original method was ftp, allowing individual files to be transported across systems as needed. Ftp can be either account and password controlled, or ***anonymous***, not requiring any user name or password.
  - Various forms of ***distributed file systems*** allow remote file systems to be mounted onto a local directory structure, and accessed using normal file access commands. (The actual files are still transported across the network as needed, possibly using ftp as the underlying transport mechanism.)
  - The WWW has made it easy once again to access files on remote systems without mounting their filesystems, generally using (anonymous) ftp as the underlying file transport mechanism.

The Client-Server Model

- When one computer system remotely mounts a filesystem that is physically located on another system, the system which physically owns the files acts as a ***server***, and the system which mounts them is the ***client.***
- User IDs and group IDs must be consistent across both systems for the system to work properly. (I.e. this is most applicable across multiple computers managed by the same organization, shared by a common group of users.)
- The same computer can be both a client and a server. (E.g. cross-linked file systems.)
- There are a number of security concerns involved in this model:
  - Servers commonly restrict mount permission to certain trusted systems only. Spoofing (a computer pretending to be a different computer) is a potential security risk.
  - Servers may restrict remote access to read-only.

- Servers restrict which filesystems may be remotely mounted. Generally the information within those subsystems is limited, relatively public, and protected by frequent backups.
- The NFS (Network File System) is a classic example of such a system.

Distributed Information Systems

- The ***Domain Name System, DNS,*** provides for a unique naming system across all of the Internet.
- Domain names are maintained by the ***Network Information System, NIS***, which unfortunately has several security issues. NIS+ is a more secure version, but has not yet gained the same widespread acceptance as NIS.
- Microsoft's ***Common Internet File System, CIFS***, establishes a ***network login*** for each user on a networked system with shared file access. Older Windows systems used ***domains***, and newer systems (XP, 2000), use ***active directories.*** User names must match across the network for this system to be valid.
- A newer approach is the ***Lightweight Directory-Access Protocol, LDAP,*** which provides a ***secure single sign-on*** for all users to access all resources on a network. This is a secure system which is gaining in popularity, and which has the maintenance advantage of combining authorization information in one central location.

Failure Modes

- When a local disk file is unavailable, the result is generally known immediately, and is generally non-recoverable. The only reasonable response is for the response to fail.
- However when a remote file is unavailable, there are many possible reasons, and whether or not it is unrecoverable is not readily apparent. Hence most remote access systems allow for blocking or delayed response, in the hopes that the remote system (or the network) will come back up eventually.

- The Andrew File System, AFS uses the following semantics:
  - Writes to an open file are not immediately visible to other users.
  - When a file is closed, any changes made become available only to users who open the file at a later time.
- According to these semantics, a file can be associated with multiple (possibly different) views. Almost no constraints are imposed on scheduling accesses. No user is delayed in reading or writing their personal copy of the file.
- AFS file systems may be accessible by systems around the world. Access control is maintained through (somewhat) complicated access control lists, which may grant access to the entire world (literally) or to specifically named users accessing the files from specifically named remote environments.

Protection

- Files must be kept safe for reliability (against accidental damage), and protection (against deliberate malicious access.) The former is usually managed with backup copies. This section discusses the latter.
- One simple protection scheme is to remove all access to a file. However this makes the file unusable, so some sort of controlled access must be arranged.

Types of Access

- The following low-level operations are often controlled:
  - Read - View the contents of the file
  - Write - Change the contents of the file.
  - Execute - Load the file onto the CPU and follow the instructions contained therein.
  - Append - Add to the end of an existing file.
  - Delete - Remove a file from the system.
  - List -View the name and other attributes of files on the system.
- Higher-level operations, such as copy, can generally be performed through combinations of the above.

Access Control

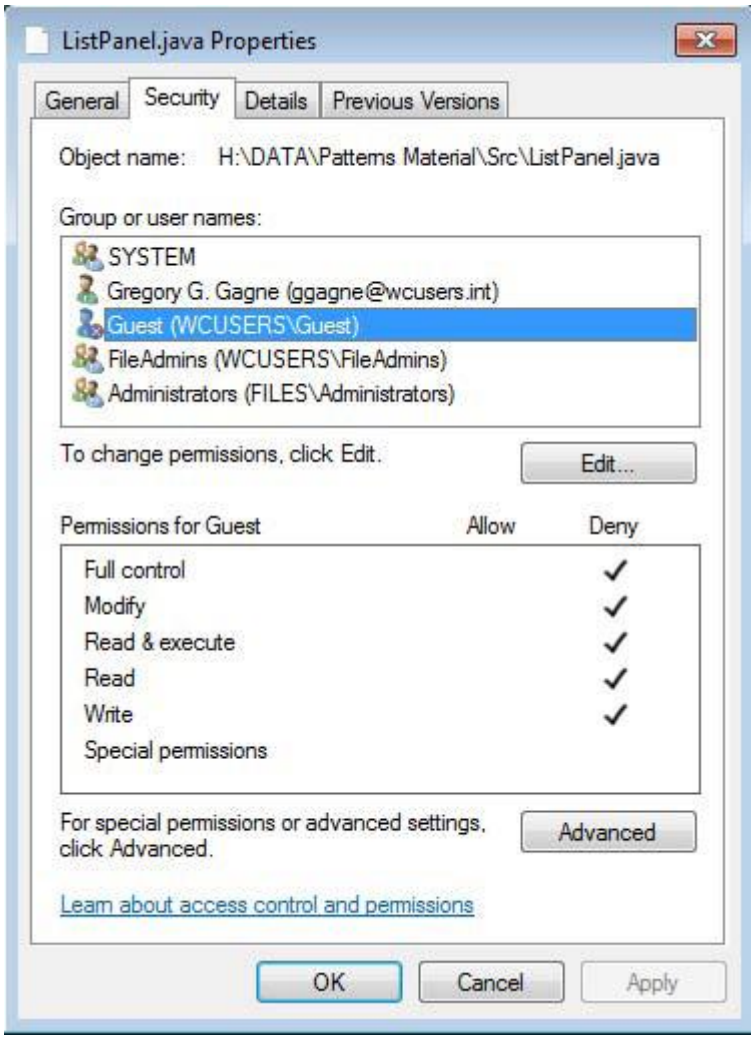
- One approach is to have complicated ***Access Control Lists, ACL,*** which specify exactly what access is allowed or denied for specific users or groups.
  - The AFS uses this system for distributed access.
  - Control is very finely adjustable, but may be complicated, particularly when the specific users involved are unknown. (AFS allows some wild cards, so for example all users on a certain remote system may be trusted, or a given username may be trusted when accessing from any remote system.)
- UNIX uses a set of 9 access control bits, in three groups of three. These correspond to R, W, and X permissions for each of the Owner, Group, and Others. (See "man chmod" for full details.) The RWX bits control the following privileges for ordinary files and directories:

	es	rectories
	ad (view) e contents.	ad directory contents. Required to get a listing of the directory.
	ite (change) e contents.	ange directory contents. Required to create or delete files.
	ecute file ntents as a ogram.	cess detailed directory information. Required to get a long listing, or to cess any specific file in the directory. Note that if a user has X but not R rmissions on a directory, they can still access specific files, but only if they eady know the name of the file they are trying to access.

- In addition there are some special bits that can also be applied:

- The set user ID (SUID) bit and/or the set group ID (SGID) bits applied to executable files temporarily change the identity of whoever runs the program to match that of the owner / group of the executable program. This allows users running specific programs to have access to files (***while running that program***) to which they would normally be unable to access. Setting of these two bits is usually restricted to root, and must be done with caution, as it introduces a potential security leak.

- Windows adjusts files access through a simple GUI:



Windows 7 access-control list management.

Other Protection Approaches and Issues

- Some systems can apply passwords, either to individual files, or to specific sub-directories, or to the entire system. There is a trade-off between the number of passwords that must be maintained (and remembered by the users) and the amount of information that is vulnerable to a lost or forgotten password.
- Older systems which did not originally have multi-user file access permissions (DOS and older versions of Mac) must now be ***retrofitted*** if they are to share files on a network.
- Access to a file requires access to all the files along its path as well. In a cyclic directory structure, users may have different access to the same file accessed through different paths.
- Sometimes just the knowledge of the existence of a file of a certain name is a security (or privacy) concern. Hence the distinction between the R and X bits on UNIX directories.