Syllabus

KCS-401: Operating System

B.TECH. (COMPUTER SCIENCE AND ENGINEERING)

FOURTH SEMESTER (DETAILED SYLLABUS)

Course Outcome (CO) Bloom's Knowledge Level (KL)

At the end of course, the student will be able to understand

CO 1: Understand the structure and functions of OS	K1, K2
CO 2: Learn about Processes, Threads and Scheduling algorithms.	K1, K2
CO 3: Understand the principles of concurrency and Deadlocks	K2
CO 4: Learn various memory management scheme	K2
CO 5: Study I/O management and File systems.	K2,K4

Unit-I

Introduction : Operating system and functions, Classification of Operating systems- Batch, Interactive, Time sharing, Real Time System, Multiprocessor Systems, Multiuser Systems, Multiprocess Systems, Multithreaded Systems, Operating System Structure- Layered structure, System Components, Operating System services, Reentrant Kernels, Monolithic and Microkernel Systems.

Unit-II

Concurrent Processes: Process Concept, Principle of Concurrency, Producer / Consumer Problem, Mutual Exclusion, Critical Section Problem, Dekker's solution, Peterson's solution, Semaphores, Test and Set operation; Classical Problem in Concurrency- Dining Philosopher Problem, Sleeping Barber Problem; Inter Process Communication models and Schemes, Process generation.

Unit-III

CPU Scheduling: Scheduling Concepts, Performance Criteria, Process States, Process Transition Diagram, Schedulers, Process Control Block (PCB), Process address space, Process identification information, Threads and their management, Scheduling Algorithms, Multiprocessor Scheduling. Deadlock: System model, Deadlock characterization, Prevention, Avoidance and detection, Recovery from deadlock.

Unit-IV

Memory Management: Basic bare machine, Resident monitor, Multiprogramming with fixed partitions, Multiprogramming with variable partitions, Protection schemes, Paging, Segmentation, Paged segmentation, Virtual memory concepts, Demand paging, Performance of demand paging, Page replacement algorithms, Thrashing, Cache memory organization, Locality of reference.

Unit-V

I/O Management and Disk Scheduling: I/O devices, and I/O subsystems, I/O buffering, Disk storage and disk scheduling, RAID. File System: File concept, File organization and access mechanism, File directories, and File sharing, File system implementation issues, File system protection and security.

Text Book:

T1: Silberschatz, Galvin and Gagne, **"Operating Systems Concepts"**, Wiley

References:

R1: Andrew S. Tanenbaum **"Modern Operating System "**Pearson Education

R2: D M Dhamdhere, "**Operating Systems : A Concept based Approach**", 2nd Edition,TMH

R3: William Stallings, "**Operating Systems: Internals and Design Principles**", 6th Edition, Pearson Education

R4: Harvey M Dietel, " **An Introduction to Operating System**", Pearson Education

R5: Sibsankar Halder and Alex A Aravind, "**Operating Systems**", Pearson Education

Unit – V

I/O Management and Disk Scheduling: I/O devices, and I/O subsystems, I/O buffering, Disk storage and disk scheduling, RAID. **File System:** File concept, File organization and access mechanism, File directories, and File sharing, File system implementation issues, File system protection and security.

Objective:

- To explain the function of file systems
- To describe the interfaces to file systems
- To discuss file-system design tradeoffs, including access methods, file sharing; file locking, and directory structures
- To explore file-system protection

File Concept

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

Program

File Attributes

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 Operations File is an **abstract data type**

Create

Write

Read

Reposition within file

Delete

Truncate

Open(F_i) – search the directory structure on disk for entry F_i, and move the content of entry to memory

Close (F_i) – move the content of entry F_i in memory to directory structure on disk

Open Files

Several pieces of data are needed to manage open files:

File pointer: pointer to last read/write location, per process that has the file open

File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it

Disk location of the file: cache of data access information

Access rights: per-process access mode information

Open File Locking

Provided by some operating systems and file systems

Mediates access to a file

Mandatory or advisory:

Mandatory – access is denied depending on locks held and requested

Advisory – processes can find status of locks and decide what to do

File Types – Name, Extension

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, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Access Methods

Sequential Access

read next

write next

reset

no read after last write

(rewrite)

Direct Access

read *n*

write *n*

position to *n*

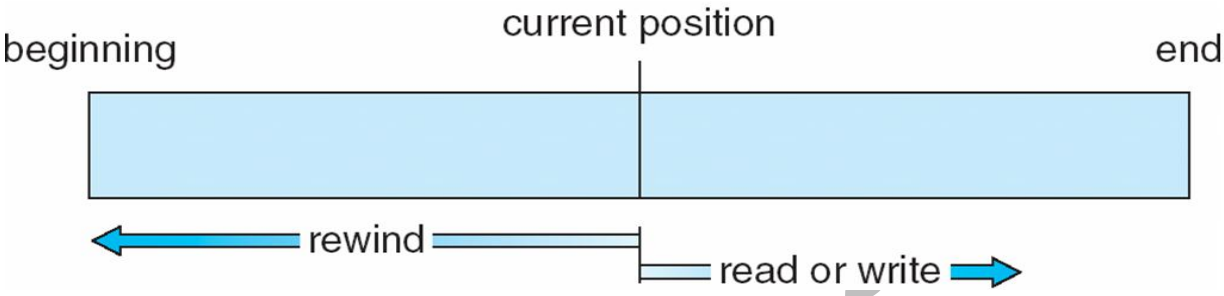
read next

write next

rewrite *n*

n = relative block number

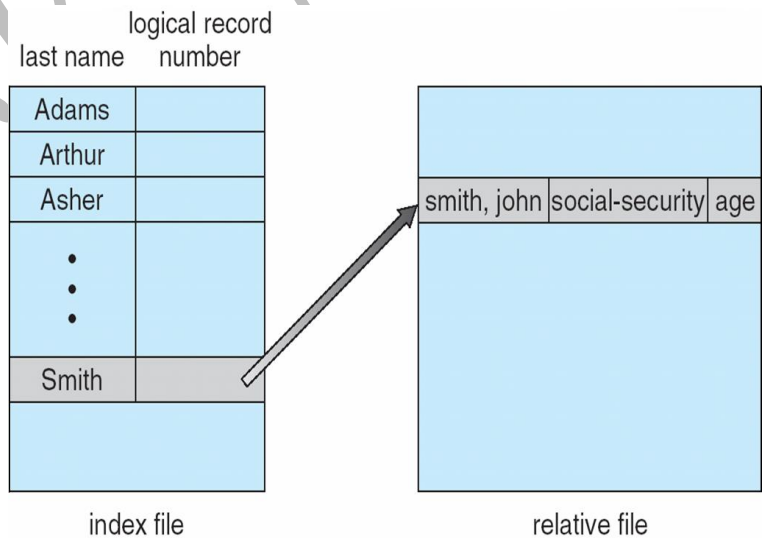
Sequential-access File



Simulation of Sequential Access on Direct-access File

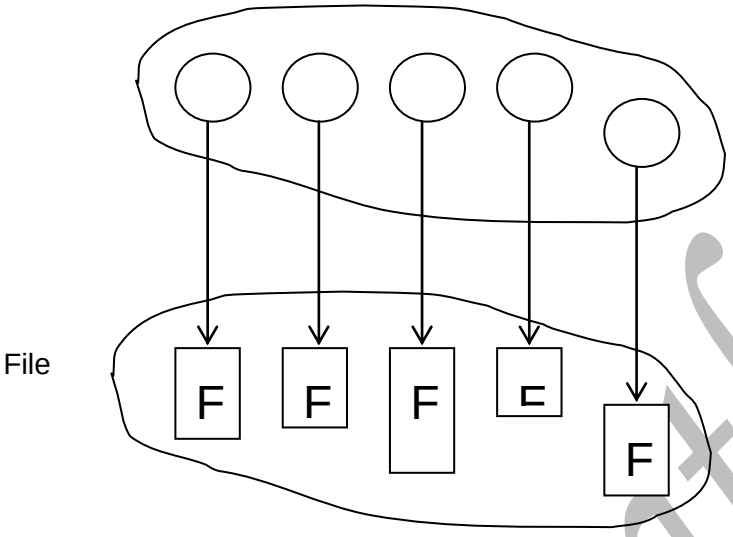
sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Example of Index and Relative Files



Directory Structure

A collection of nodes containing information about all files



Both the directory structure and the files reside on disk

Backups of these two structures are kept on tapes

Disk Structure

Disk can be subdivided into partitions

Disks or partitions can be RAID protected against failure Disk or partition can be used raw – without a file system, or formatted with a file system

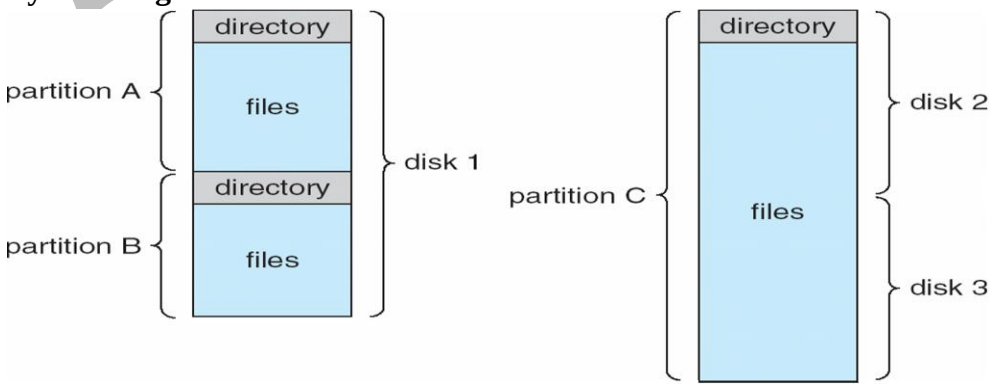
Partitions also known as minidisks, slices

Entity containing file system known as a volume

Each volume containing file system also tracks that file system's info in device directory or volume table of contents

As well as general-purpose file systems there are many special-purpose file systems, frequently all within the same operating system or computer

A Typical File-system Organization



Operations Performed on Directory

Search for a file

Create a file

Delete a file

List a directory

Rename a file

Traverse the file system

Organize the Directory (Logically) to Obtain

Efficiency – locating a file quickly

Naming – convenient to users

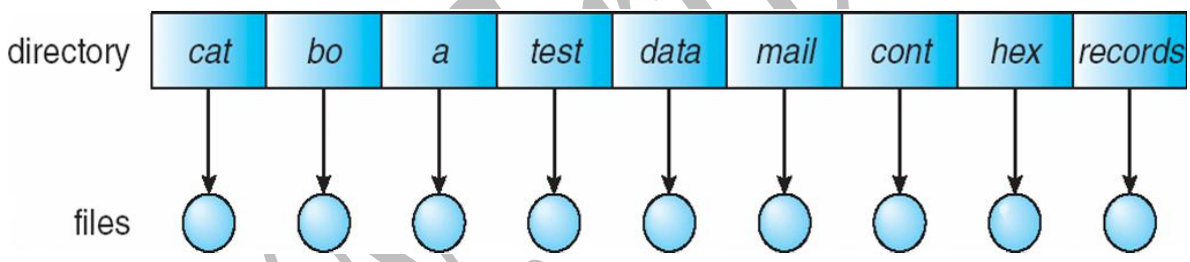
Two users can have same name for different files

The same file can have several different names

Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

Single-Level Directory

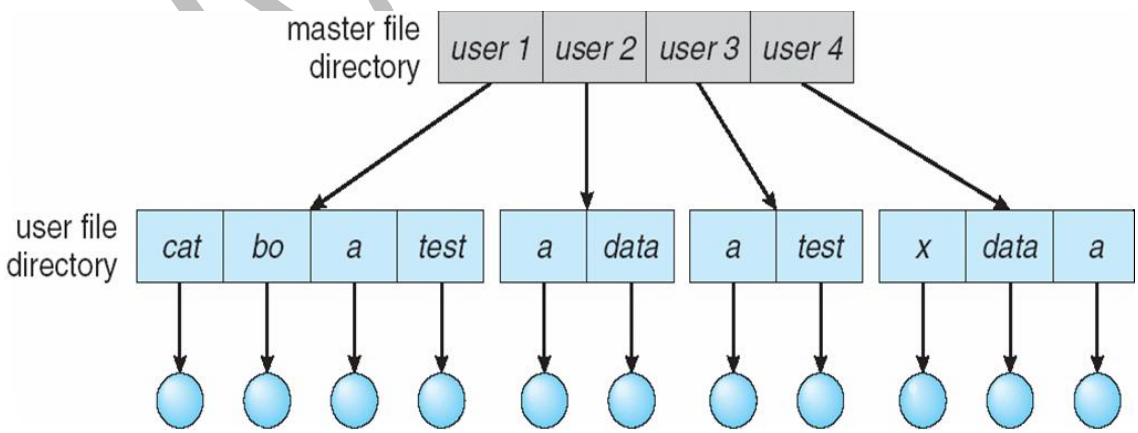
A single directory for all users



Naming problem
Grouping problem

Two-Level Directory

Separate directory for each user



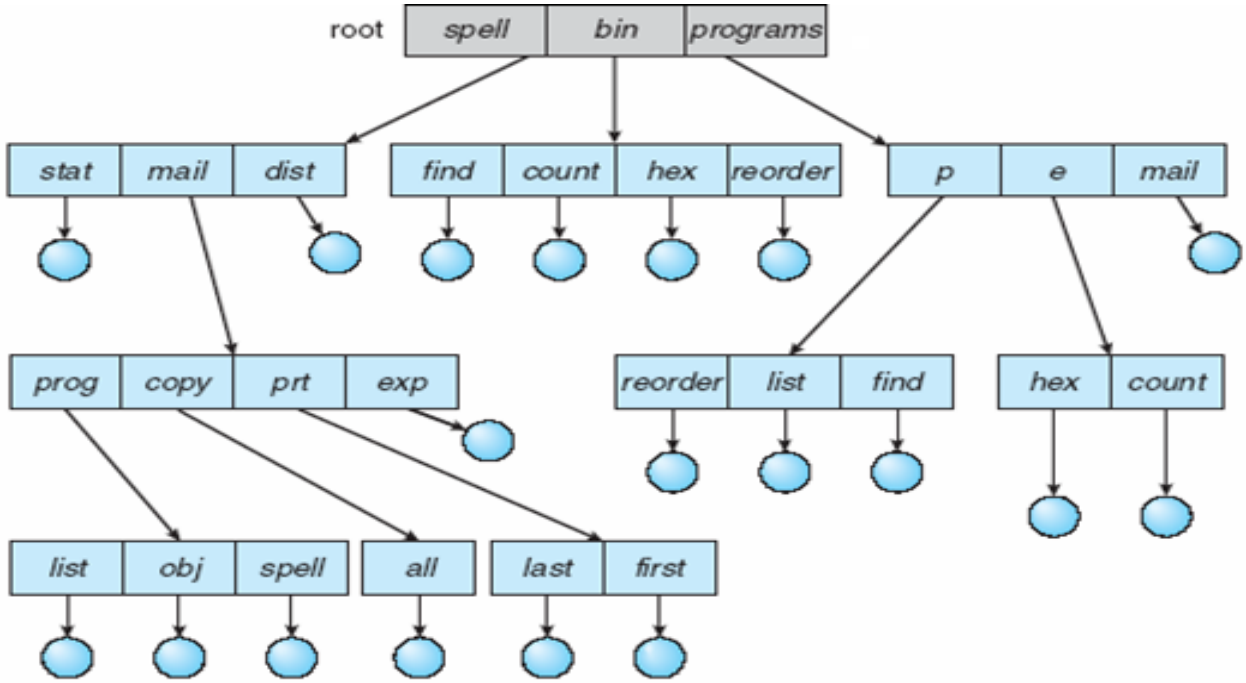
Path name

Can have the same file name for different user

Efficient searching

No grouping capability

Tree-Structured Directories



Efficient searchingnGrouping CapabilitynCurrent directory (working directory) cd /spell/mail/prog type list

Absolute or **relative** path name

Creating a new file is done in current directory

Delete a file

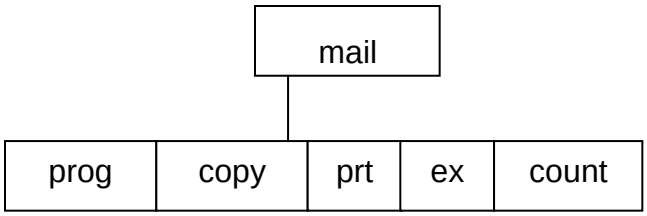
rm <file-name>

Creating a new subdirectory is done in current directory

mkdir <dir-name>

Example: if in current directory /mail

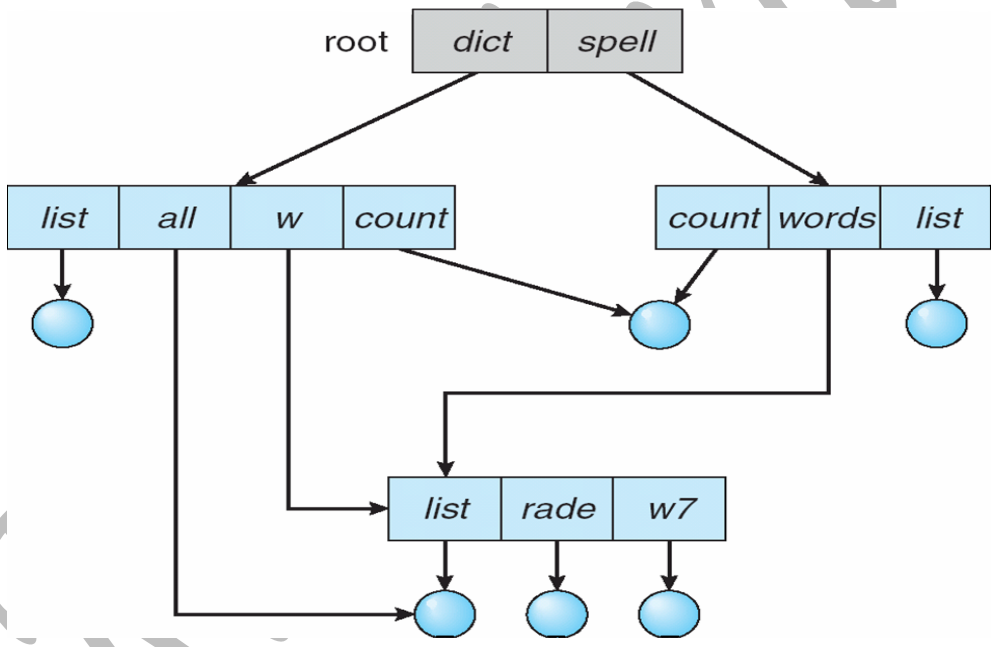
mkdir count



Deleting "mail" Þ deleting the entire subtree rooted by "mail"

Acyclic-Graph Directories

Have shared subdirectories and files



Two different names (aliasing) If *dict* deletes *list* Þ dangling pointer

Solutions:

Backpointers, so we can delete all pointers
Variable size records a problem

Backpointers using a daisy chain organization

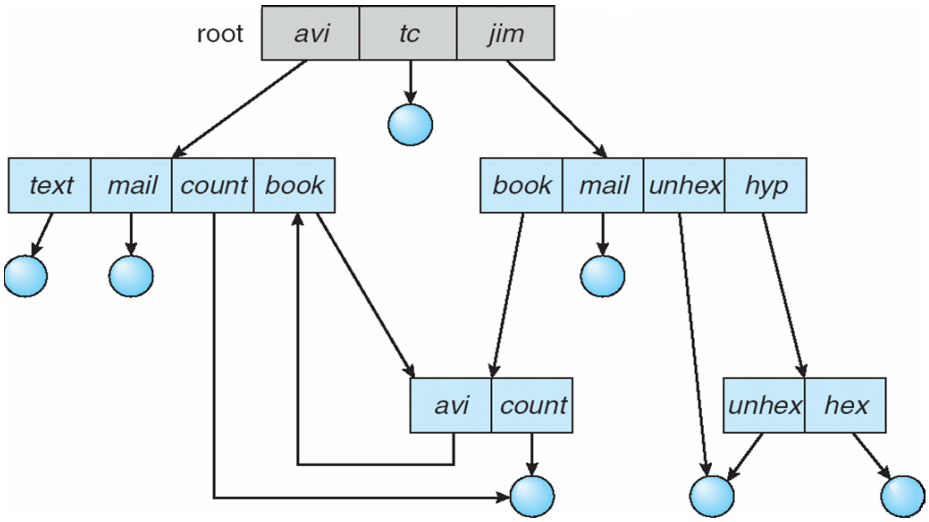
Entry-hold-count solution

New directory entry type

Link – another name (pointer) to an existing file

Resolve the link – follow pointer to locate the file

General Graph Directory



How do we guarantee no cycles?

Allow only links to file not subdirectories

Garbage collection

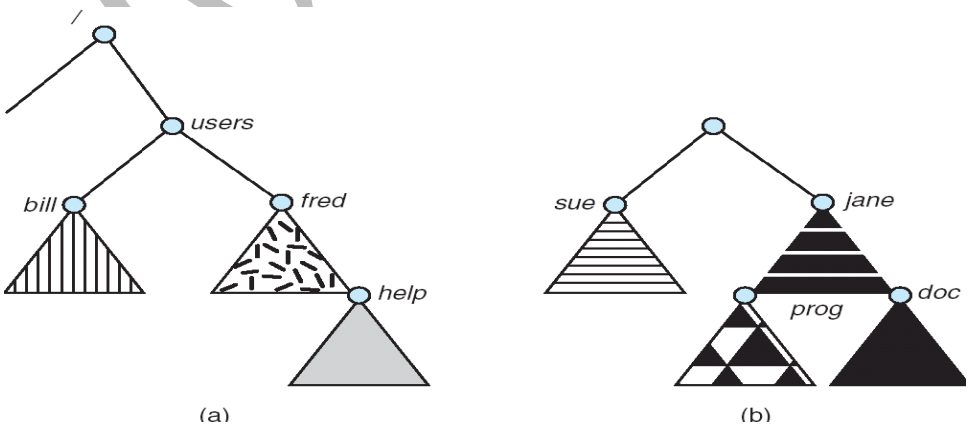
Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File System Mounting

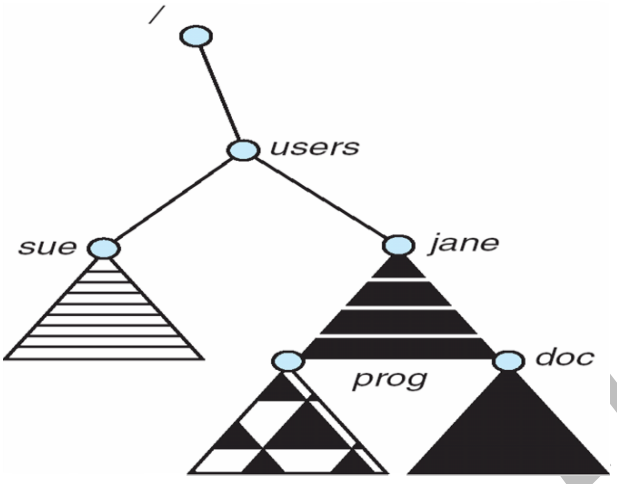
A file system must be **mounted** before it can be accessed

A unmounted file system (i.e. Fig. (b)) is mounted at a **mount point**

(a) Existing. (b) Unmounted Partition



Mount Point



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

File Sharing – Multiple Users

User IDs identify users, allowing permissions and protections to be per-user **Group IDs** allow users to be in groups, permitting group access rights

File Sharing – Remote File Systems

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

File Sharing – Failure Modes

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

File Sharing – Consistency Semantics

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

Protection

File owner/creator should be able to control:

what can be done

by whom

Types of access

Read

Write

Execute

Append

Delete

List

Access Lists and Groups

Mode of access: read, write, execute

Three classes of users

RWX

a) **owner access** 7 :1 1 1
RWX

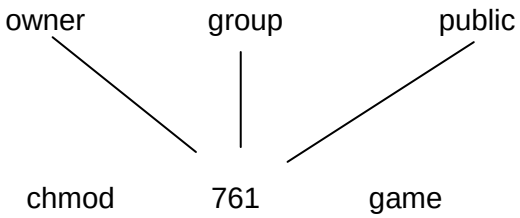
b) **group access** 6 :1 1 0

RWX

c) **public access** 1 :0 0 1

Ask manager to create a group (unique name), say G, and add some users to the group.

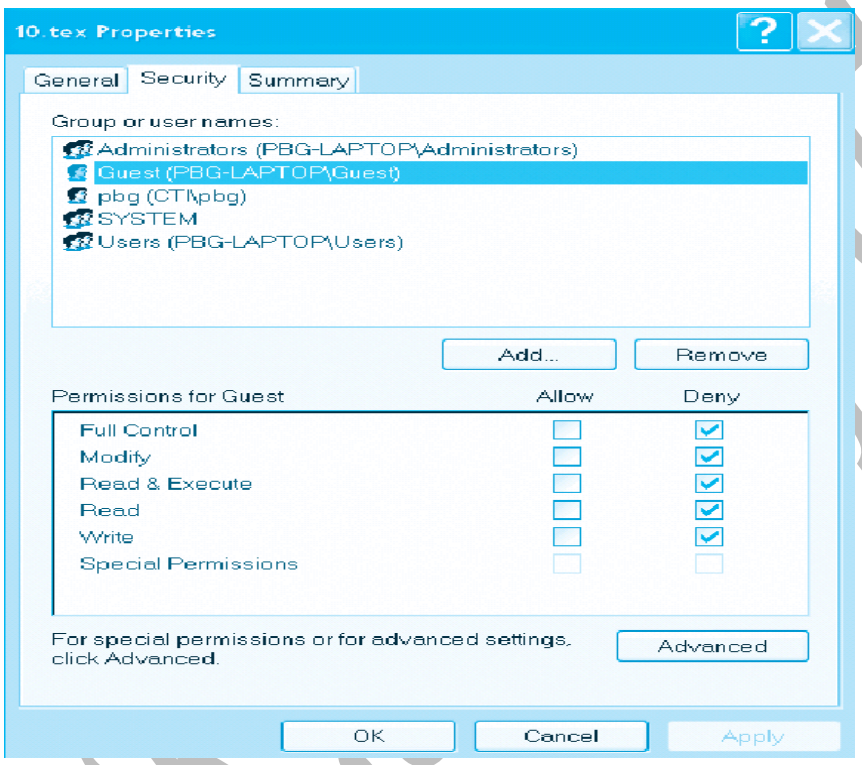
For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

Windows XP Access-control List Management



A Sample UNIX Directory Listing

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Mass-Storage Systems

Describe the physical structure of secondary and tertiary storage devices and the resulting effects on the uses of the devices Explain the performance characteristics of mass-storage devices. Discuss operating-system services provided for mass storage, including RAID and HSM

Overview of Mass Storage Structure

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

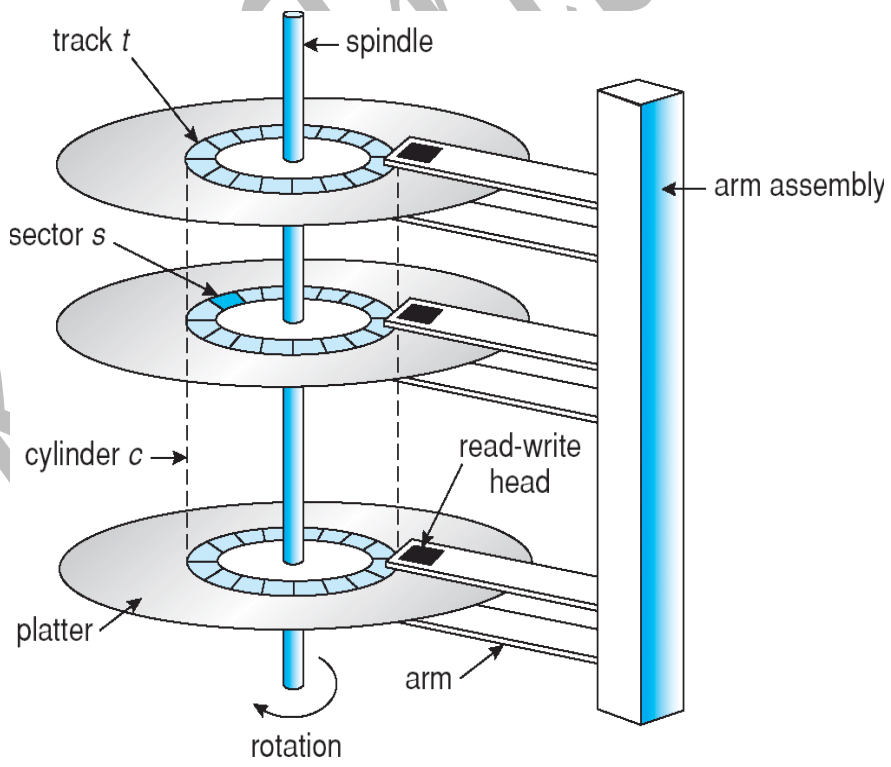
Disks can be removable

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

Moving-head Disk Mechanism



Magnetic tape

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

Common technologies are 4mm, 8mm, 19mm, LTO-2 and SDLT

Disk Structure

Disk drives are addressed as large 1-dimensional arrays of logical locks, 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 Attachment

Host-attached storage accessed through I/O ports talking to I/O busses

SCSI itself is a bus, up to 16 devices on one cable, SCSI initiator requests operation and SCSI targets perform tasks

Each target can have up to 8 logical units (disks attached to device controller

FC is high-speed serial architecture

Can be switched fabric with 24-bit address space – the basis of storage area networks (SANs) in which many hosts attach to many storage units

Can be arbitrated loop (FC-AL) of 126 devices

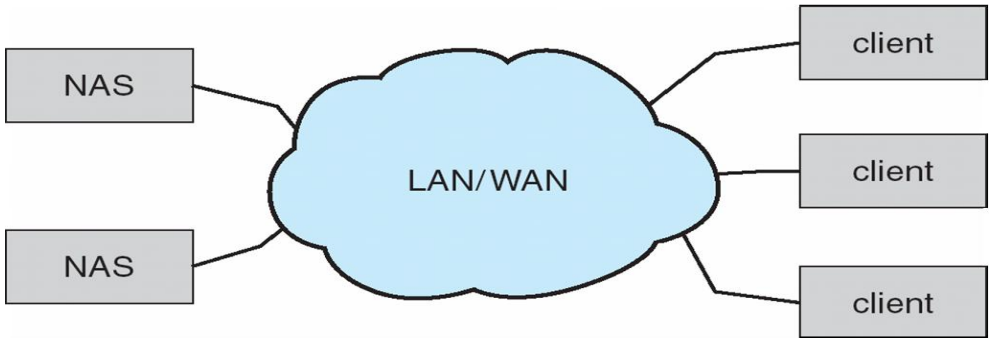
Network-Attached Storage

Network-attached storage (NAS) is storage made available over a network rather than over a local connection (such as a bus)

NFS and CIFS are common protocols

Implemented via remote procedure calls (RPCs) between host and storage

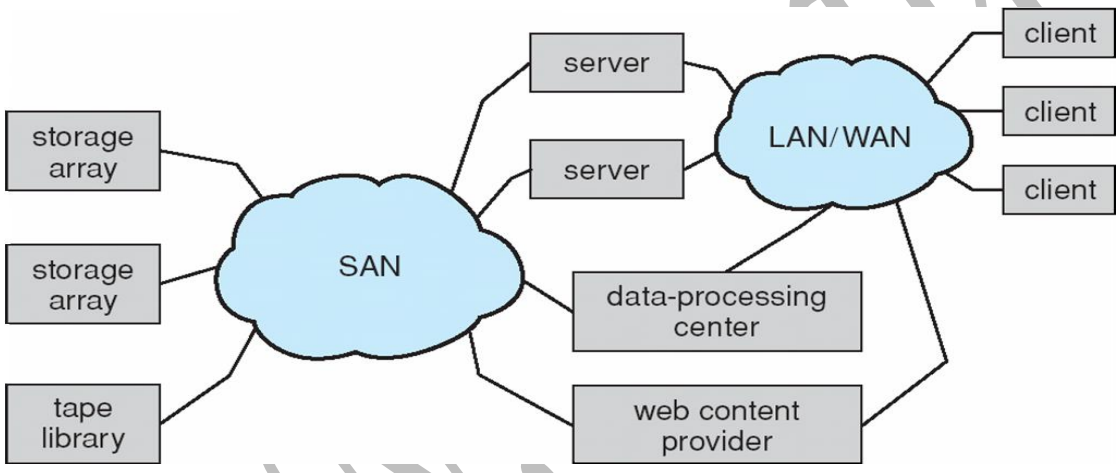
New iSCSI protocol uses IP network to carry the SCSI protocol



Storage Area Network

Common in large storage environments (and becoming more common)

Multiple hosts attached to multiple storage arrays - flexible



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

Several algorithms exist to schedule the servicing of disk I/O requests

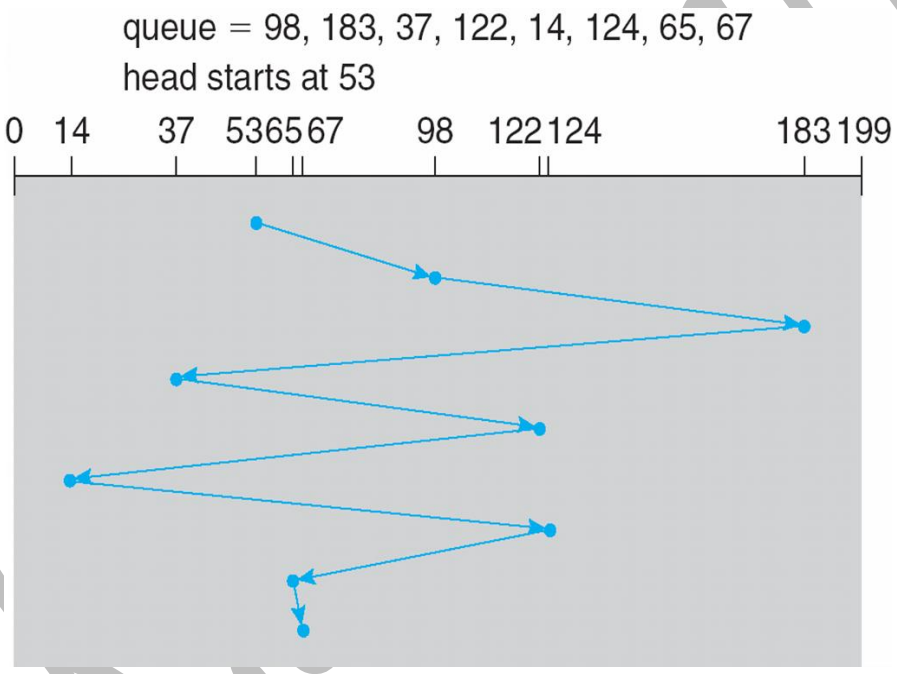
We illustrate them with a request queue (0-199)

98, 183, 37, 122, 14, 124, 65, 67

Head pointer 53

First Come & First Serve

Illustration shows total head movement of 640 cylinders

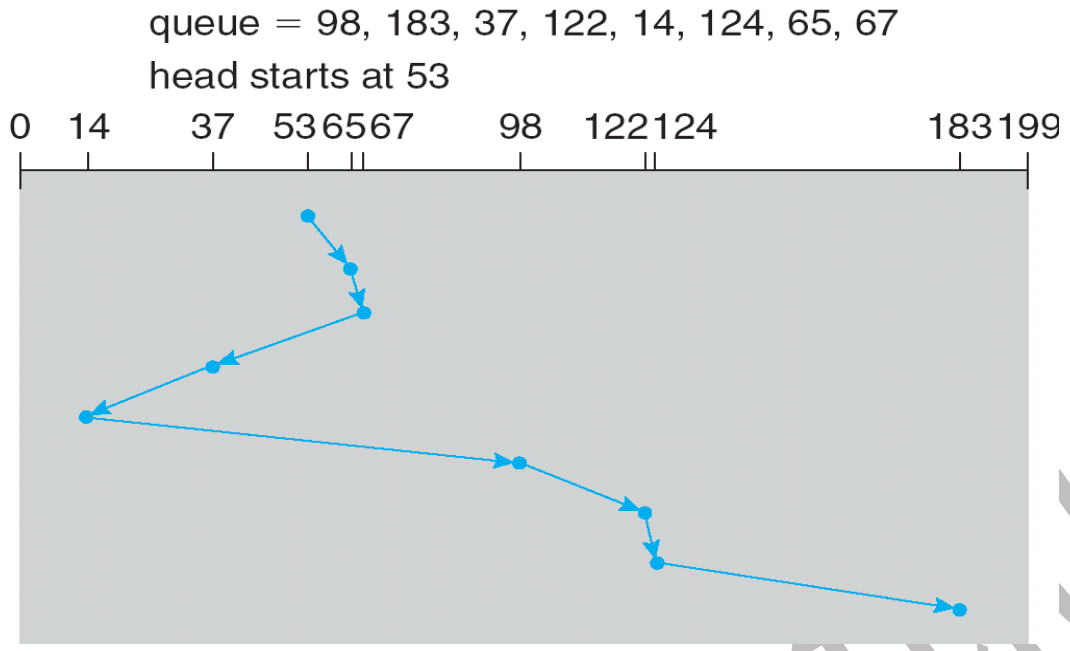


Shortest Seek Time First

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

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

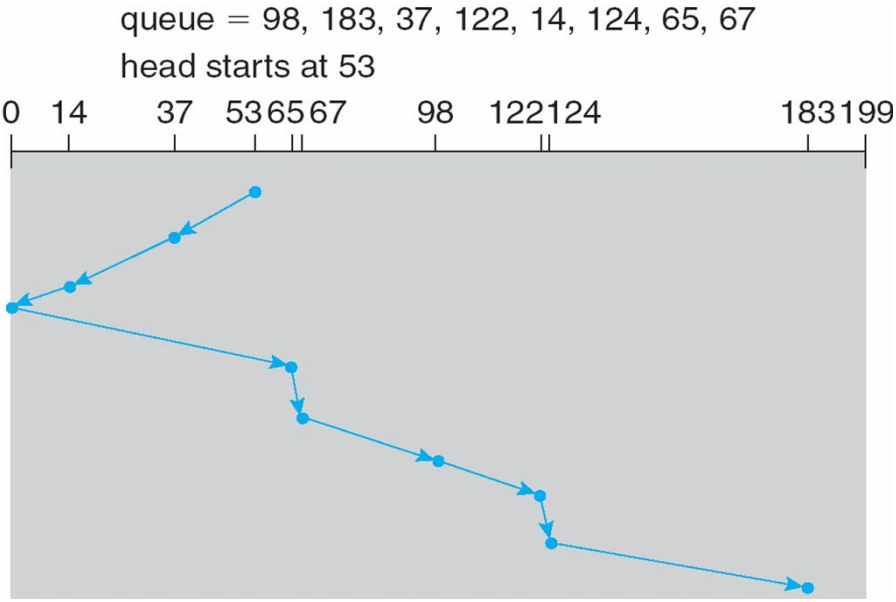
Illustration shows total head movement of 236 cylinders



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. SCAN algorithm Sometimes called the elevator algorithm

Illustration shows total head movement of 208 cylinders

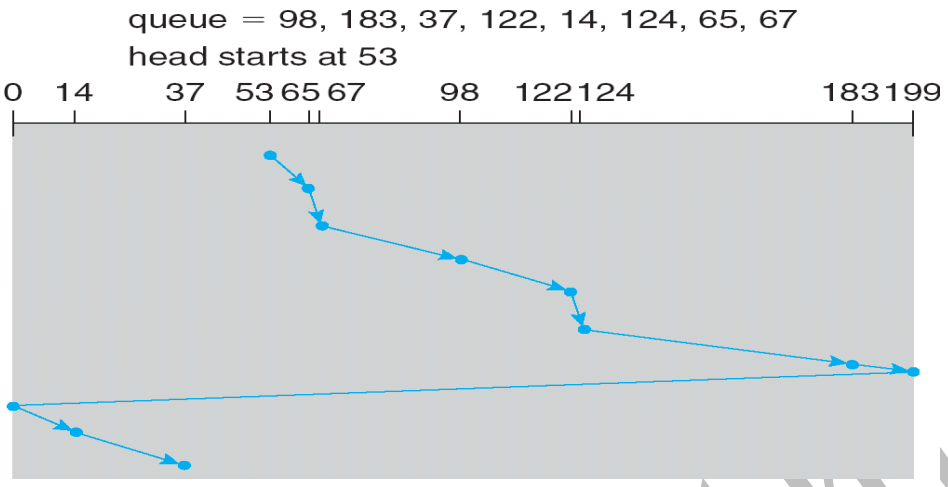


C-SCAN

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.

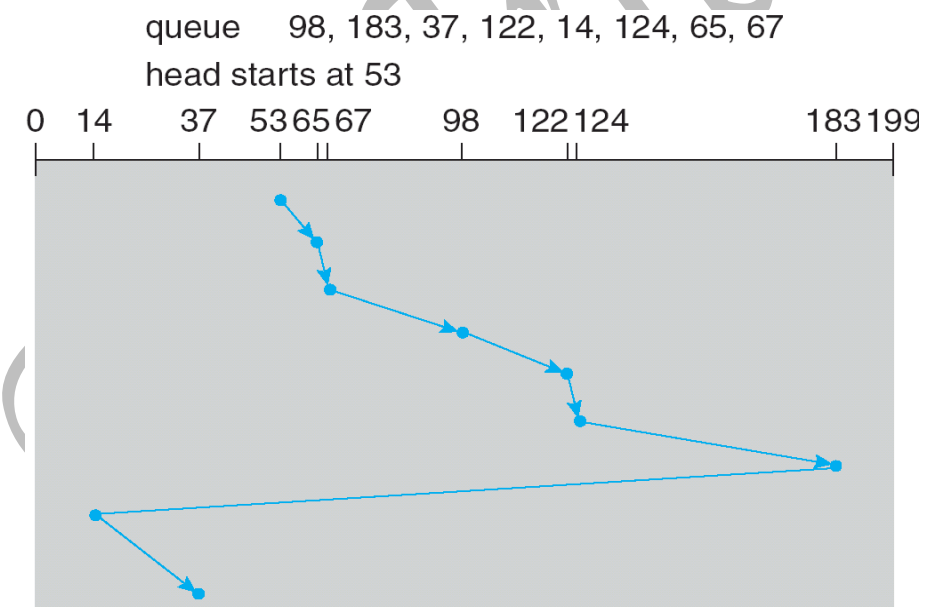
Treats the cylinders as a circular list that wraps around from the last cylinder to the first one



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



Selecting a Disk-Scheduling Algorithm

SSTF is common and has a natural appeal.

SCAN and C-SCAN perform better for systems that place a heavy load on the disk.

Performance depends on the number and types of requests.

Requests for disk service can be influenced by the file-allocation method.

The disk-scheduling algorithm should be written as a separate module of the operating system, allowing it to be replaced with a different algorithm if necessary.

Either SSTF or LOOK is a reasonable choice for the default algorithm.

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".

To increase efficiency most file systems group blocks into clusters

- Disk I/O done in blocks
- File I/O done in clusters

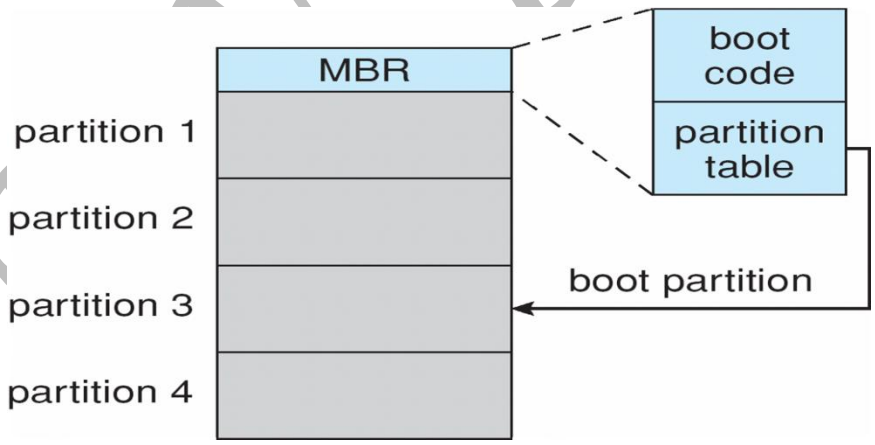
Boot block initializes system

The bootstrap is stored in ROM

Bootstrap loader program

Methods such as sector sparing used to handle bad blocks

Booting from a Disk in Windows 2000



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

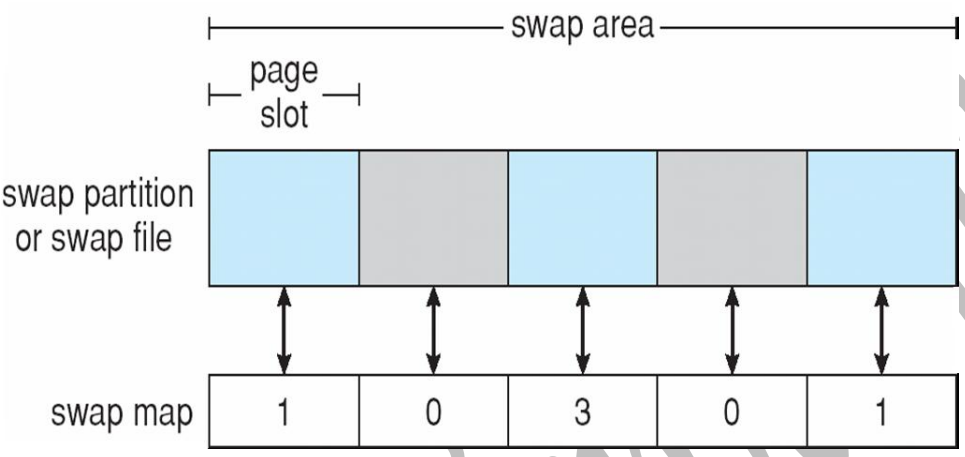
Swap-space management

l4.3BSD allocates swap space when process starts; holds text segment (the program) and data segment

Kernel uses swap maps to track swap-space use

Solaris 2 allocates swap space only when a page is forced out of physical memory, not when the virtual memory page is first created

Data Structures for Swapping on Linux Systems



Redundant Array of Independent Disk Structure:

RAID – multiple disk drives provides reliability via redundancy. Increases the mean time to failure Frequently combined with NVRAM to improve write performance.

RAID is arranged into six different levels.

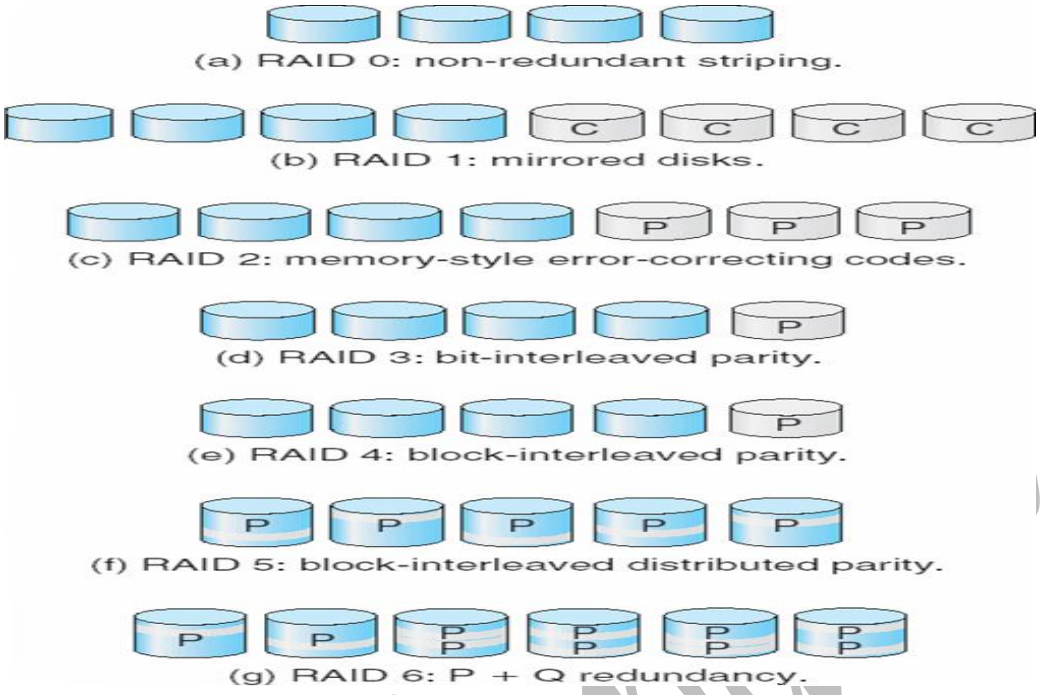
Several improvements in disk-use techniques involve the use of multiple disks working co-operatively Disk striping uses a group of disks as one storage unit. RAID schemes improve performance and improve the reliability of the storage system by storing redundant data Mirroring or shadowing (RAID 1) keeps duplicate of each disk.

Striped mirrors (RAID 1+0) or mirrored stripes (RAID 0+1) provide high performance and high reliability.

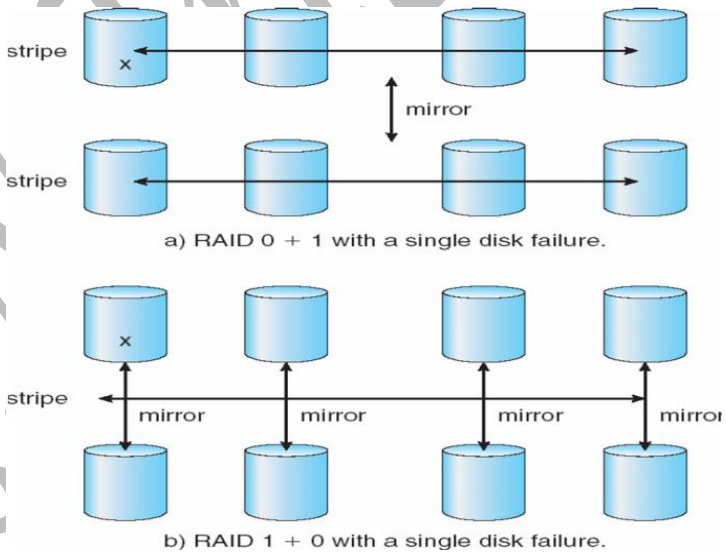
Block interleaved parity (RAID 4, 5, 6) uses much less redundancy.

RAID within a storage array can still fail if the array fails, so automatic replication of the data between arrays is common.

Frequently, a small number of hot-spare disks are left unallocated, automatically replacing a failed disk and having data rebuilt onto them.



RAID (0 + 1) and (1 + 0)



Extensions

RAID alone does not prevent or detect data corruption or other errors, just disk failures

Solaris ZFS adds checksums of all data and metadata

Checksums kept with pointer to object, to detect if object is the right one and whether it changed

Can detect and correct data and metadata corruption

ZFS also removes volumes, partititions

Disks allocated in pools

Filesystems with a pool share that pool, use and release space like “malloc” and “free” memory allocate / release calls

Protection

Discuss the goals and principles of protection in a modern computer system

Explain how protection domains combined with an access matrix are used to specify the resources a process may access

Examine capability and language-based protection systems

Goals of Protection

Operating system consists of a collection of objects, hardware or software. Each object has a unique name and can be accessed through a well-defined set of operations Protection problem - ensure that each object is accessed correctly and only by those processes that are allowed to do son

Principles of Protection

Guiding principle – principle of least privilege

lPrograms, users and systems should be given just enough privileges to perform their tasks

Access Matrix

View protection as a matrix (*access matrix*)

Rows represent domains

Columns represent objects

Access(i, j) is the set of operations that a process executing in Domain$_i$ can invoke on Object$_j$

domain \ object	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Use of Access Matrix

If a process in Domain D_i tries to do “op” on object O_j, then “op” must be in the access matrixnCan be expanded to dynamic protection

Operations to add, delete access rights

Special access rights:

- *owner of O_i*
- *copy op from O_i to O_j*
- *control – D_i can modify D_j access rights*
- *transfer – switch from domain D_i to D_j*

Access matrix design separates mechanism from policy

Mechanism

- Operating system provides access-matrix + rules
- If ensures that the matrix is only manipulated by authorized agents and that rules are strictly enforced Policy
- User dictates policy
- Who can access what object and in what mode

Implementation of Access Matrix

Each column = Access-control list for one object
Defines who can perform what operation.

Domain 1 = Read, Write
Domain 2 = Read
Domain 3 = Read

M Each Row = Capability List (like a key)

Fore each domain, what operations allowed on what objects.

Object 1 – Read

Object 4 – Read, Write, Execute

Object 5 – Read, Write, Delete, Copy

Access Matrix of Figure A With Domains as Objects

domain \ object	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Access Matrix with *Copy* Rights

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	execute		write*
D_2	execute	read*	execute
D_3	execute	read	

(b)

Access Matrix With *Owner* Rights

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		read* owner	read* owner write
D_3	execute		

(a)

object / domain	F_1	F_2	F_3
D_1	owner execute		write
D_2		owner read* write*	read* owner write
D_3		write	write

(b)

Modified Access Matrix of Figure B

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch control
D_3		read	execute					
D_4	write		write		switch			

Access Control

Protection can be applied to non-file resources

Solaris 10 provides role-based access control **(**RBAC**)** to implement least privilege

Privilege is right to execute system call or use an option within a system call

Can be assigned to processes

Users assigned roles granting access to privileges and programs

I/O Systems

Explore the structure of an operating system's I/O subsystem

Discuss the principles of I/O hardware and its complexity

Provide details of the performance aspects of I/O hardware and software

I/O Hardware

Incredible variety of I/O devices

Port

Bus (daisy chain or shared direct access)

Controller (host adapter)

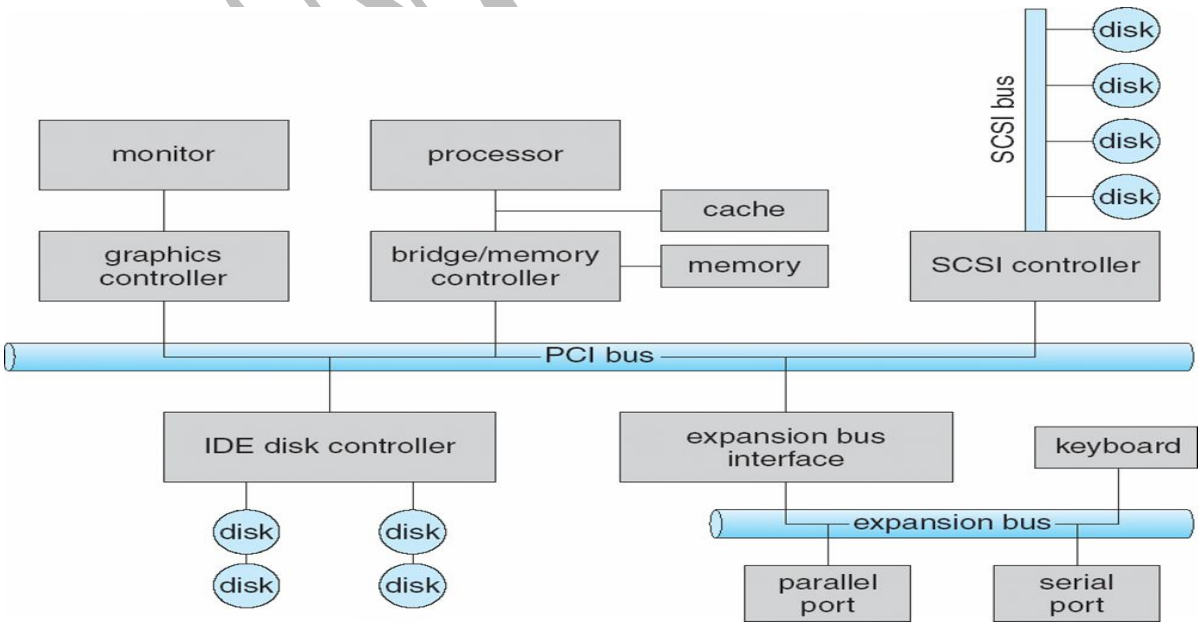
I/O instructions control devices

Devices have addresses, used by

Direct I/O instructions

Memory-mapped I/O

A Typical PC Bus Structure



Device I/O Port Locations on PCs (partial)

I/O address range (hexadecimal)	device
000–00F	DMA controller
020–021	interrupt controller
040–043	timer
200–20F	game controller
2F8–2FF	serial port (secondary)
320–32F	hard-disk controller
378–37F	parallel port
3D0–3DF	graphics controller
3F0–3F7	diskette-drive controller
3F8–3FF	serial port (primary)