File and File–System Management

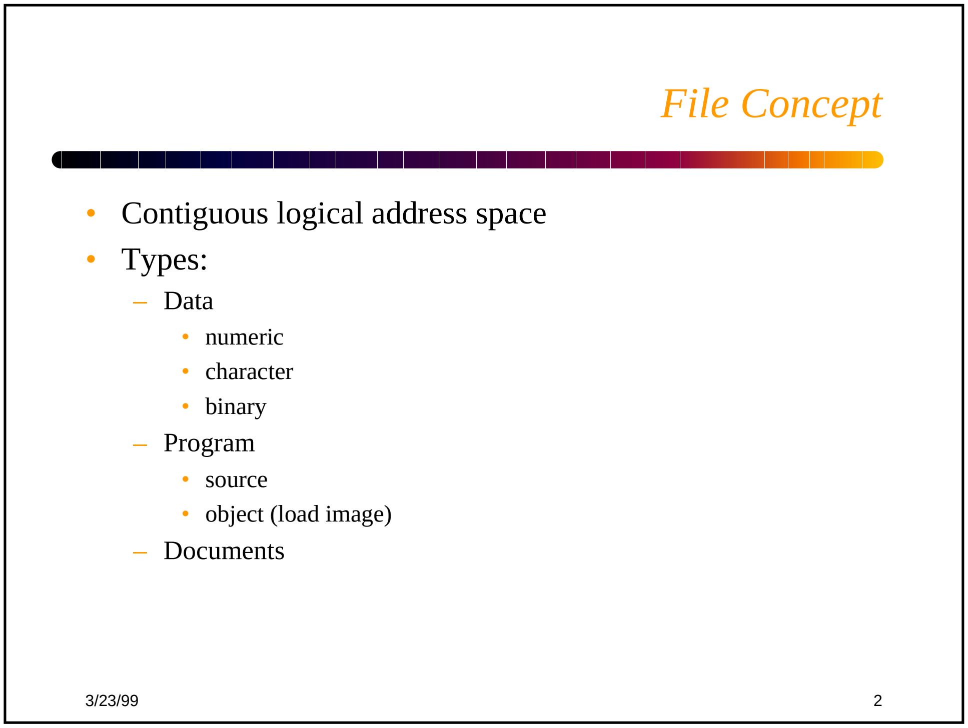
CS 502

Spring 99

WPI MetroWest/Southboro Campus

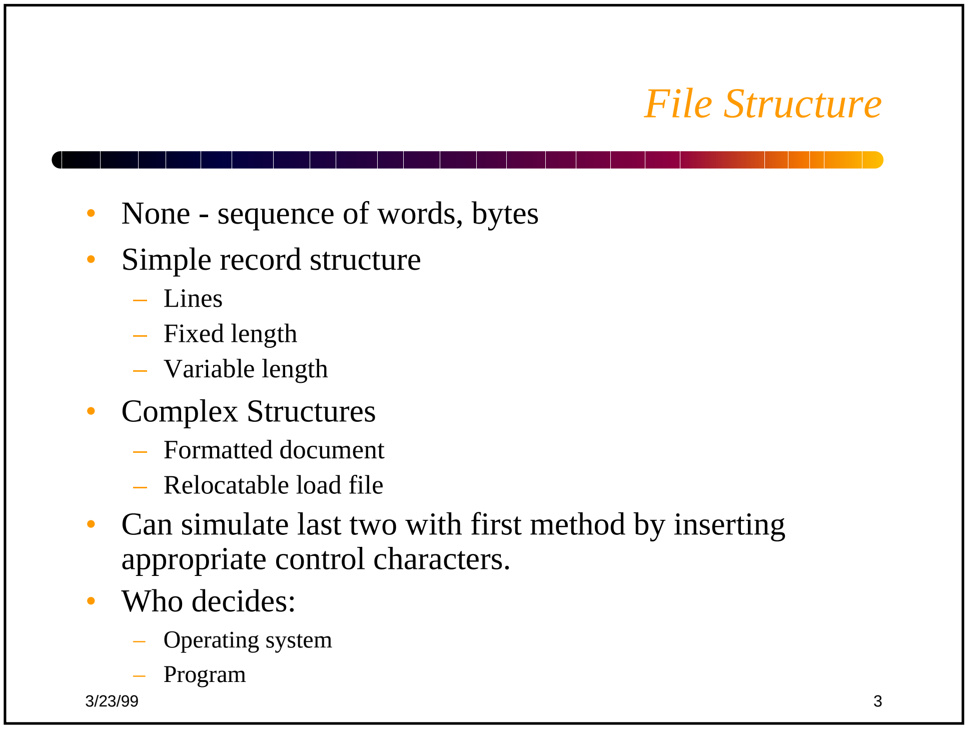
File and File–System Management Outline

- File–System Interface
  - File Concept
  - Access Methods
  - Directory Structure
  - Protection
  - Consistency Semantics
- File–System Implementation
  - File-System Structure
  - Allocation Methods
  - Free-Space Management
  - Directory Implementation
  - Efficiency and Performance
  - Recovery



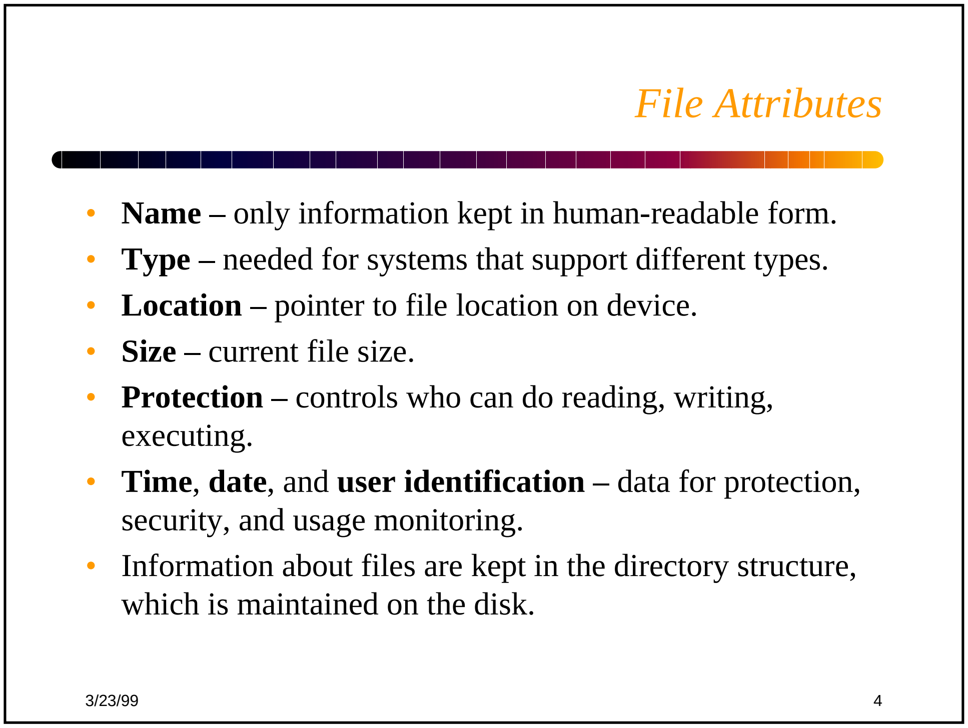
File Concept

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program
    - source
    - object (load image)
  - Documents



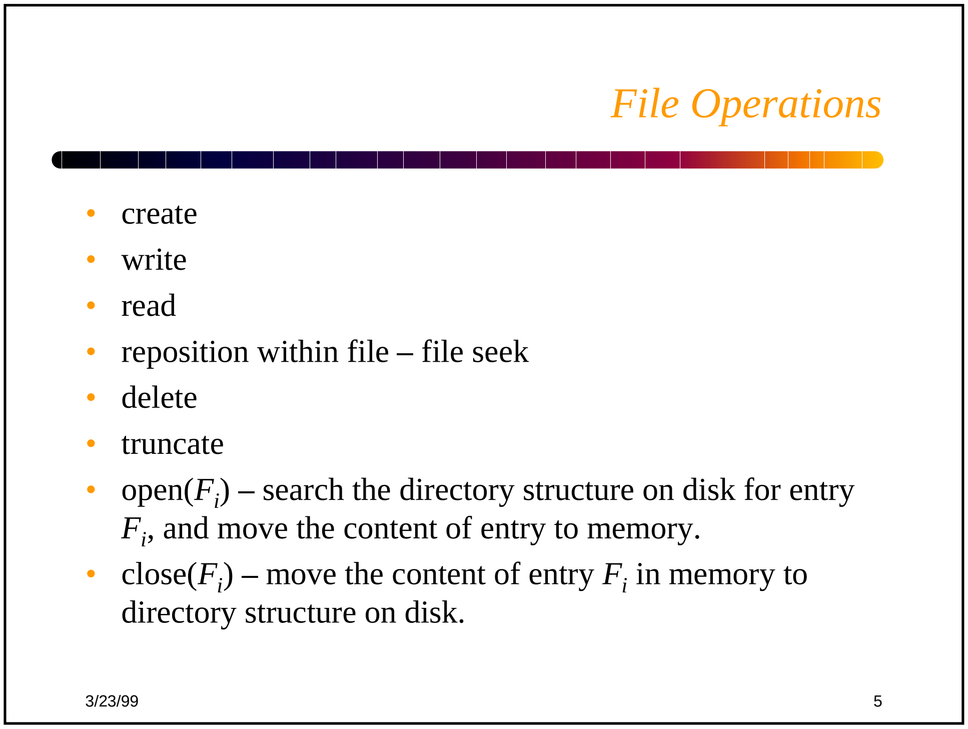
File Structure

- 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.
- **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

- create
- write
- read
- reposition within file – file seek
- 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.

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, p, pas f77, asm, a	Source code in various languages.
Batch	bat, sh	Collections of commands to the command interpreter.
Text	txt, doc	Textual data, documents.
Word Processor	doc, wp, tex, rrf, …	Various work-processor formats
Library	lib, a	Libraries of routines
Print or View	ps, div, gif, …	ASCII or binary file
Archive	arc, zip, tar	Related files grouped into one file, sometimes compressed.

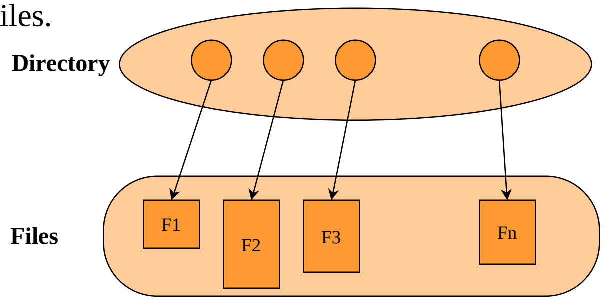
Access Methods

- 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

Directory Structure

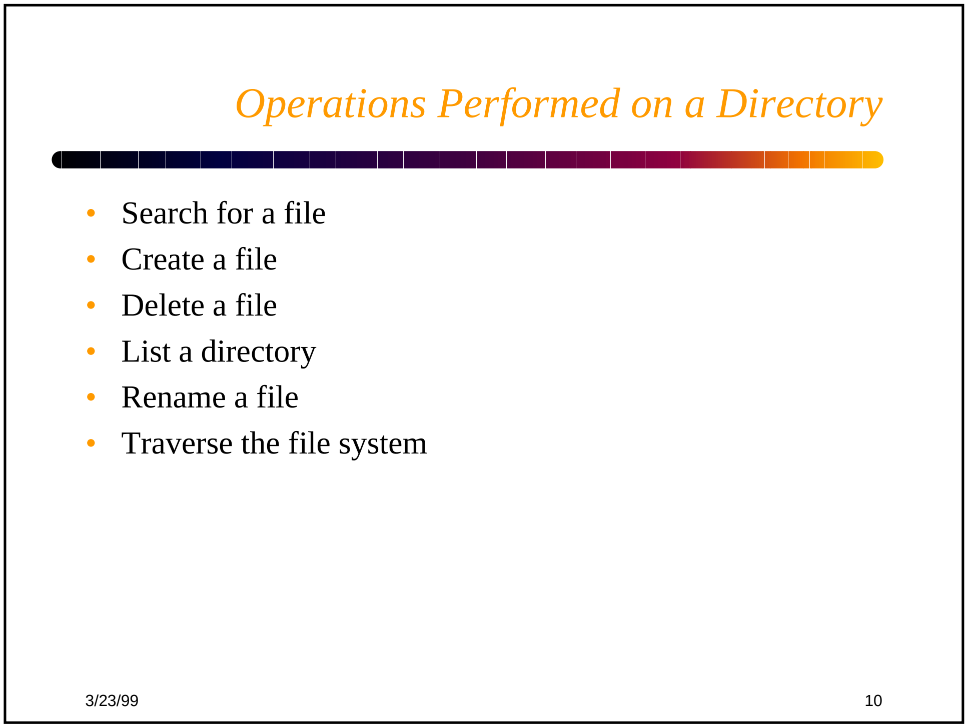
- A collection of nodes containing information about all files.



- Both the directory structure and the files reside on disk.
- Backups of these two structures are kept on tapes.

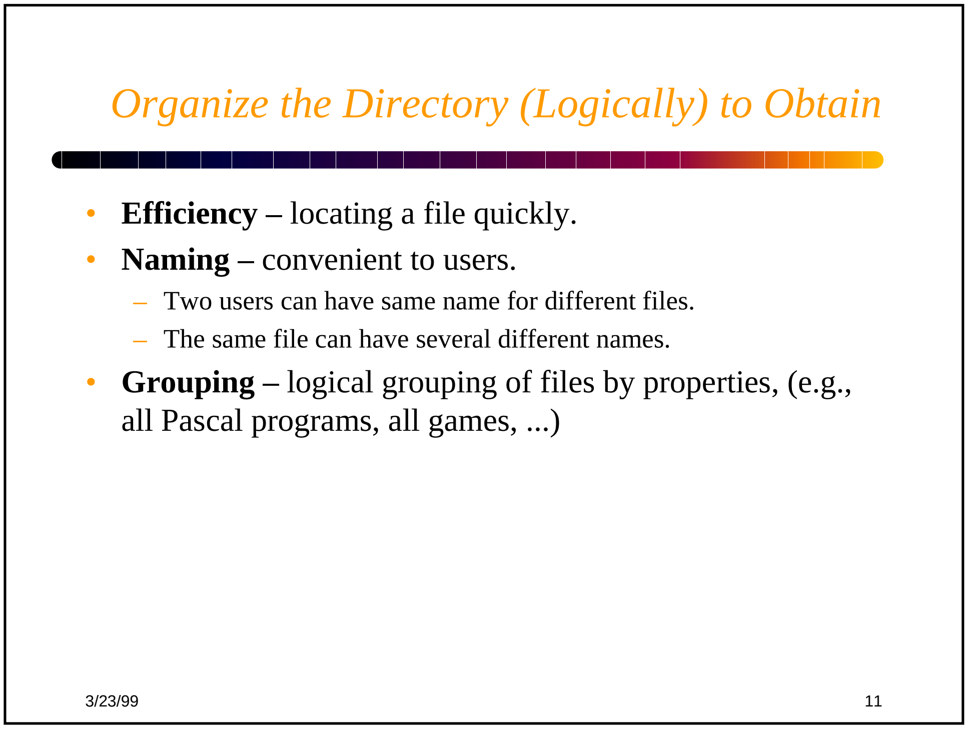
Information in a Device Directory

- Name
- Type
- Address
- Current length
- Maximum length
- Date last accessed (for archival)
- Date last updated (for dump)
- Owner ID (who pays)
- Protection information (discuss later)



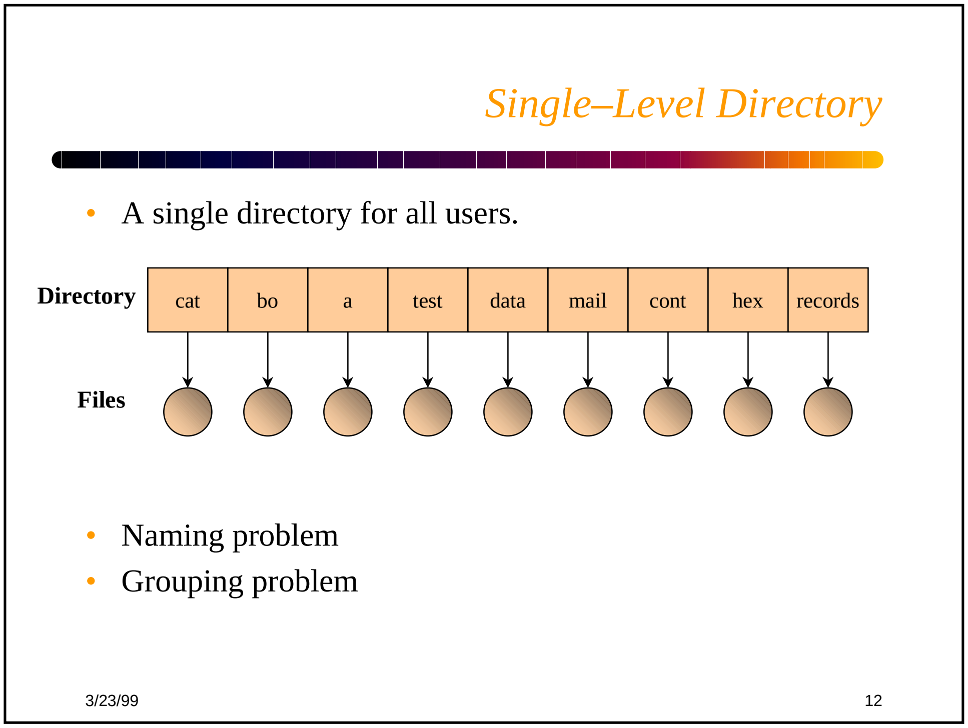
Operations Performed on a Directory

- 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

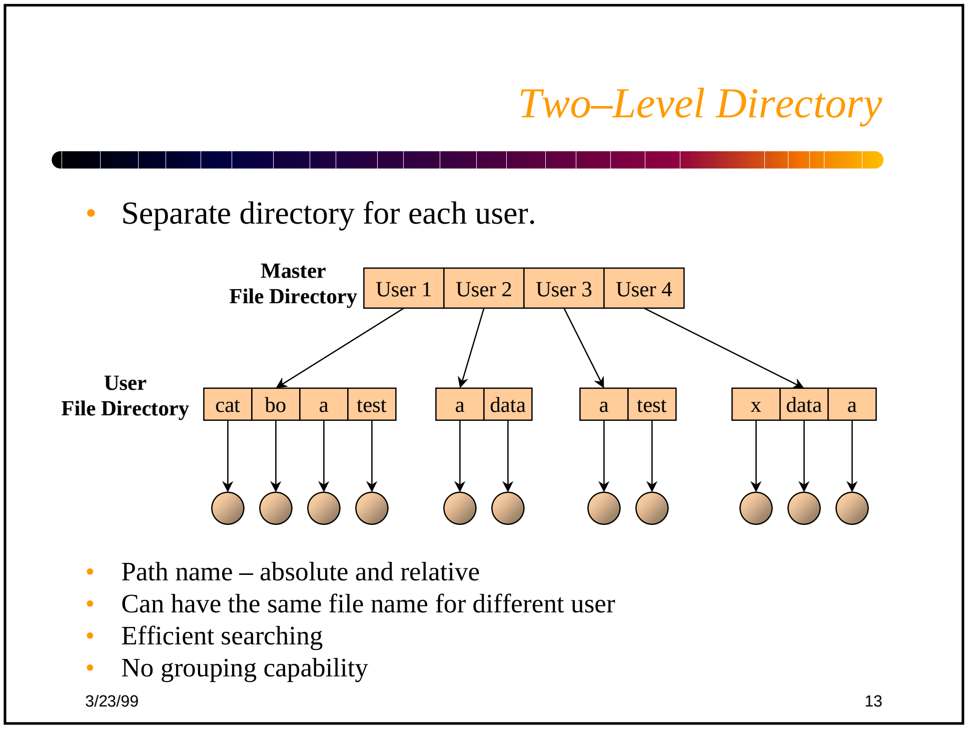
- **Efficiency** – locating a file quickly.
- **Naming** – convenient to users.
  - Two users can have same name for different files.
  - The same file can have several different names.
- **Grouping** – logical grouping of files by properties, (e.g., all Pascal programs, all games, ...)



Single–Level Directory

- A single directory for all users.

- Naming problem
- Grouping problem

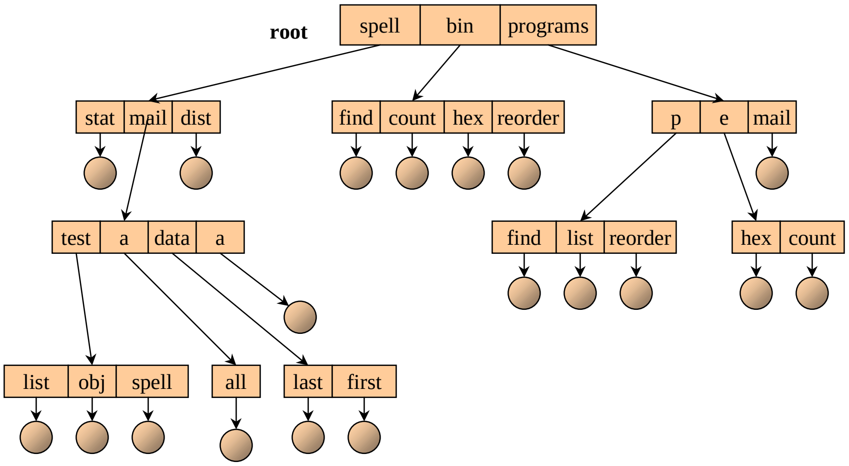


Two–Level Directory

- Separate directory for each user.

- Path name – absolute and relative
- Can have the same file name for different user
- Efficient searching
- No grouping capability

Tree–Structured Directories



root

spell bin programs

stat mail dist

find count hex reorder

p e mail

test a data a

find list reorder

hex count

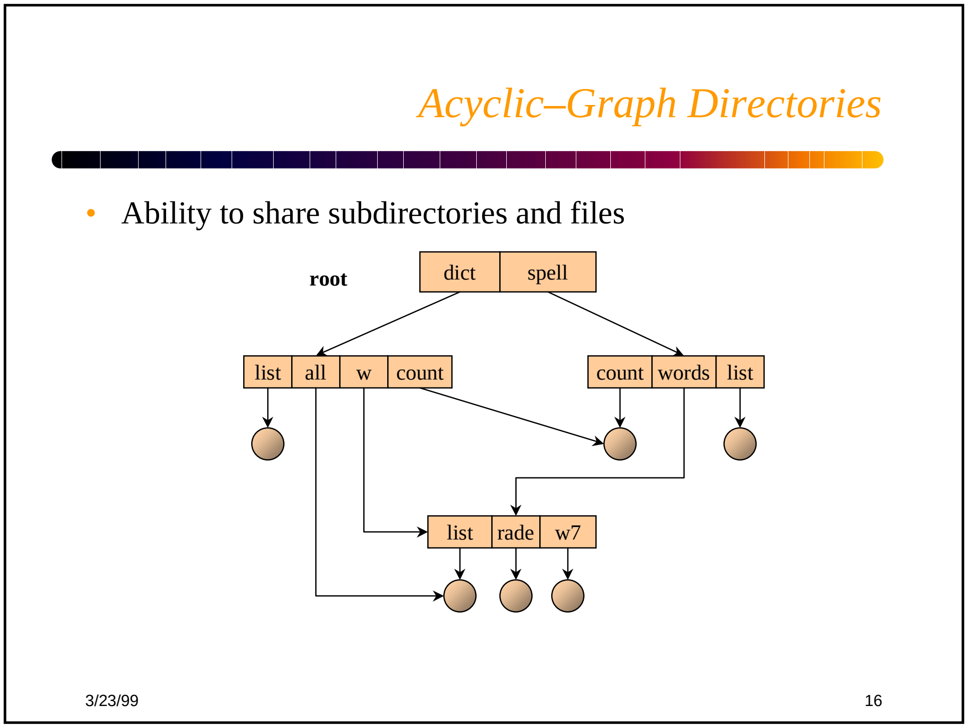
list obj spell

all

last first

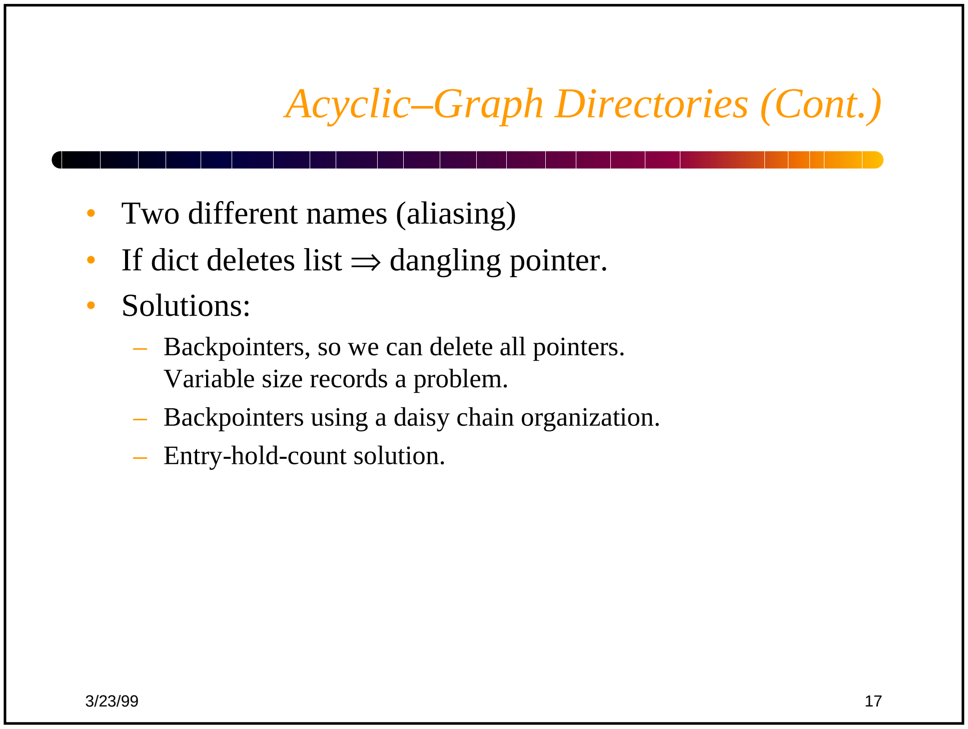
Tree–Structured Directories (Cont.)

- Efficient searching
- Grouping capability
- New concept of the current directory (working directory)
  - cd /spell/mail/prog
  - type list
- Absolute or relative path names
- Implicit relative operations
  - Create a file
  - Delete a file
  - Create a subdirectory
- Deletion semantics
  - Entire subtree or ensure empty subtree



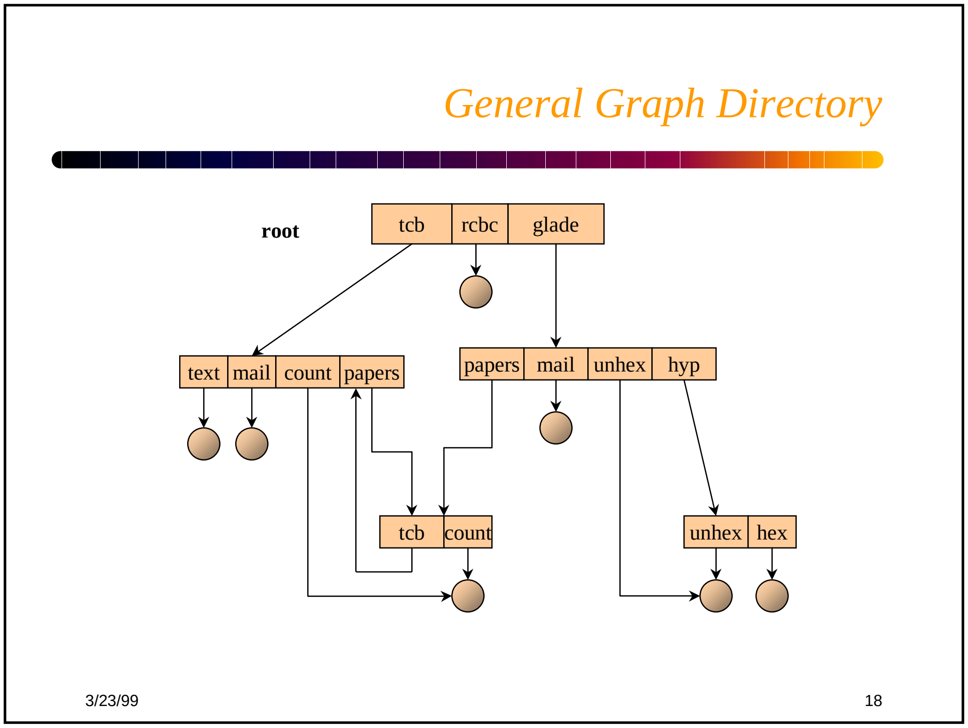
Acyclic–Graph Directories

- Ability to share subdirectories and files



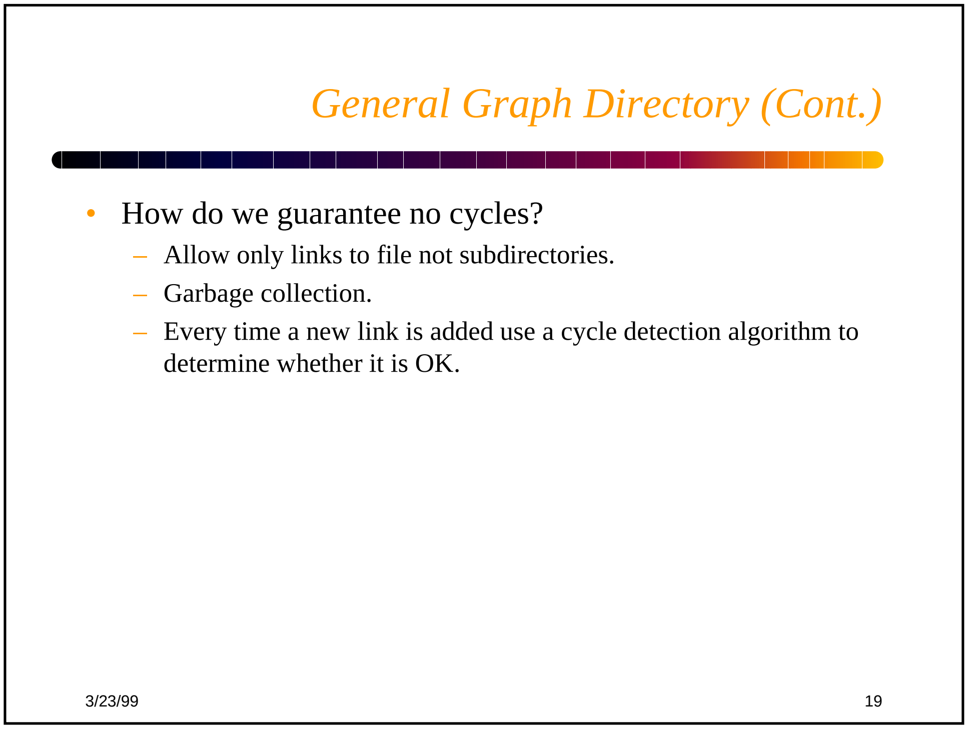
Acyclic–Graph Directories (Cont.)

- Two different names (aliasing)
- If dict deletes list ⇒ 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.



General Graph Directory

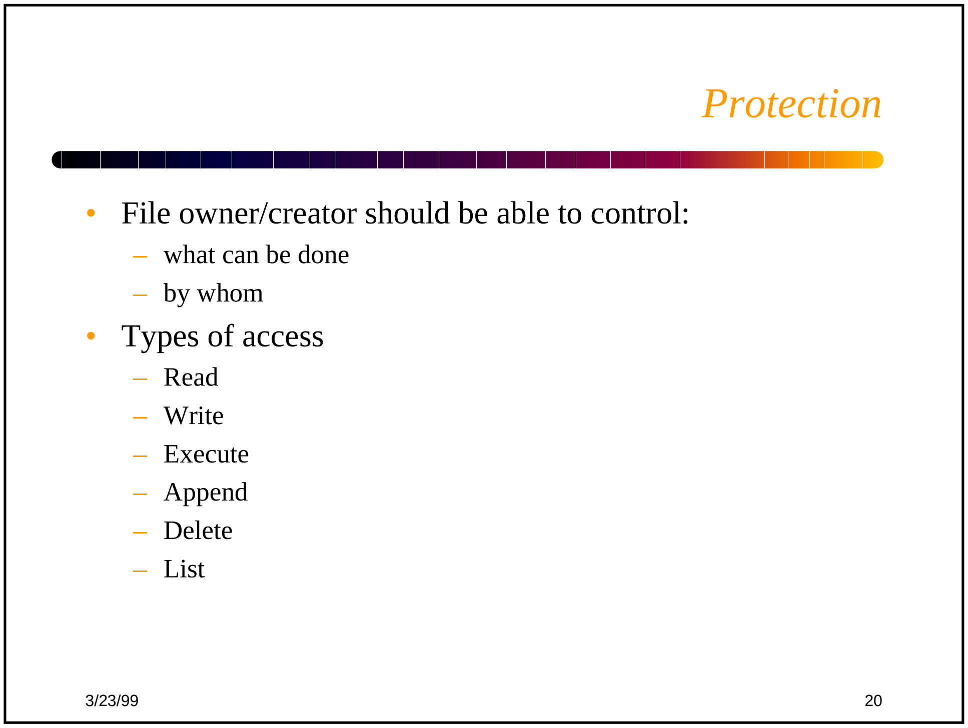
3/23/99 18



General Graph Directory (Cont.)

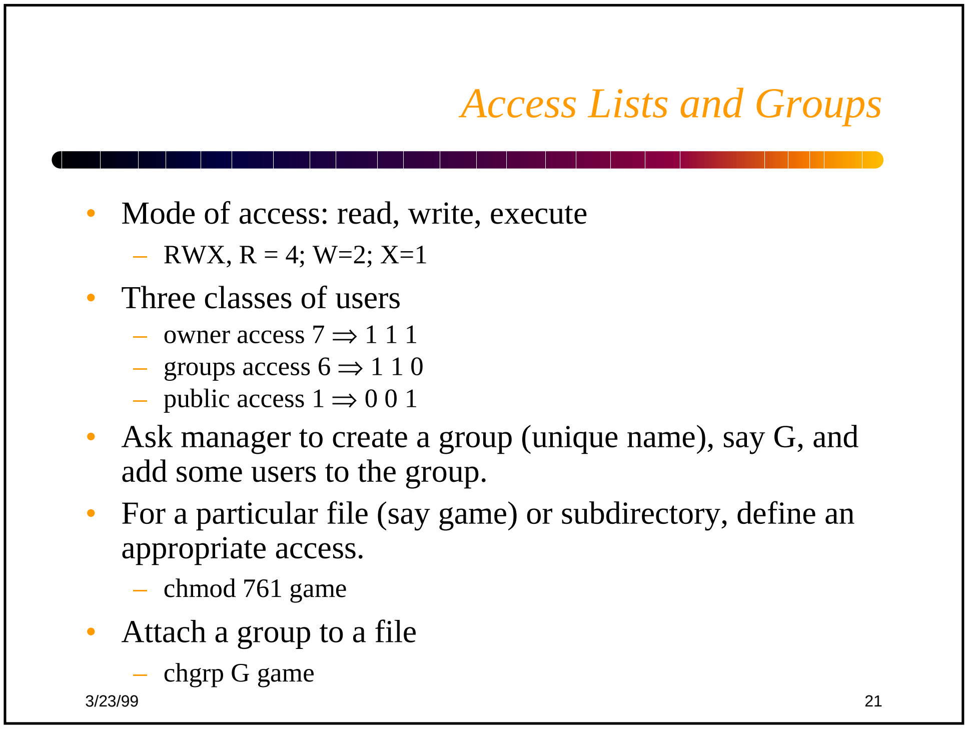
- How do we guarantee no cycles?
  - Allow only links to file not subdirectories.
  - Garbage collection.
  - Every time a new link is added use a cycle detection algorithm to determine whether it is OK.

3/23/99 19



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

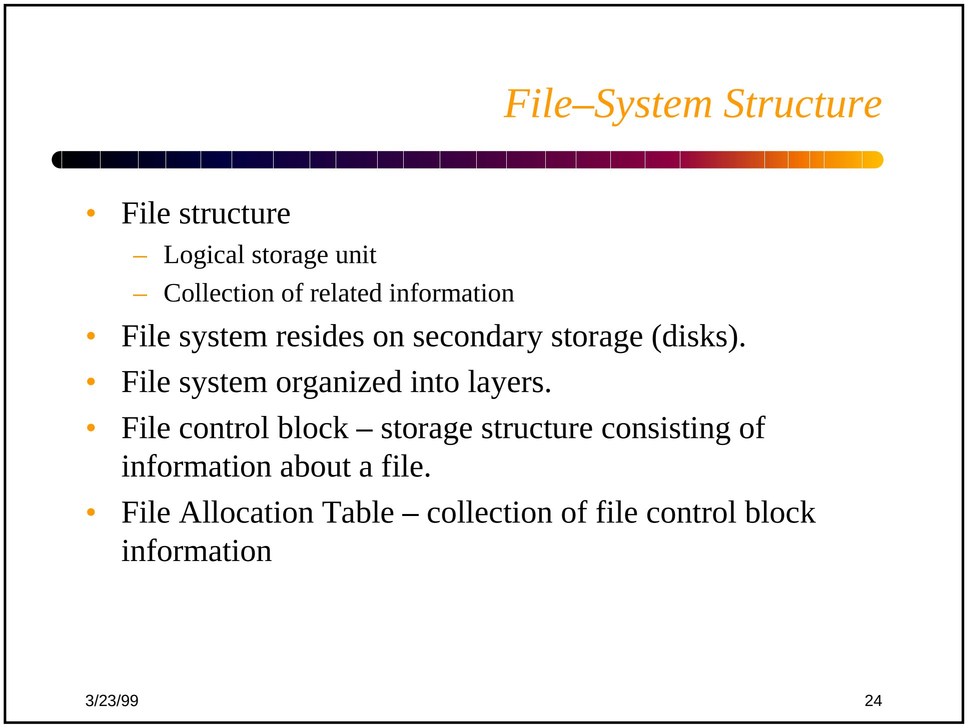
- Mode of access: read, write, execute
  - RWX, R = 4; W=2; X=1
- Three classes of users
  - owner access 7 ⇒ 1 1 1
  - groups access 6 ⇒ 1 1 0
  - public access 1 ⇒ 0 0 1
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say game) or subdirectory, define an appropriate access.
  - chmod 761 game
- Attach a group to a file
  - chgrp G game

Consistency Semantics

- Specify “what happens” when multiple users access a shared file concurrently:
- File Session – set of operations bracketed by open and close.
- Unix Semantics
  - writes to a file are visible to concurrent sessions
  - common file pointer sharing
- Session Semantics
  - writes to a file are not visible to concurrent sessions
  - Upon a close, updates are visible to successor sessions
- Immutable Shared File semantics

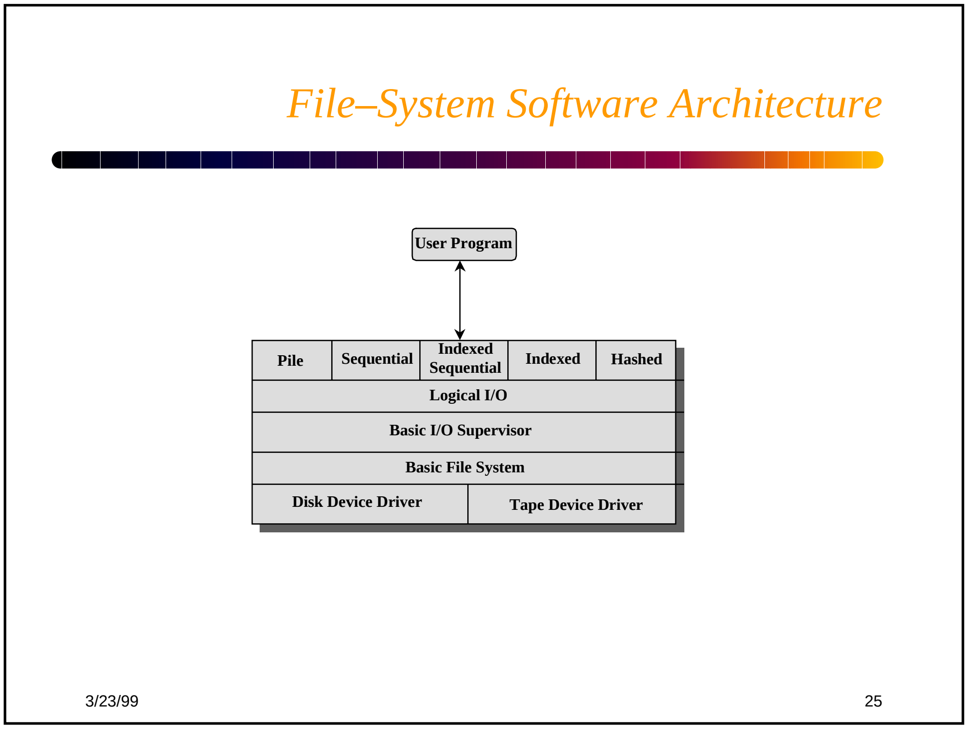
File–System Implementation

- File-System Structure
- Allocation Methods
- Free-Space Management
- Directory Implementation
- Efficiency and Performance
- Recovery



File–System Structure

- File structure
  - Logical storage unit
  - Collection of related information
- File system resides on secondary storage (disks).
- File system organized into layers.
- File control block – storage structure consisting of information about a file.
- File Allocation Table – collection of file control block information



File–System Software Architecture

User Program

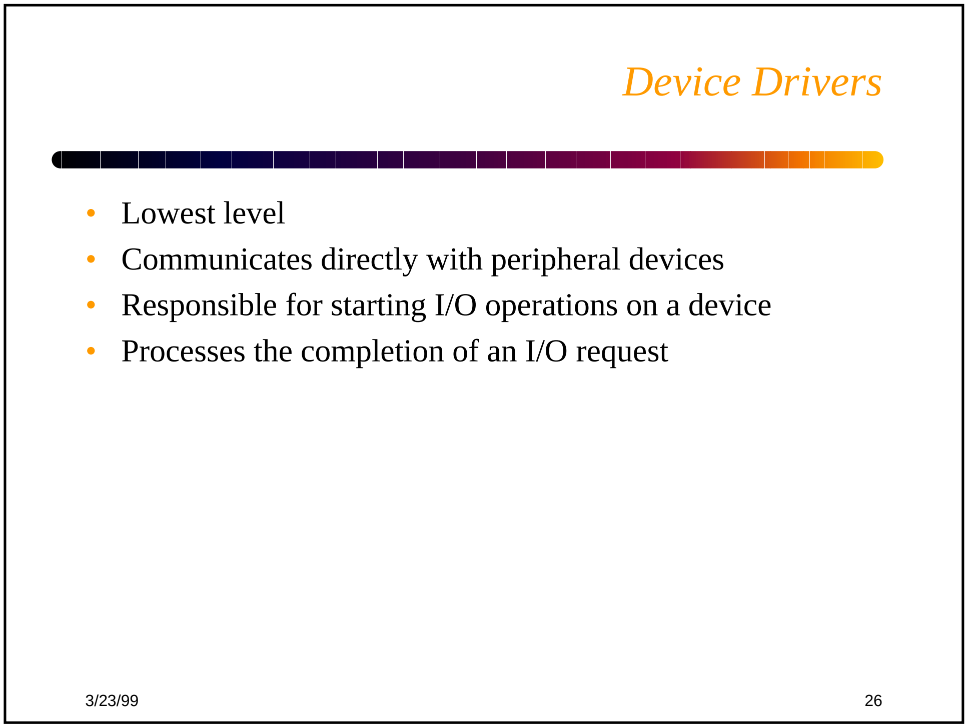
Pile | Sequential | Indexed Sequential | Indexed | Hashed

Logical I/O

Basic I/O Supervisor

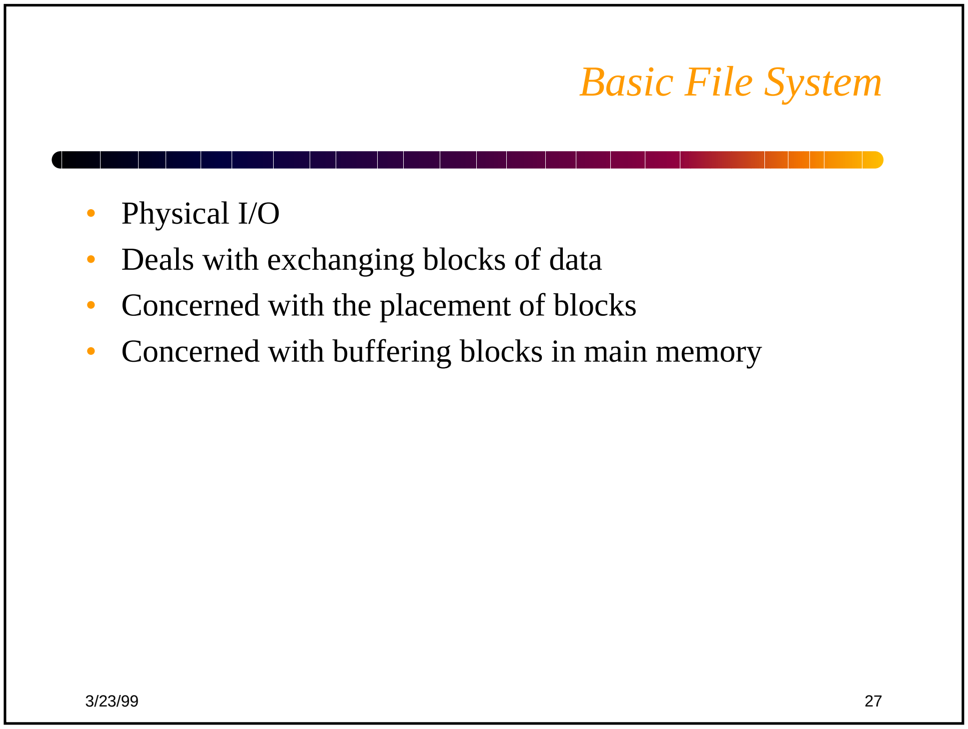
Basic File System

Disk Device Driver | Tape Device Driver



Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- Responsible for starting I/O operations on a device
- Processes the completion of an I/O request



Basic File System

- Physical I/O
- Deals with exchanging blocks of data
- Concerned with the placement of blocks
- Concerned with buffering blocks in main memory

Basic I/O Supervisor

- Responsible for file I/O initiation and termination
- Control structures are maintained
- Concerned with scheduling access to optimize performance
- Part of the operating system

Logical I/O

- Allows users and applications to access records
- Maintains basic data about file

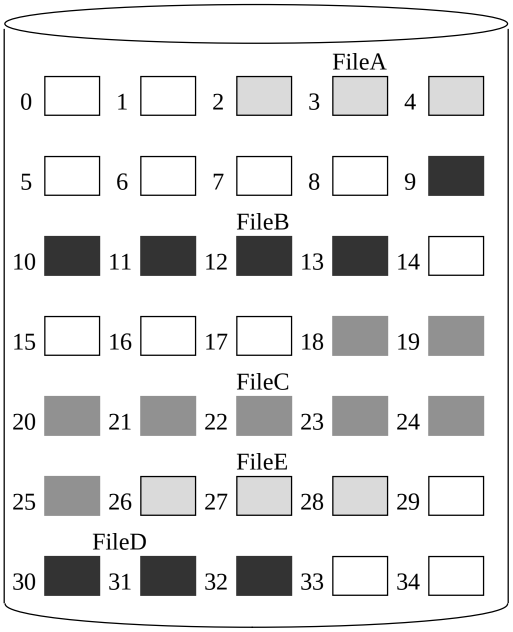
Access Method

- Reflect different file structures
- Different ways to store and process data

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple – only starting location (block #) and length (number of blocks) are required.
- Random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.
- Mapping from logical to physical.
- LA/512: Quotient Q, Remainder R
  - Block to be accessed = Q + starting address
  - Displacement into block = R

Contiguous File Allocation

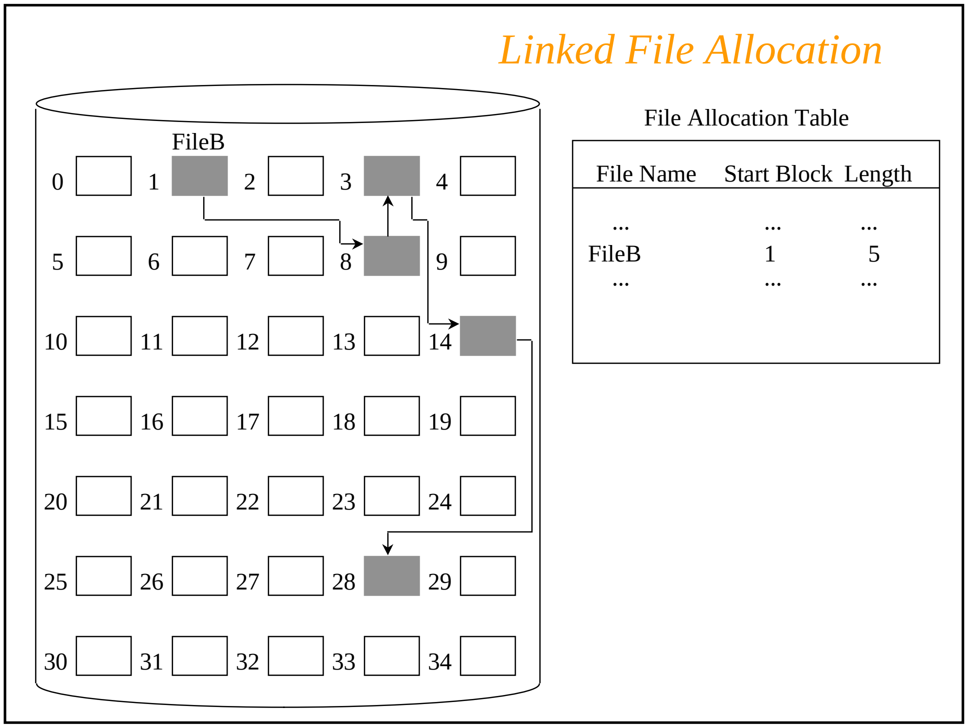


File Allocation Table

File Name	Start Block	Length
FileA	2	3
FileB	9	5
FileC	18	8
FileD	30	2
FileE	26	3

Linked Allocation

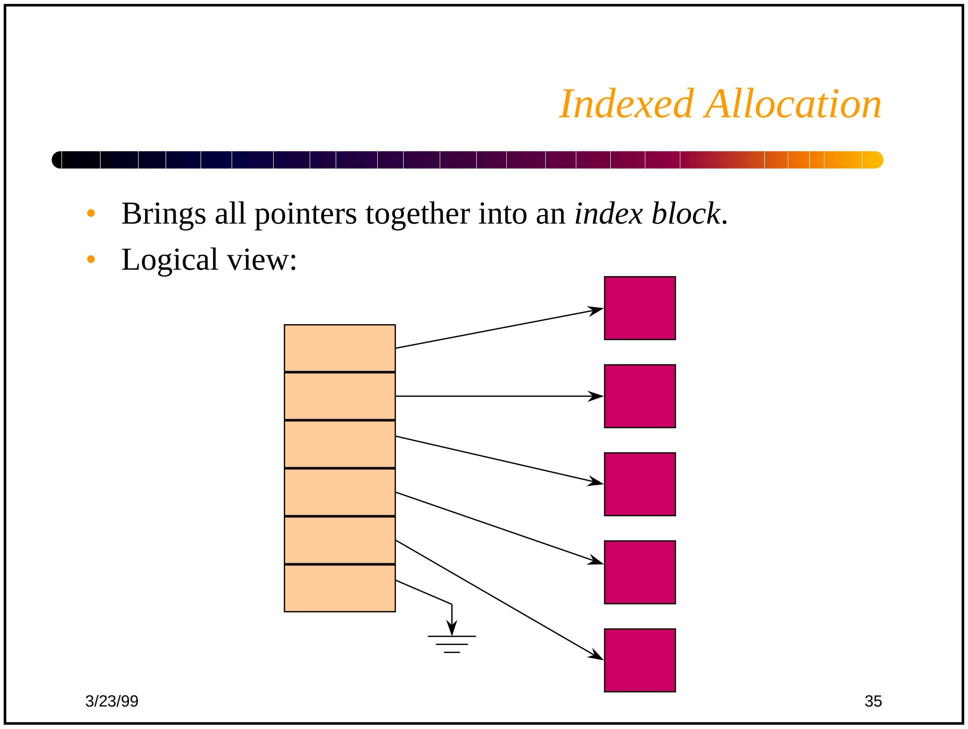
- Allocation on basis of individual block
- Each block contains a pointer to the next block in the chain
- Only single entry in the file allocation table
  - starting block and length of file
- No fragmentation
- Any free block can be added to the chain
- No accommodation of the principle of locality – no random access.
- LA/511 Quotient Q; Remainer R
  - Block to be accessed is the Qth block in the linked chain of blocks representing the file.
  - Displacement into block = R+1



Linked File Allocation

File Allocation Table

File Name	Start Block	Length
...	...	...
FileB	1	5
...	...	...



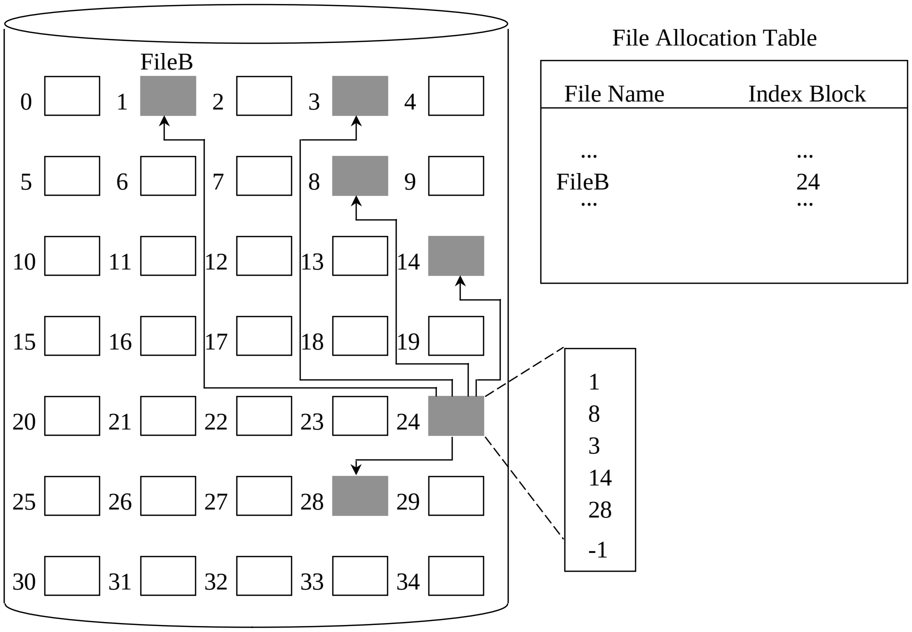
Indexed Allocation

- Brings all pointers together into an *index block*.
- Logical view:

3/23/99

35

Indexed Allocation with Block Portions



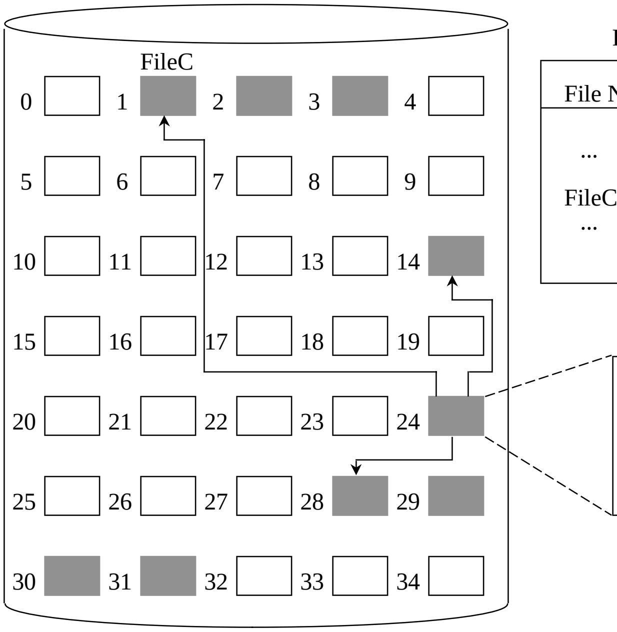
File Allocation Table

File Name	Index Block
…	…
FileB	24
…	…

Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.
- Mapping from logical to physical in a file of maximum size of 256K words and block size of 512 words. We need only 1 block for index table.
- LA/512
  - Q = displacement into index table
  - R = displacement into block

Indexed Allocation - Var Length Portions



File Allocation Table

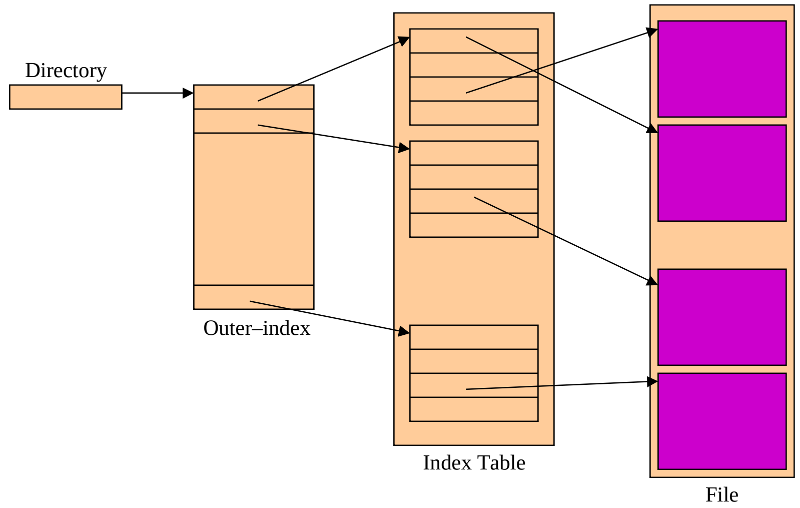
File Name	Index Block
…	…
FileC	24
…	…

Start Block	Length
1	3
28	4
14	1

Indexed Allocation – Mapping (Cont.)

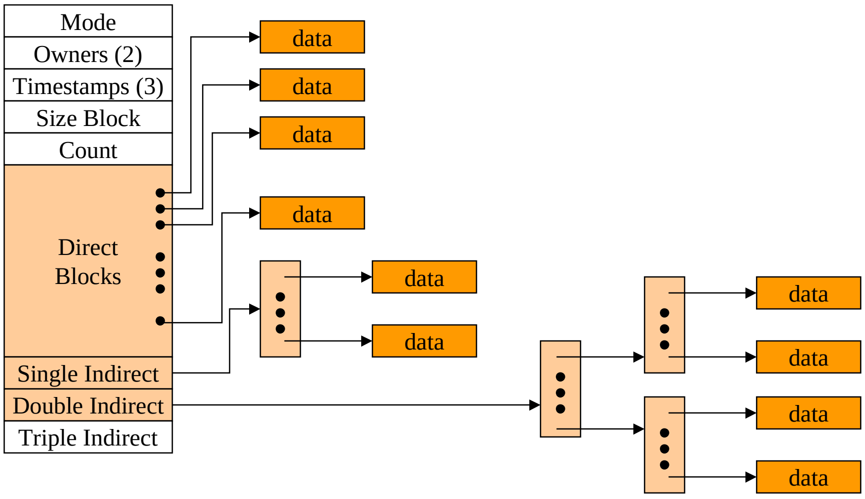
- Mapping from logical to physical in a file of unbounded length (block size of 512 words).
- Linked scheme -- Link blocks of index tables (no limit on size).
- LA/(512 x 511)
  - Q 1 block of index table
  - R 1 is used as follows:
- R 1 / 512
  - Q 2 = displacement into block of index table
  - R 2 = displacement into block of file

Indexed Allocation – Two Level Index

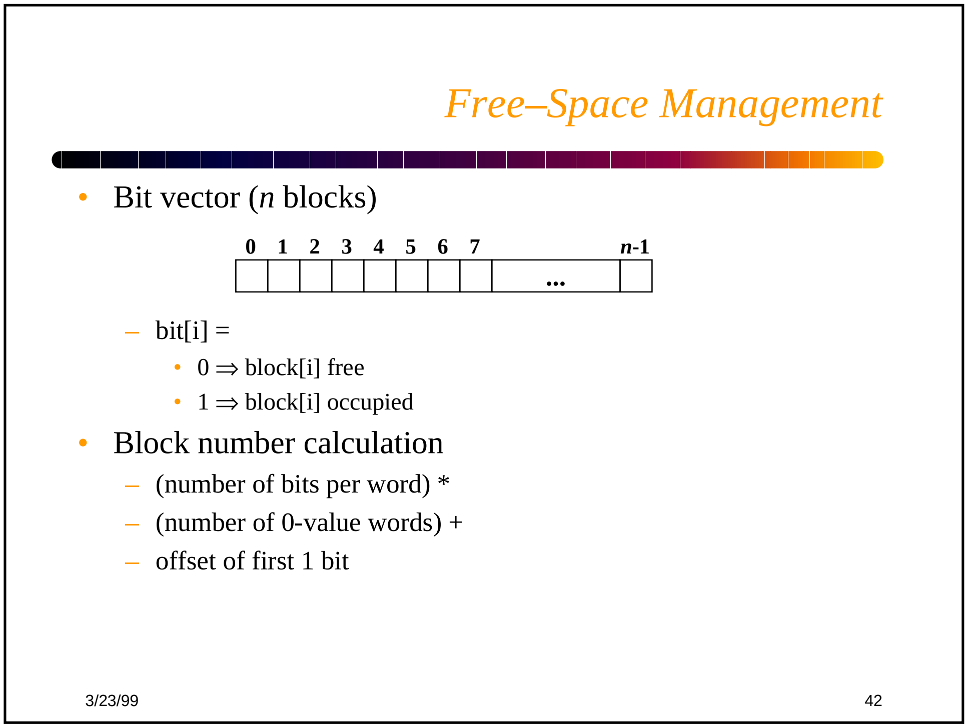


3/23/99 40

Unix File Allocation (4K bytes per block)



3/23/99 41

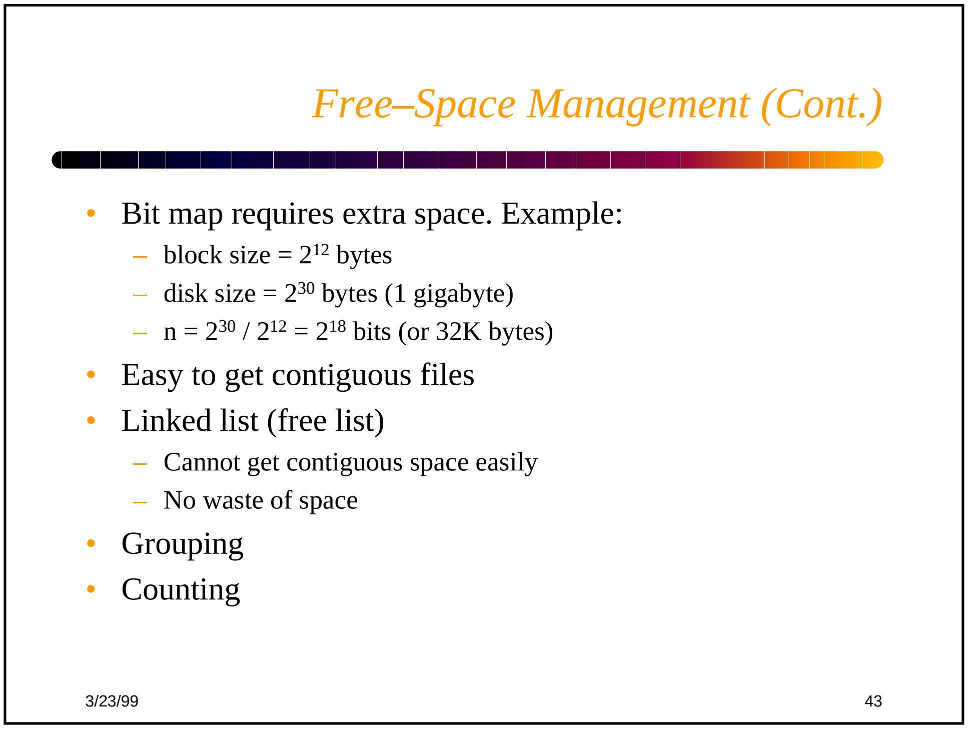


Free–Space Management

- Bit vector (*n* blocks)

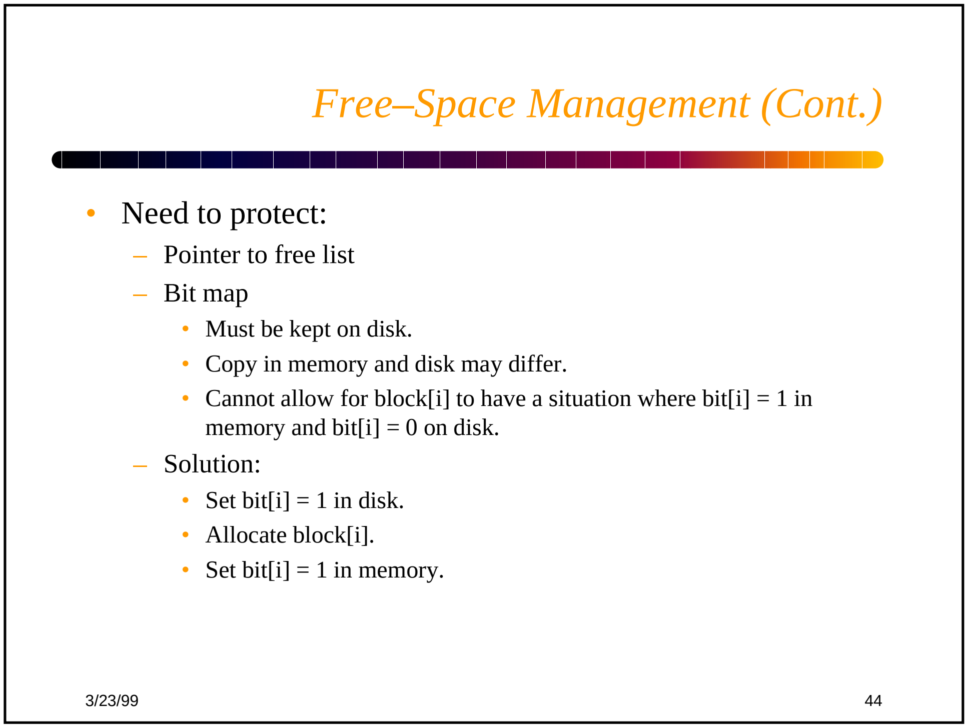
0	1	2	3	4	5	6	7		***n*-1**
								…	

  - bit[i] =
    - 0 ⇒ block[i] free
    - 1 ⇒ block[i] occupied
- Block number calculation
  - (number of bits per word) *
  - (number of 0-value words) +
  - offset of first 1 bit



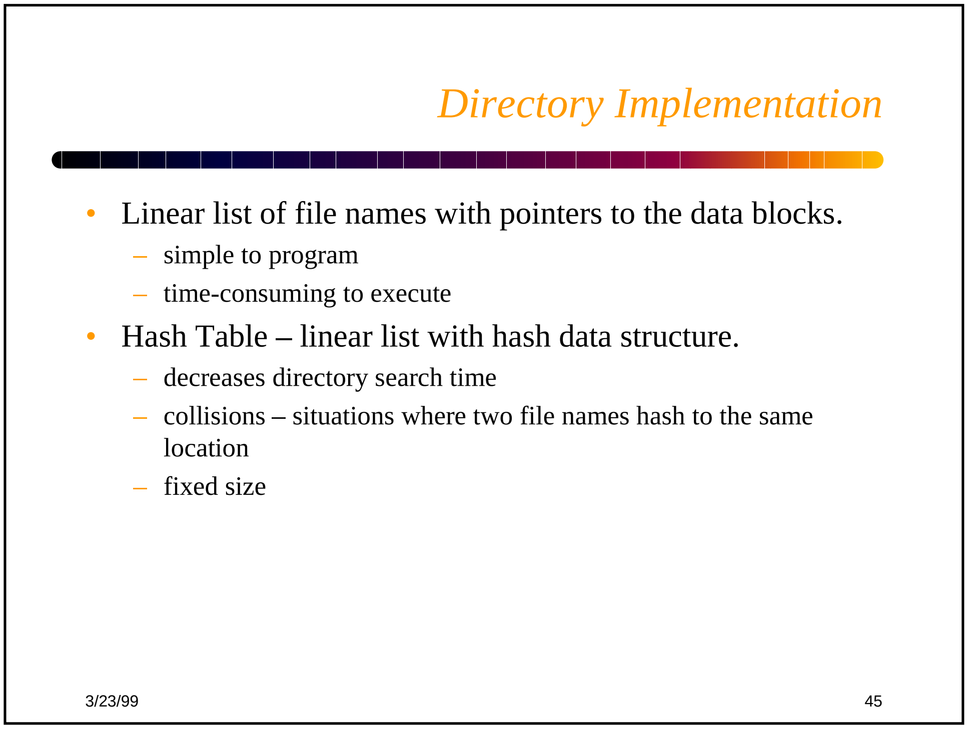
Free–Space Management (Cont.)

- Bit map requires extra space. Example:
  - block size = 2^{12} bytes
  - disk size = 2^{30} bytes (1 gigabyte)
  - n = 2^{30} / 2^{12} = 2^{18} bits (or 32K bytes)
- Easy to get contiguous files
- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
- Grouping
- Counting



Free–Space Management (Cont.)

- Need to protect:
  - Pointer to free list
  - Bit map
    - Must be kept on disk.
    - Copy in memory and disk may differ.
    - Cannot allow for block[i] to have a situation where bit[i] = 1 in memory and bit[i] = 0 on disk.
  - Solution:
    - Set bit[i] = 1 in disk.
    - Allocate block[i].
    - Set bit[i] = 1 in memory.



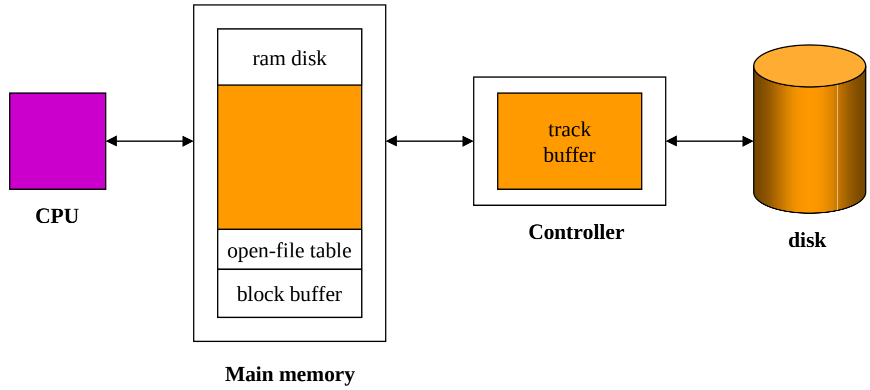
Directory Implementation

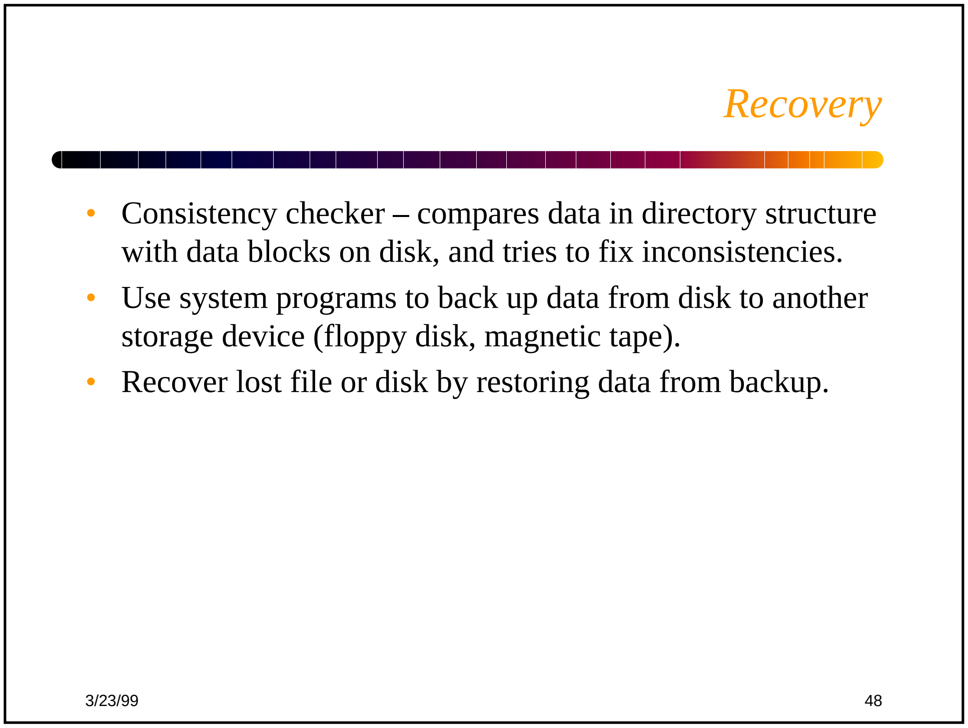
- Linear list of file names with pointers to the data blocks.
  - simple to program
  - time-consuming to execute
- Hash Table – linear list with hash data structure.
  - decreases directory search time
  - collisions – situations where two file names hash to the same location
  - fixed size

Efficiency and Performance

- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache – separate section of main memory for frequently used blocks
  - free-behind and read-ahead -- techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk

Various Disk–Caching Locations





Recovery

- Consistency checker – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies.
- Use system programs to back up data from disk to another storage device (floppy disk, magnetic tape).
- Recover lost file or disk by restoring data from backup.