Storage Management

File Concept

- File is a named collection of related information that is recorded on secondary storage.
- File is the smallest allotment of logical secondary storage; that is data cannot be written to secondary memory unless they are within a file.
- Contiguous logical address space
- Types of files:
  - Data
    - numeric
    - character
    - binary
  - Program (both source & object forms)

File Structure

- File types also can be used to indicate structure of a file.
- Files must confirm to a required structure that is understood by the OS.
- For example, the OS requires that an executable file have a specific structure so that it can determine where in memory to load the file and what the location of the first instruction is.

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**
- **Append**
- **Rename**

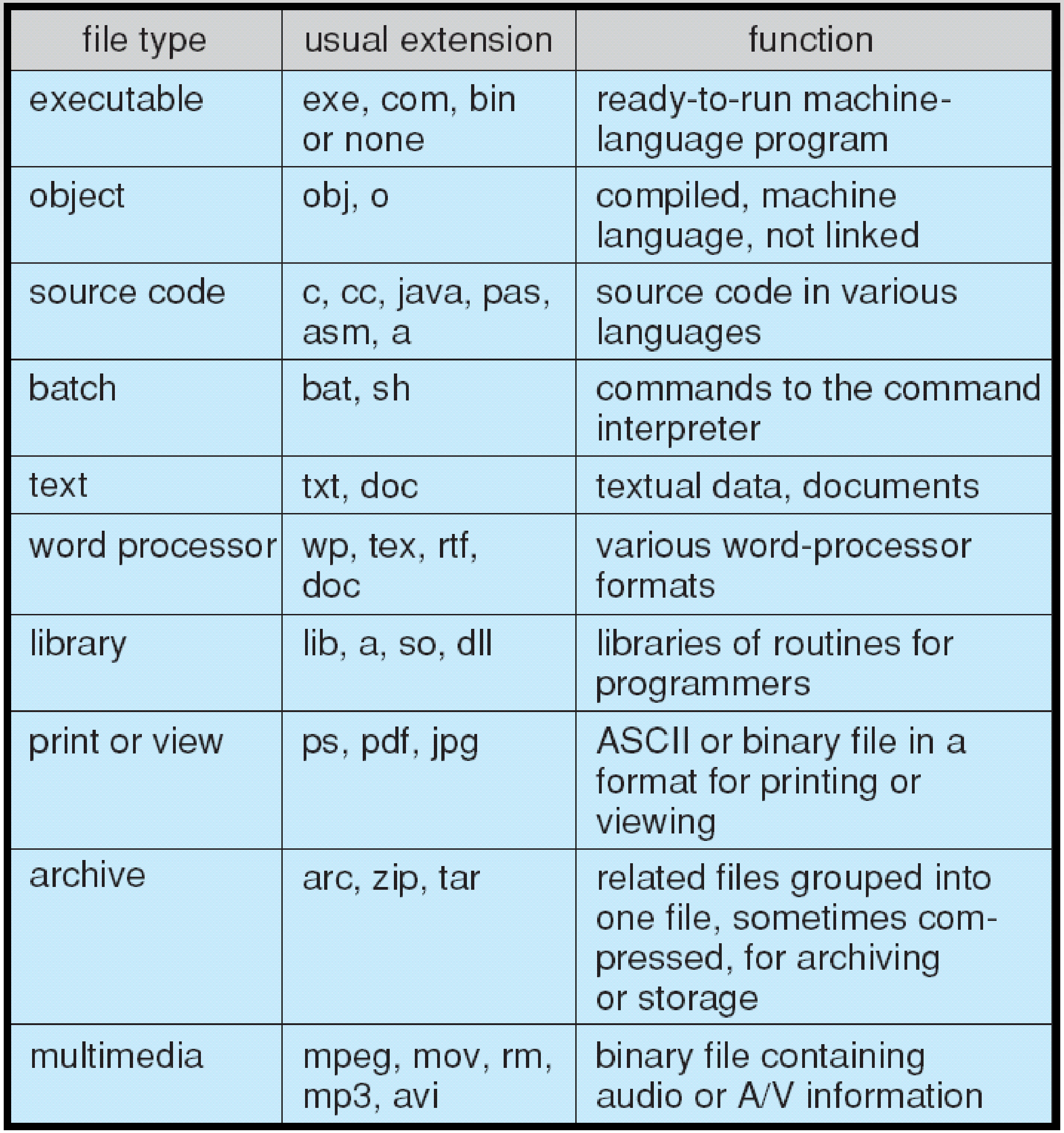
Open Files

- 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:
    - When a file is opened then an entry is made in open-file table.
    - Count of number of times a file is open – to allow removal of data from open-file table when last process closes it
  - 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. Similar to Exclusive or writer lock.
  - **Advisory** – processes can find status of locks and decide what to do. Similar to Shared lock or reader lock.

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

- When a file is used, the information must be accessed and read into computer memory.

- The information in the file is accessed in several ways.
  - Sequential Access
  - Direct Access

Sequential Access

- Information in the file is processed in order, one record after the other.

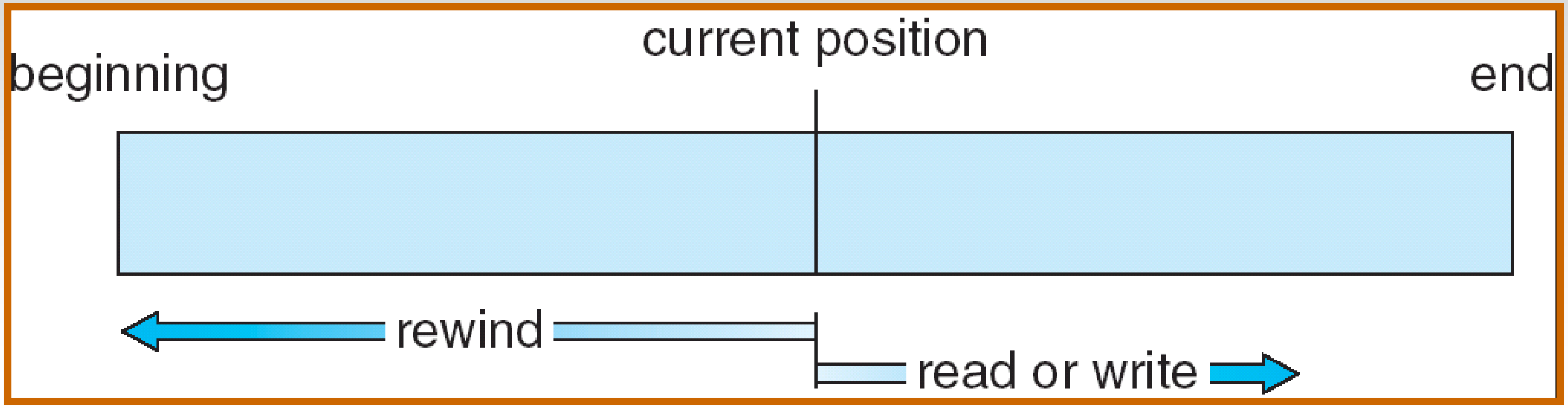
Example:

Editors and compilers access files in this fashion.

Sequential Access

- Read (read next)
  - Reads the next portion of the file and automatically advances a file pointer.
- Write (write next)
  - Appends to the end of file and advances to the end of the newly written material.
- Reset
  - Program may be able to skip forward or backward n records for some integer n.

Sequential-access File



Direct Access

- Also called as Relative access.
- A file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order.
- For Direct access, the file is viewed as a numbered sequence of blocks or records. Thus we may read block 14 then 53 and then write block 7.
- There are no restrictions on order of reading or writing for a direct-access file.
- Example: databases

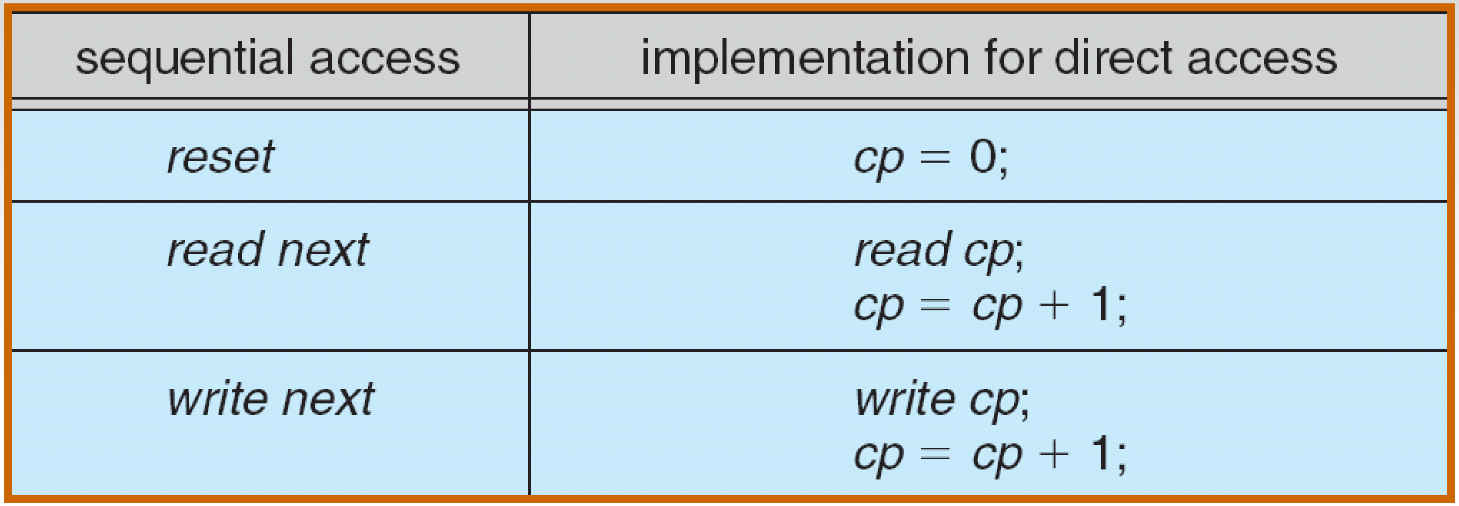
Direct Access

- The file operations are modified to include the block number as parameter.
  - Read n
  - Write n

where n is the block number.

- The block number provided by the user to OS is relative block number.(index relative to beginning of a file).

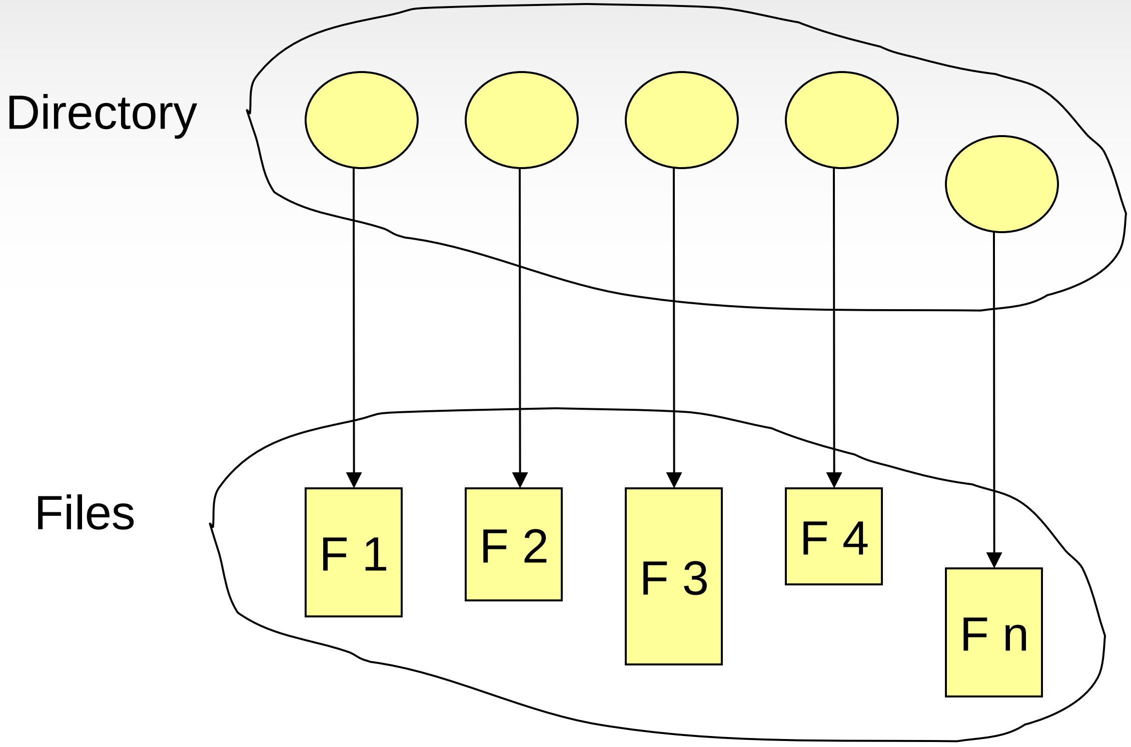
Simulation of Sequential Access on a Direct-access File



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

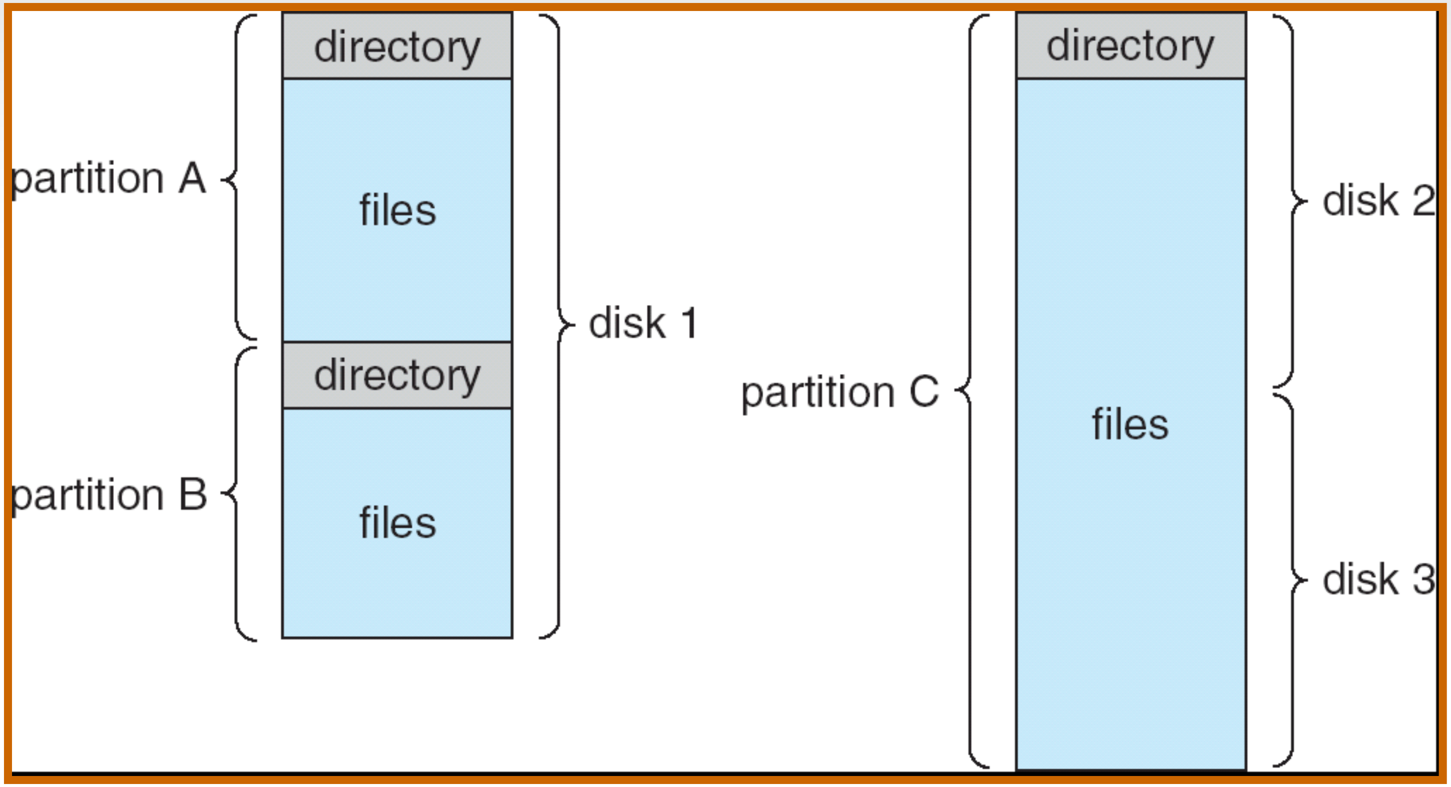
Directory Structure

A collection of nodes containing info about all the files



Both the directory structure and the files reside on disk

A Typical File-system Organization

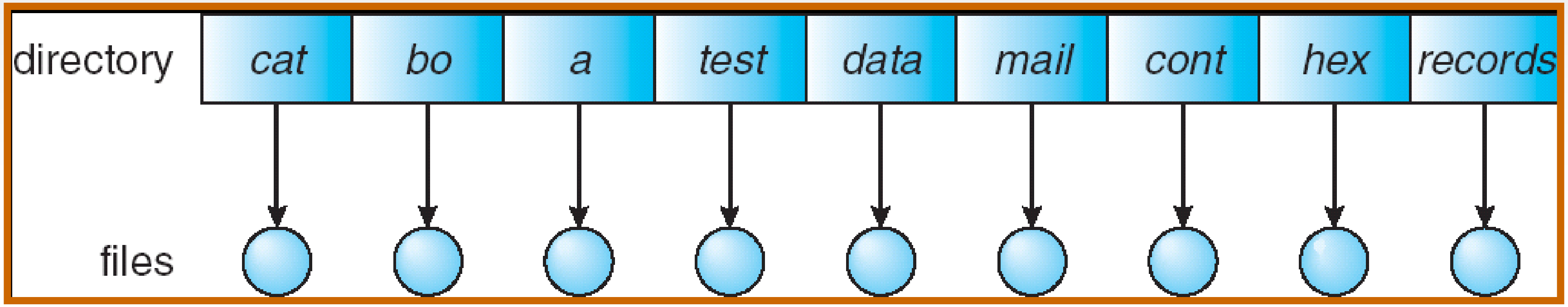


Organize the Directory (Logically)

- 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

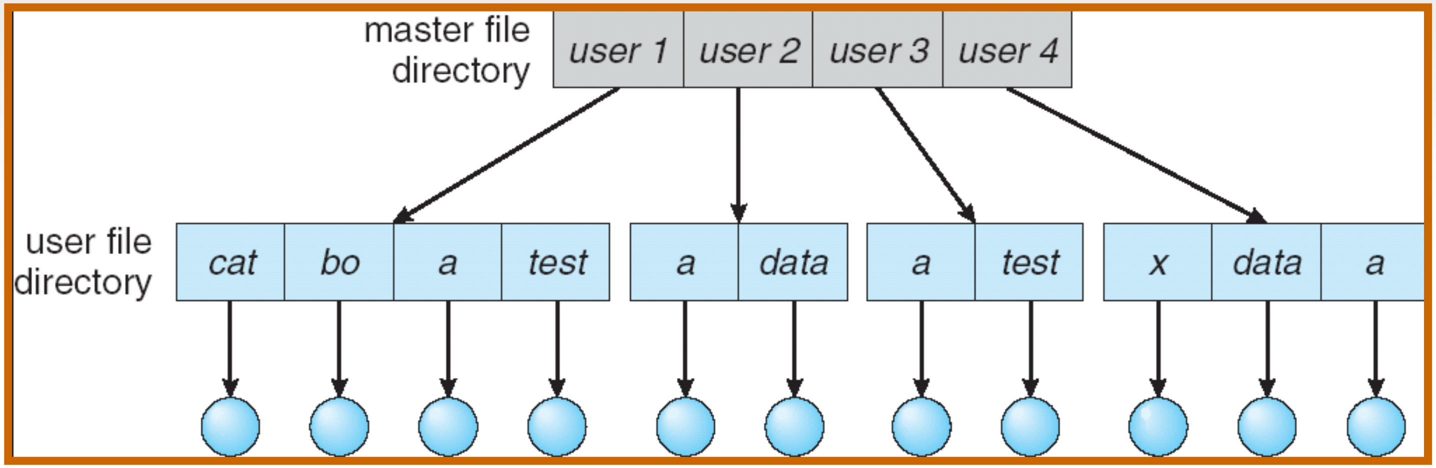


Naming problem

Grouping problem

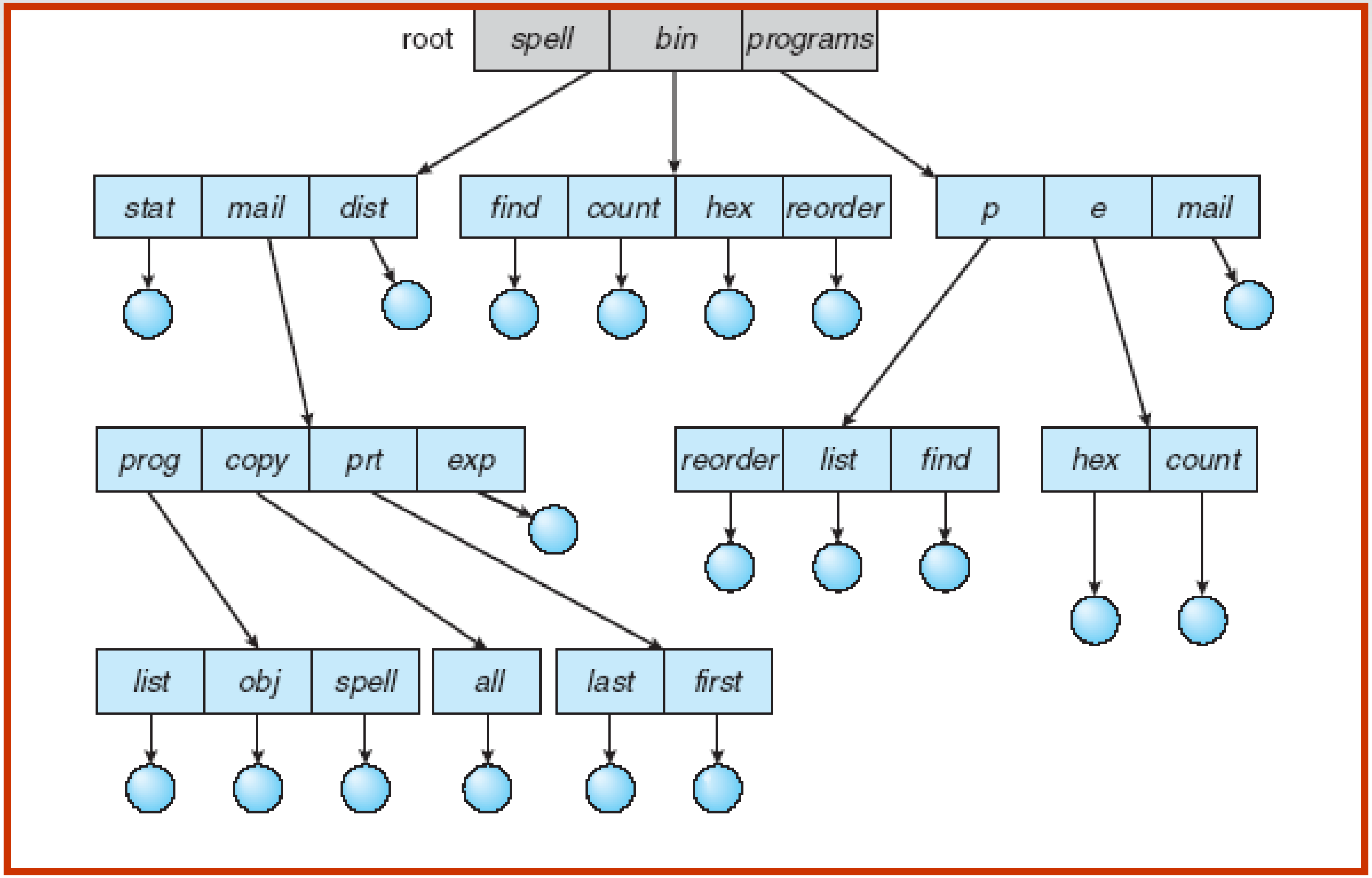
Two-Level Directory

- Separate directory for each user



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

Tree-Structured Directories



Tree-Structured Directories (Cont)

- Efficient searching
- Grouping Capability
- Current directory (working directory)

Tree-Structured Directories (Cont)

- **Absolute** or **relative** path name
- Creating a new file is done in current directory
- Delete a file

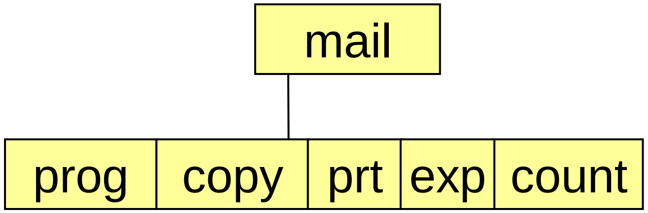
  rm <file-name>

- Creating a new subdirectory is done in current directory

  mkdir <dir-name>

  Example: if in current directory /mail

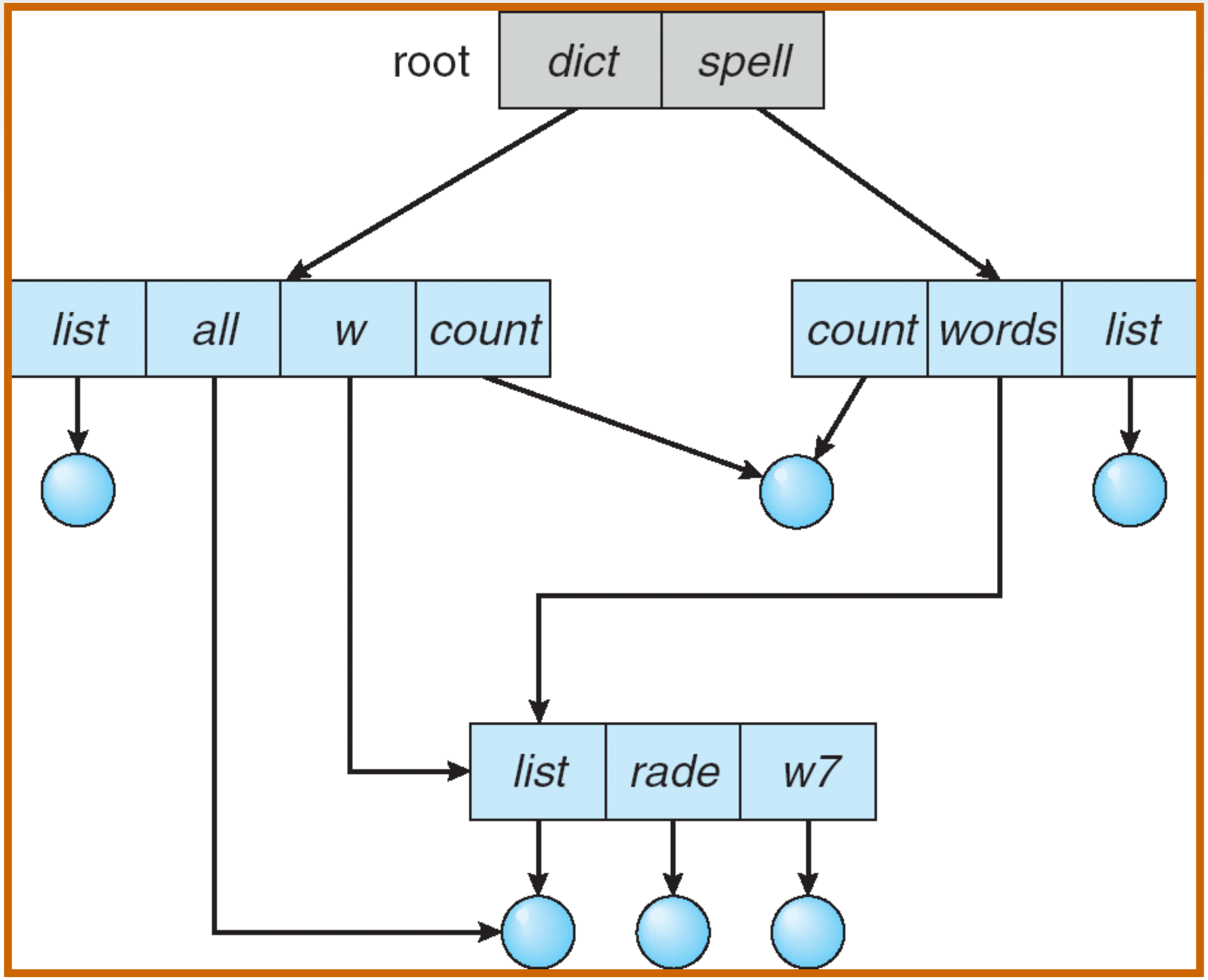
  mkdir count



Deleting “mail” ⇒ deleting the entire subtree rooted by “mail”

Acyclic-Graph Directories

- Have shared subdirectories and files



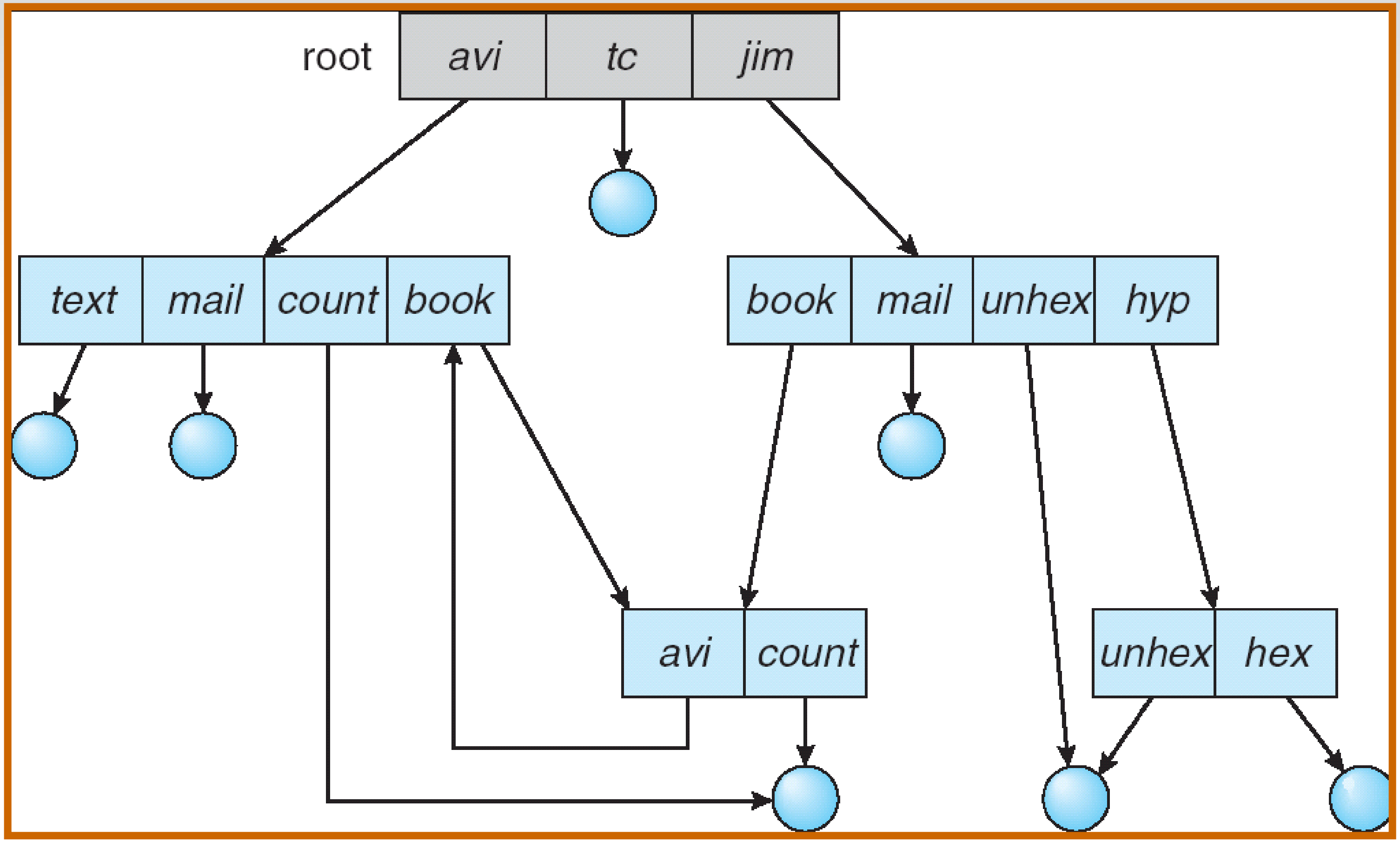
Acyclic-Graph Directories (Cont.)

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

  Solutions:

  - 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



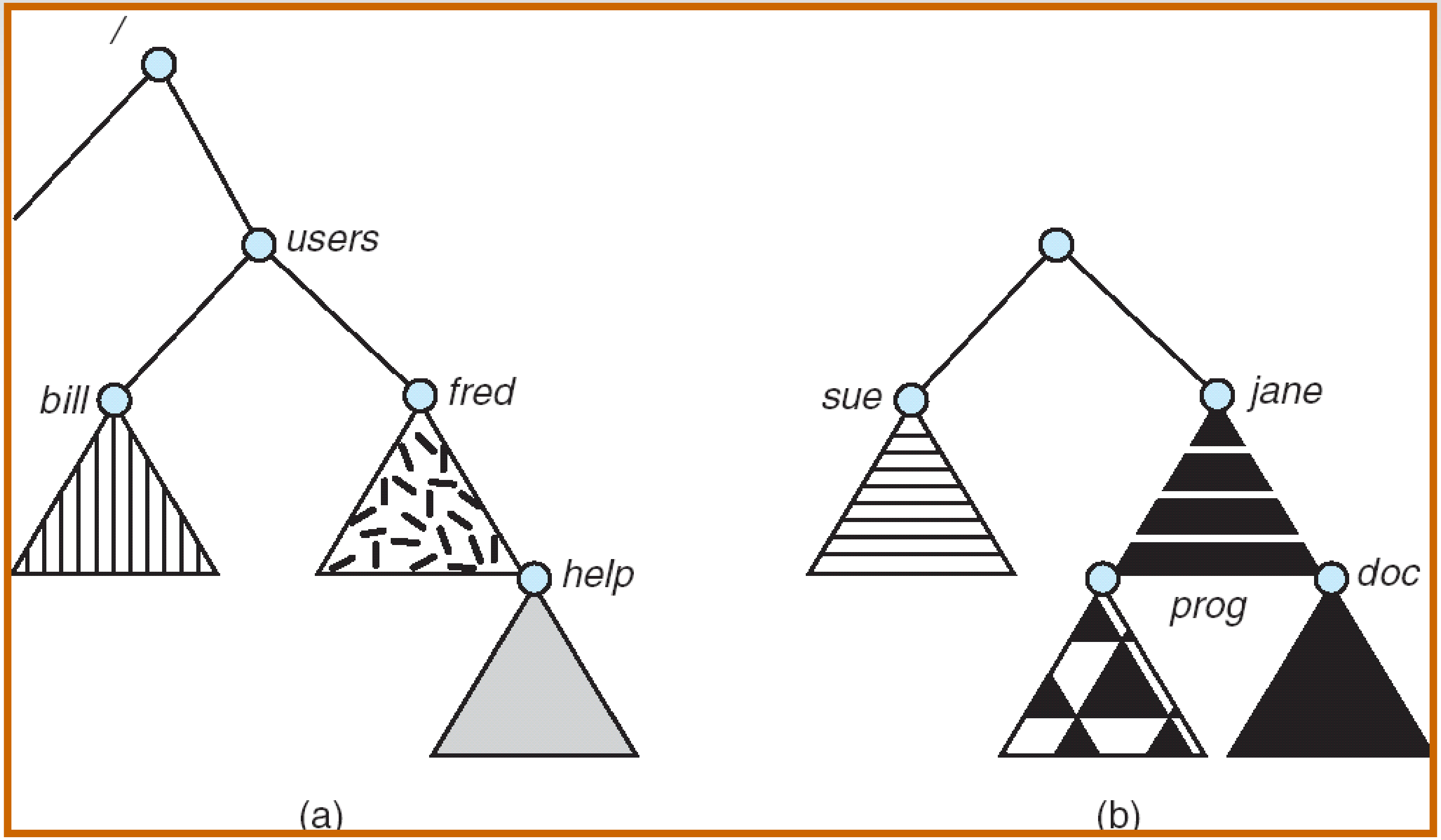
General Graph Directory (Cont.)

- 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

File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system is mounted at a **mount point**

(a) Existing. (b) Unmounted Partition



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

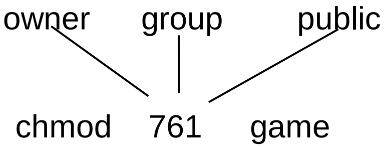
- Mode of access: read, write, execute
- Three classes of users

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.



```
chmod 761 game
```

Attach a group to a file

```
chgrp G game
```

Implementing File Systems

File System Structure

- Rather than transferring a byte at a time, to improve I/O efficiency, I/O transferred between memory and disk are performed in units of ***blocks***.

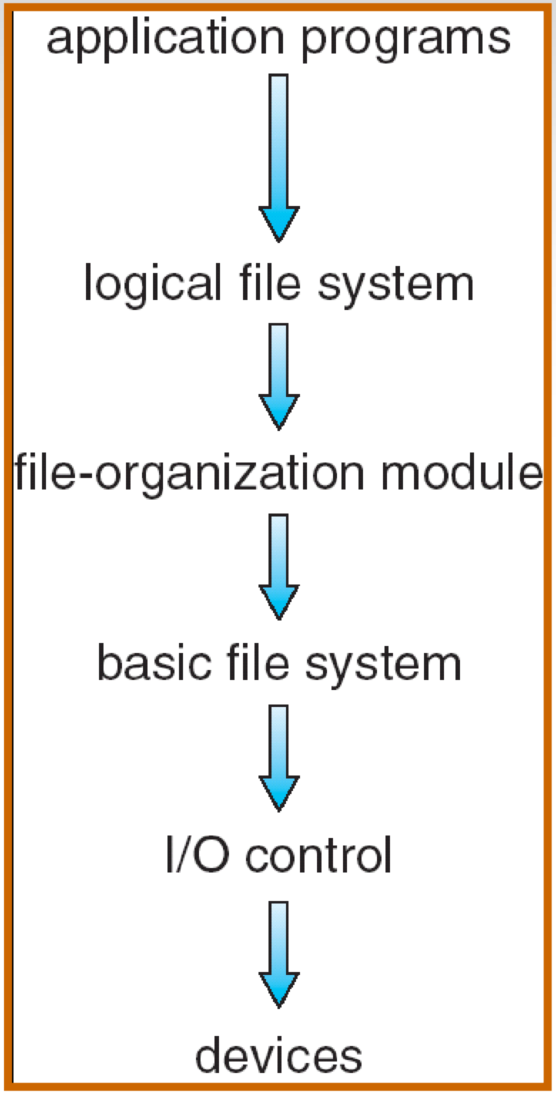
- Each block has one or more ***sectors***.

- Depending on the disk drive, sectors vary from 32 bytes to 4096 bytes; usually they are 512 bytes.

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

Layered File System


application programs

logical file system

file-organization module

basic file system

I/O control

devices

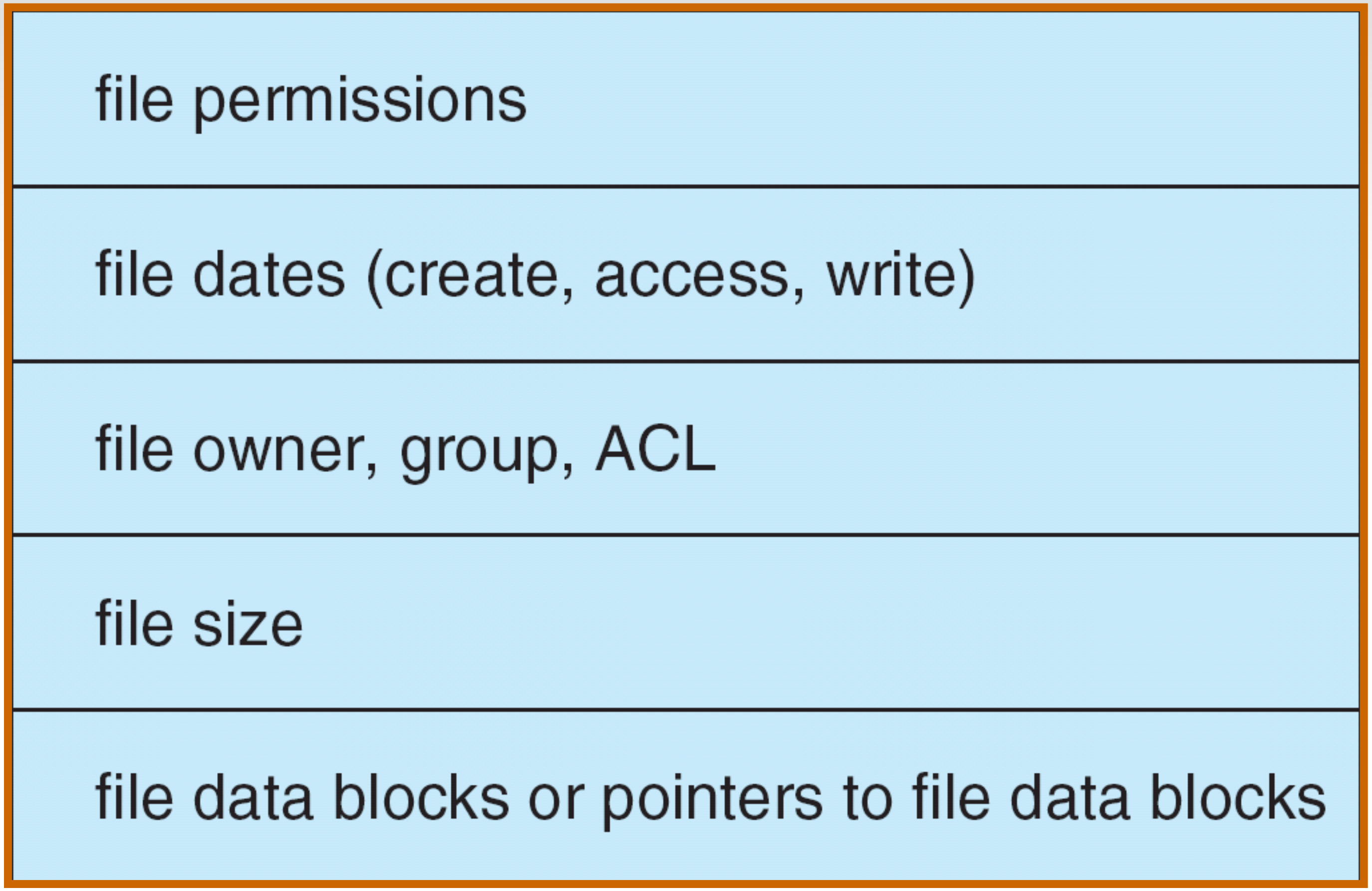
Layers

- Logical File System:
    - It manages metadata information. Metadata includes all of the file-system structure except the actual data(or contents of file).
- File-organization Module:
    - It knows about files and their logical blocks, as well as physical blocks.
    - By knowing the type of file allocation used and the location of file, this module can translate logical block addresses to physical block addresses for the basic file system to transfer.

Layers

- Basic File System:
  - Needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk.

File Control Block



file permissions

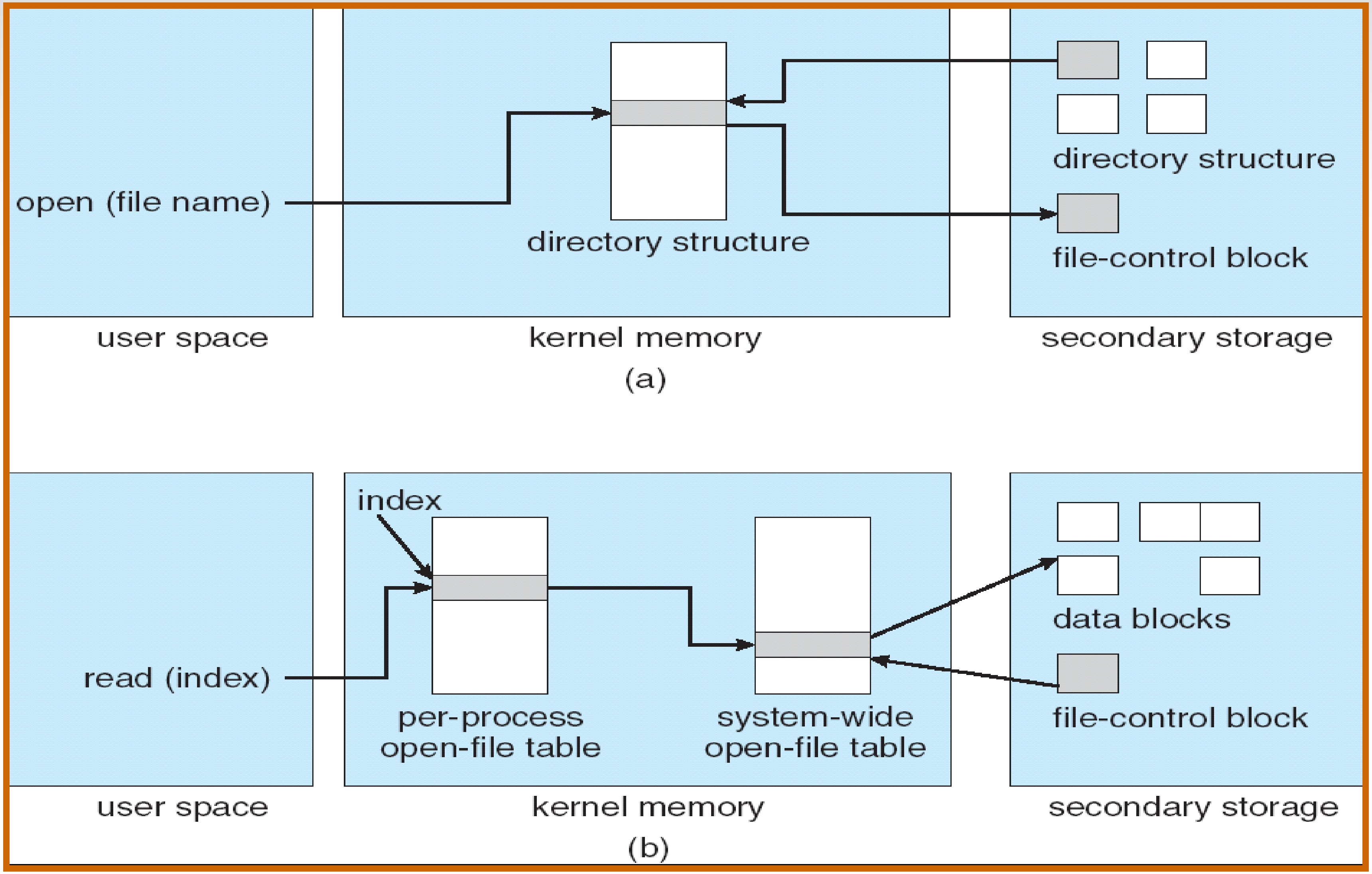
file dates (create, access, write)

file owner, group, ACL

file size

file data blocks or pointers to file data blocks

In-Memory File System Structures



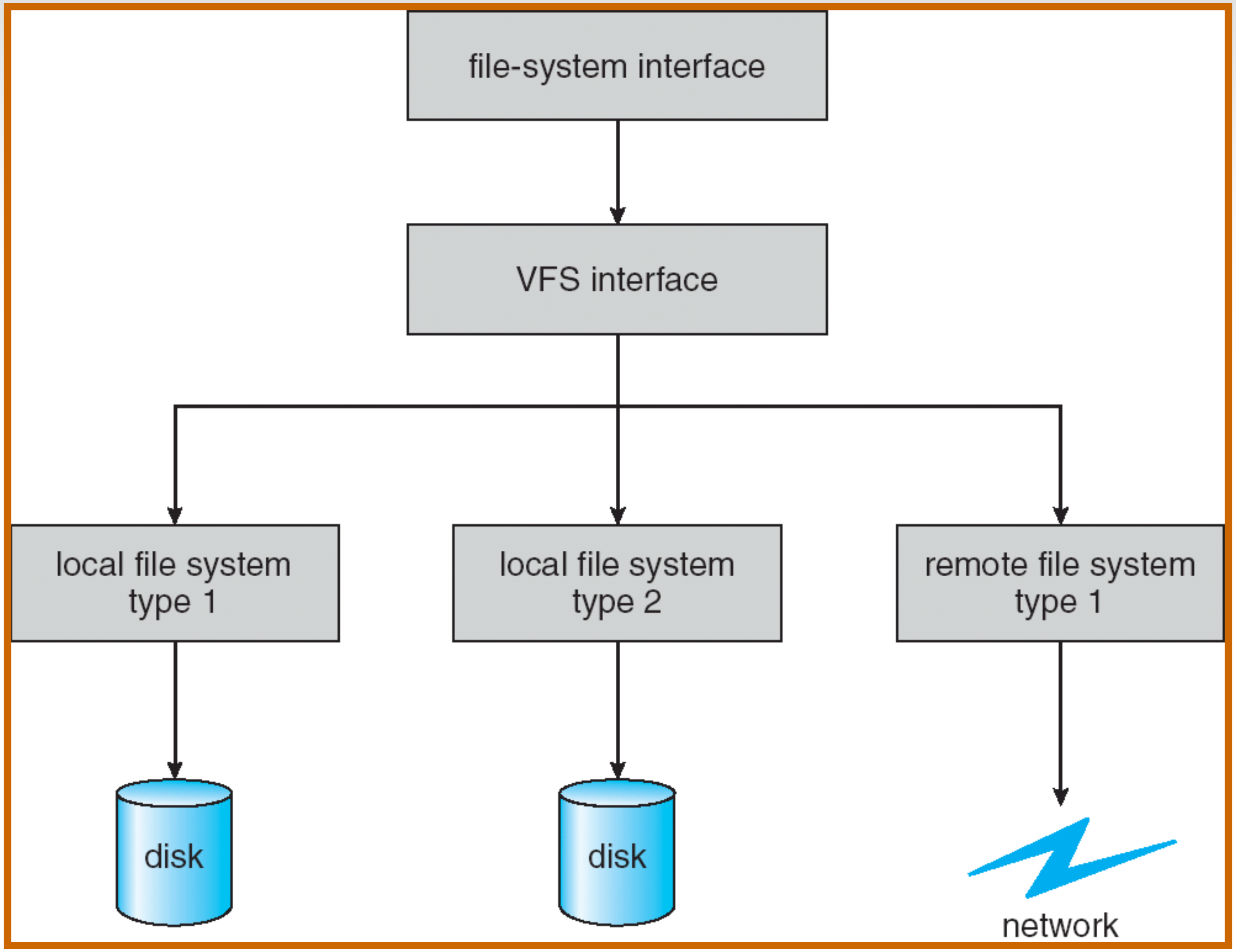
In-Memory File System Structures

- The open() system call first searches the system-wide open-file table to see if the file is already in use by other process.
- If it is already opened then a per-process open file table entry is created pointing to the existing system-wide open file table.
- If the file is not already open, the directory structure is searched for the given file name.
- Once the file is found, FCB is copied into a system-wide open file table in memory. This table not only stores the FCB but also tracks the number of processes that have the file open.

Virtual File Systems

- Virtual File Systems (VFS) provide an object-oriented way of implementing file systems.
- VFS allows the same system call interface (the API) to be used for different types of file systems.
- The API is to the VFS interface, rather than any specific type of file system.

Schematic View of Virtual File System



VFS Architecture in Linux

- Four main object types defined by Linux VFS are:
  - inode object – which represents an individual file.
  - file object – which represents an open file.
  - superblock object – which represents an entire file system.
  - dentry object – which represents an individual directory entry.

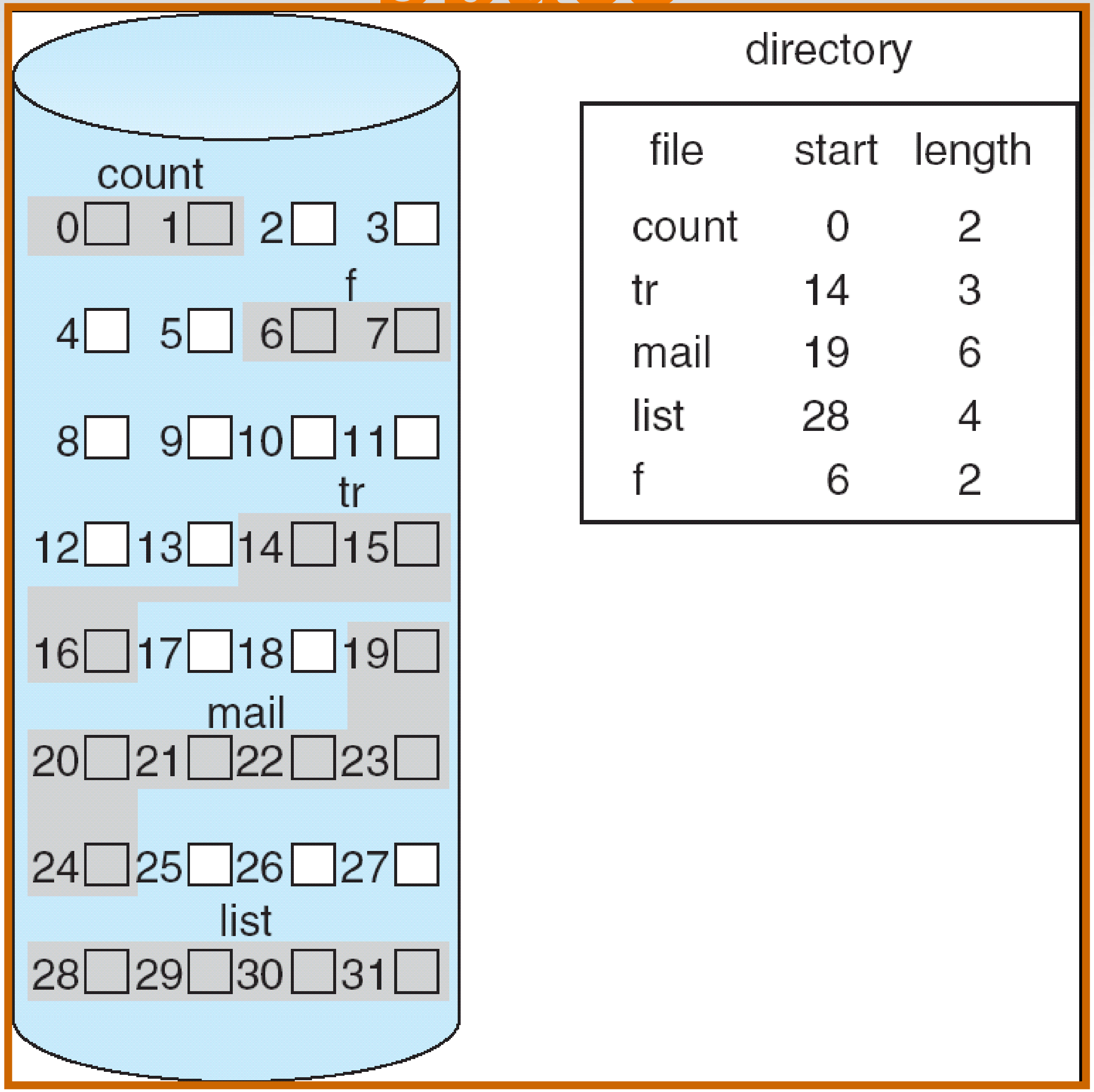
Allocation Methods

- An allocation method refers to how disk blocks are allocated for files:
- **Contiguous allocation**
- **Linked allocation**
- **Indexed allocation**

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

Contiguous Allocation of Disk Space

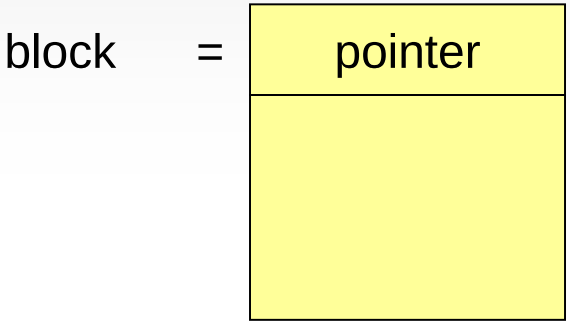


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Linked Allocation

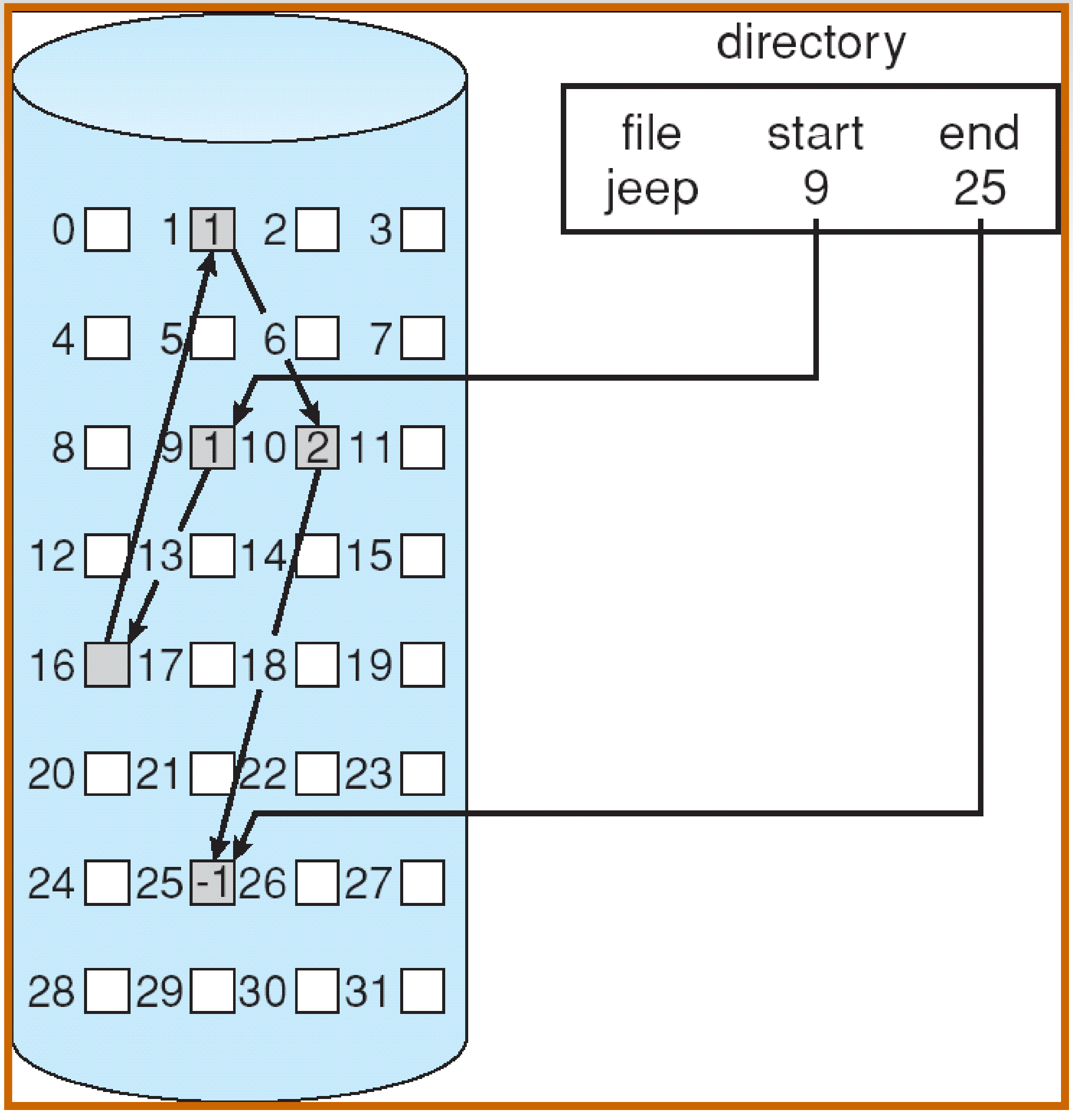
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.



Linked Allocation (Cont.)

- Simple – need only starting address
- Free-space management system – no waste of space
- No random access
- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.

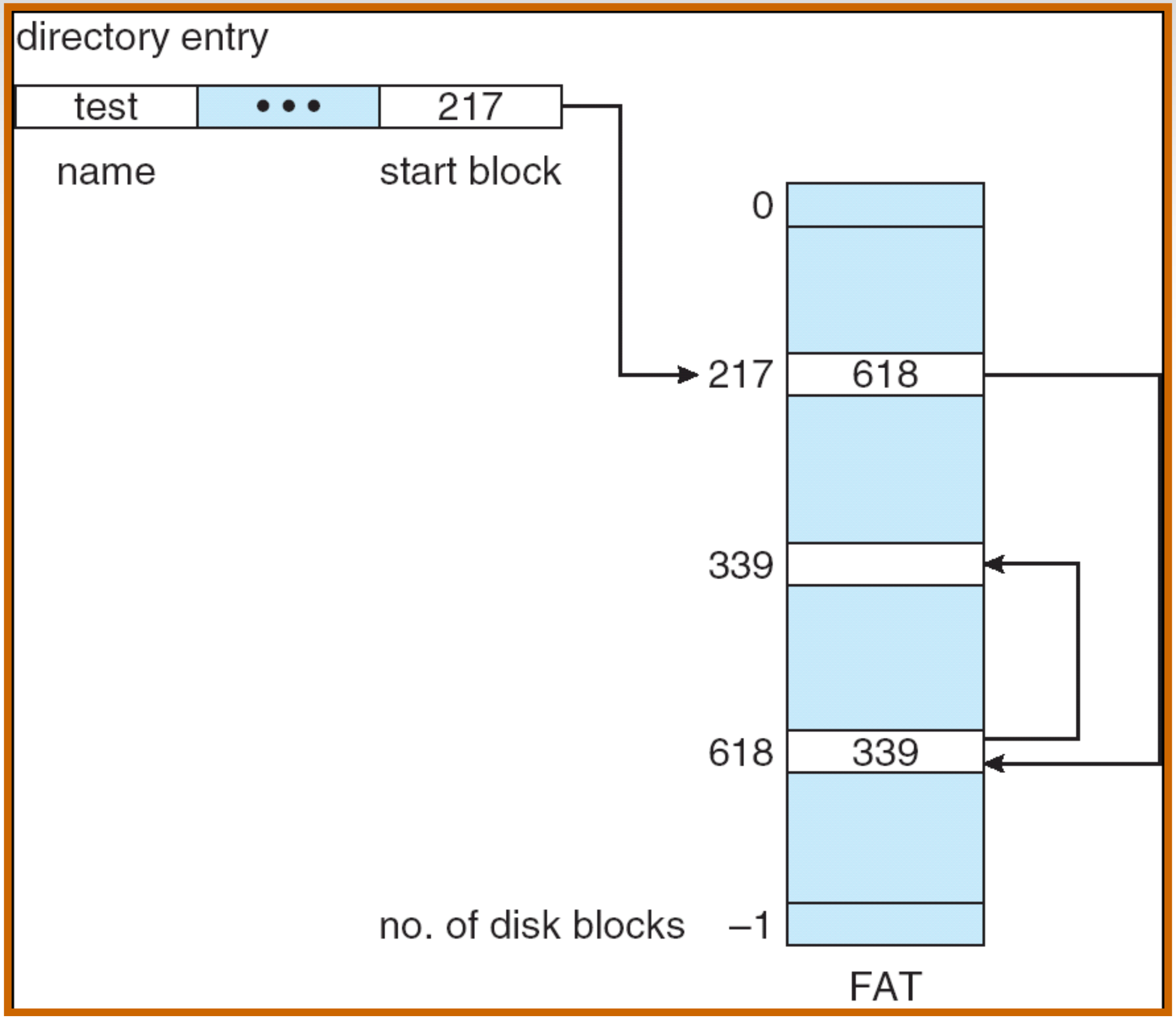
Linked Allocation



directory

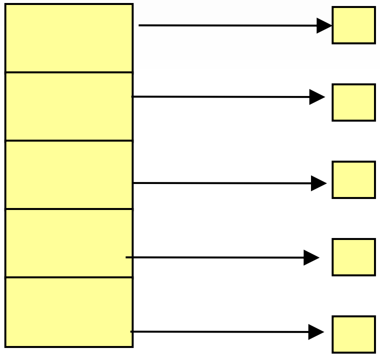
file	start	end
jeep	9	25

File-Allocation Table



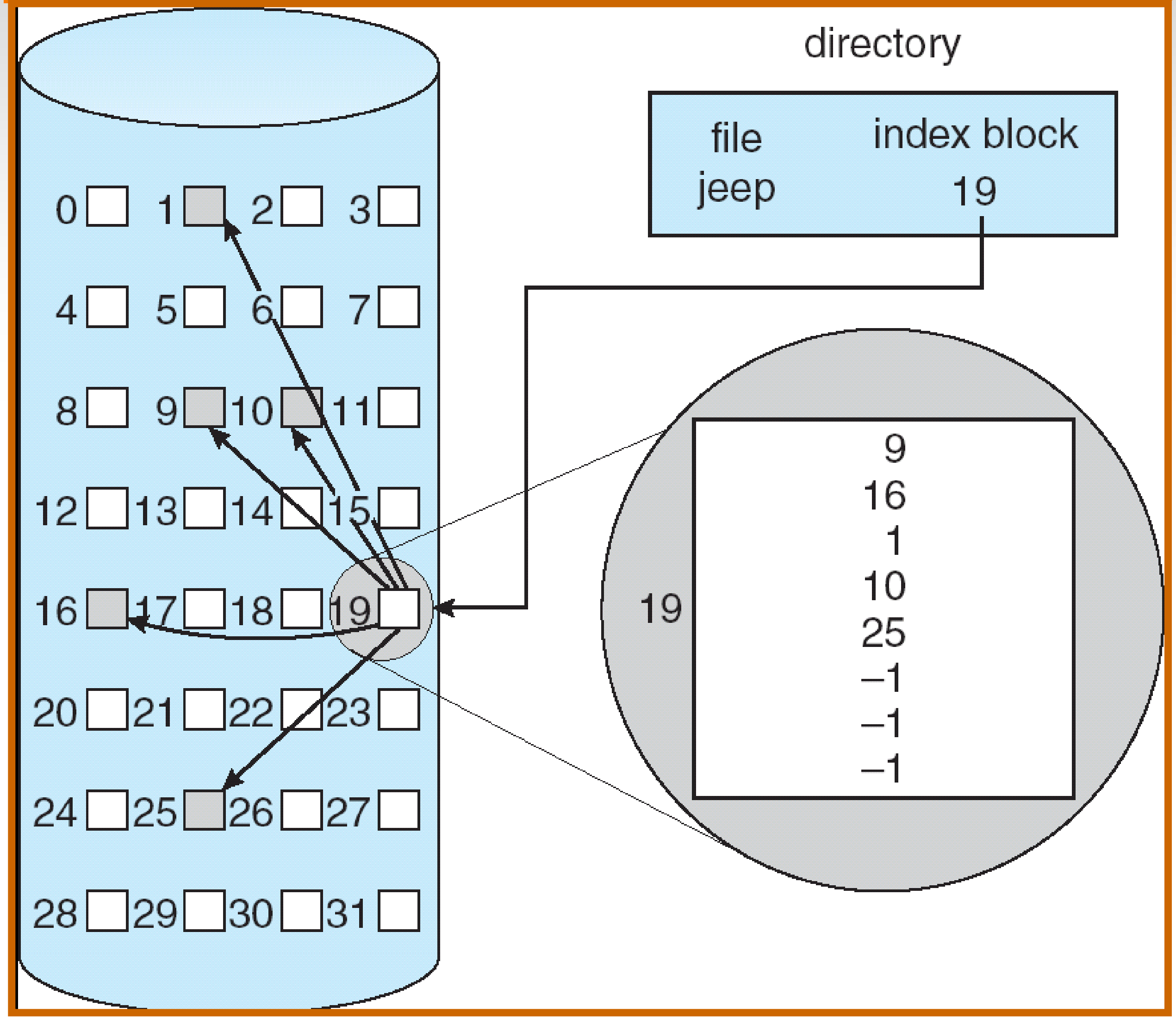
Indexed Allocation

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



index table

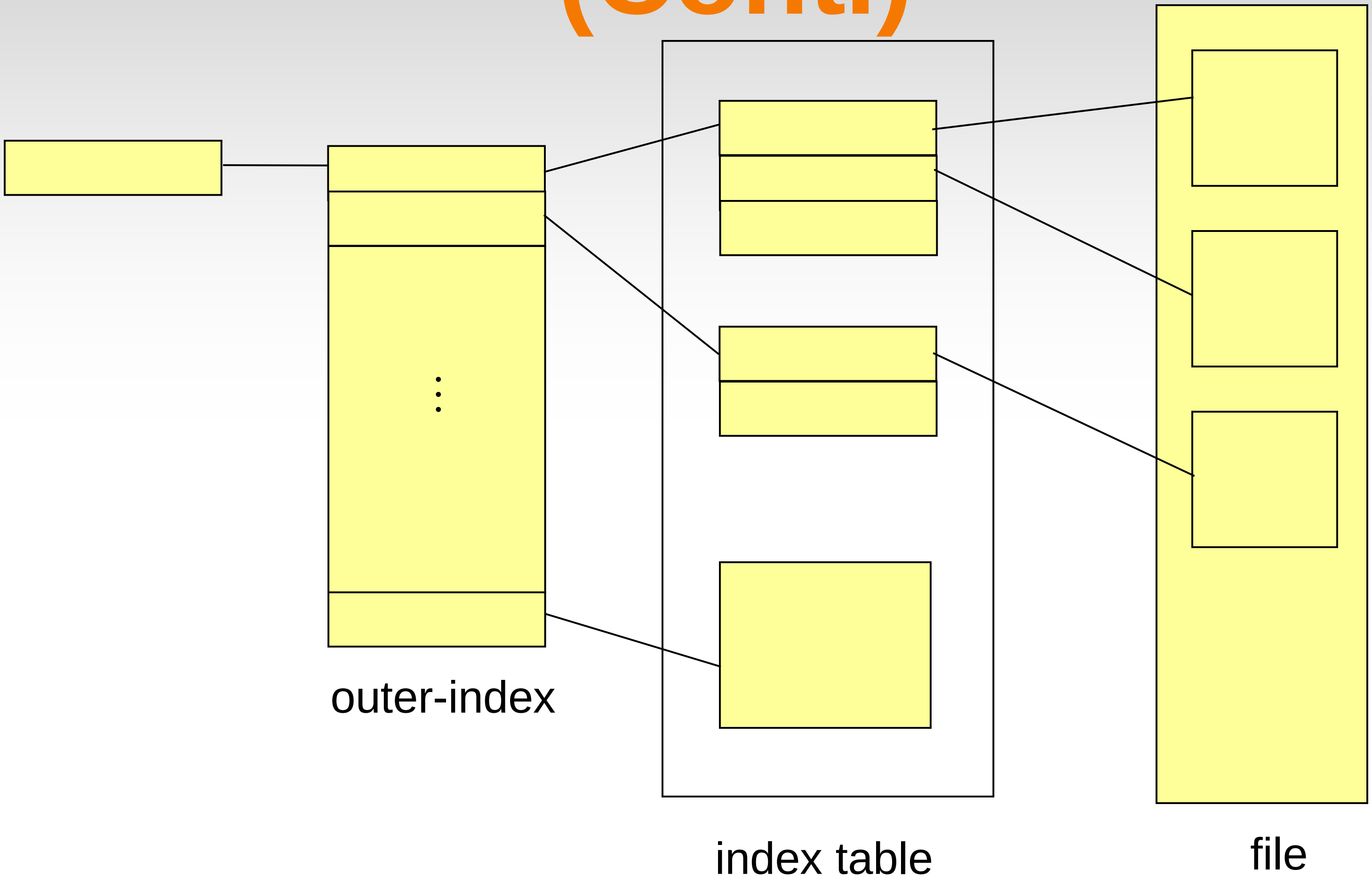
Example of Indexed Allocation



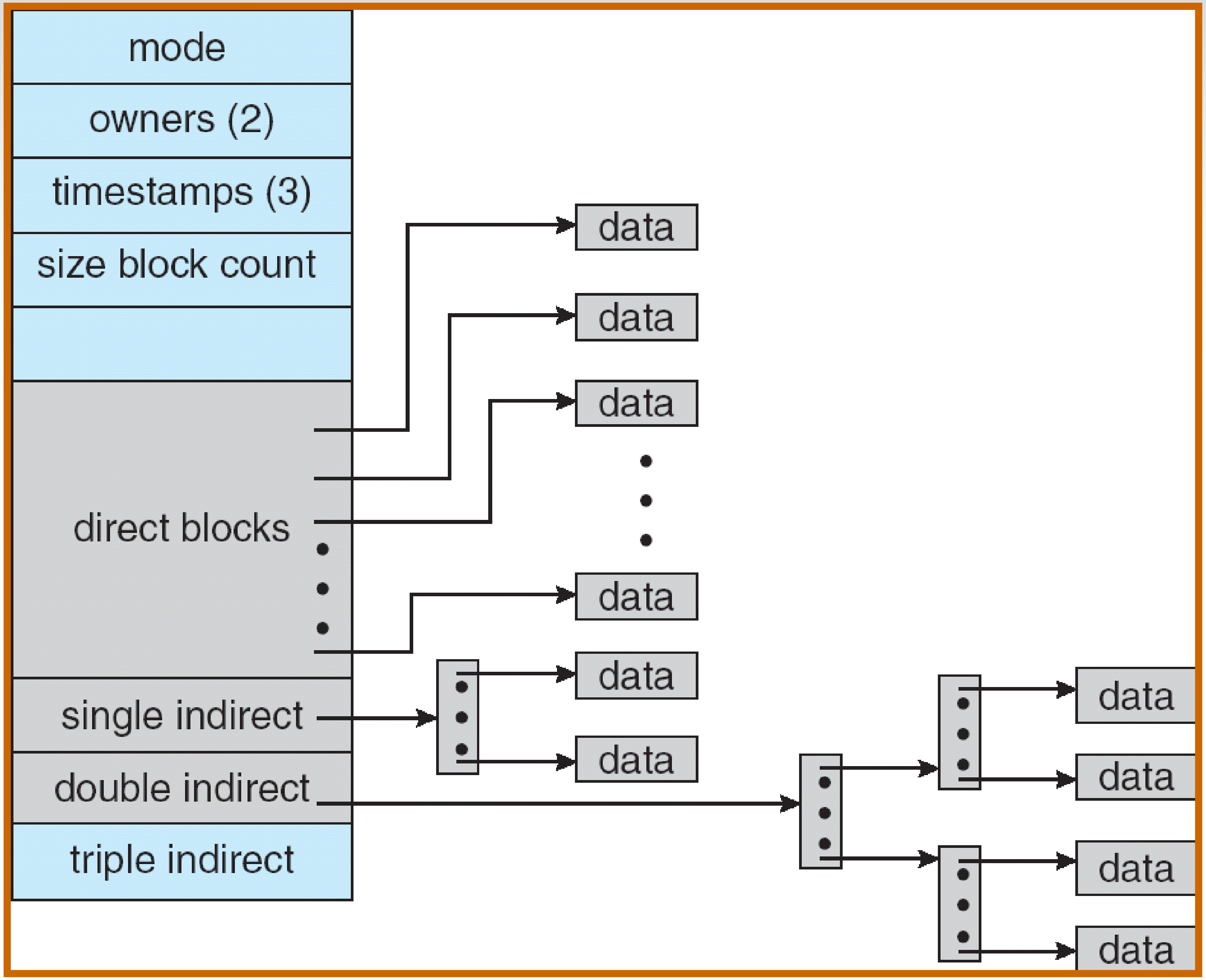
Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation, but have overhead of index block.

Indexed Allocation – Mapping (Cont.)



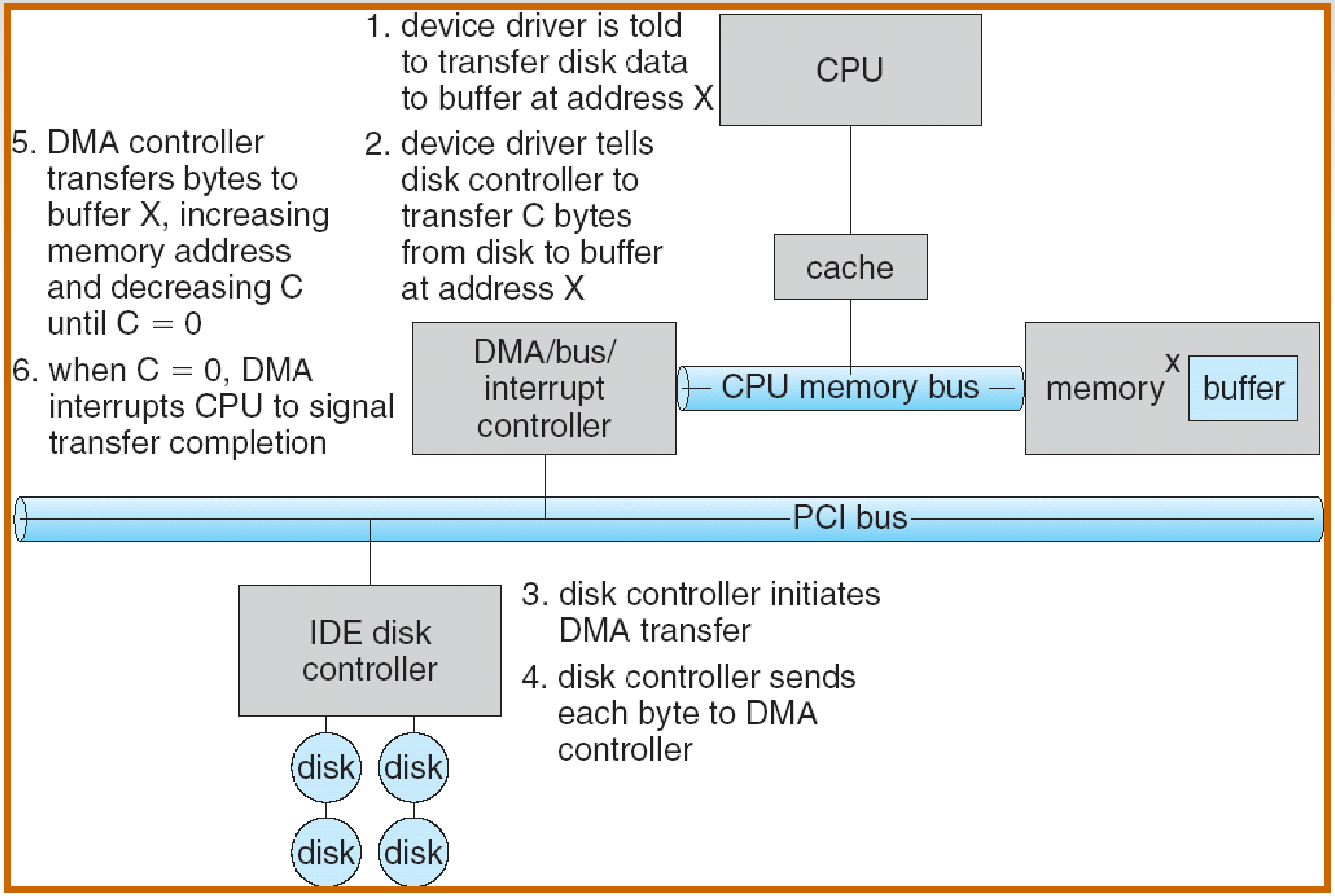
Combined Scheme: UNIX (4K bytes per block)



Direct Memory Access

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

Six Step Process to Perform DMA Transfer



1. device driver is told to transfer disk data to buffer at address X
2. device driver tells disk controller to transfer C bytes from disk to buffer at address X
3. disk controller initiates DMA transfer
4. disk controller sends each byte to DMA controller
5. DMA controller transfers bytes to buffer X, increasing memory address and decreasing C until C = 0
6. when C = 0, DMA interrupts CPU to signal transfer completion

Thank You