Chapter - Six

File System

Outline

- Introduction
- File system concepts and definition
- File attributes
- File access methods
- File operation, organization
- Directories
- File allocation
- File permission

Introduction

- Secondary storage is the non-volatile repository for (both user and system) data and programs.

- As (integral or separate) part of an operating system, the file system manages this information on secondary storage.

- Uses of secondary storage include:

  - storing various forms of programs (source, object, and executable) and

  - temporary storage of virtual memory pages (paging device or swap space).

Introduction...

- Generally there are three essential requirements for long term information storage:

    1. It must be possible to store a very large amount of information.
    2. The information must survive the termination of the process using it.
    3. Multiple processes must be able to access the information concurrently.

- The solution to all these problems is to store information on disks and other external media in units called Files.

File systems

- A file system provides a mapping between the logical and physical views of a file, through a set of services and an interface.

- Simply put, the file system hides all the device-specific aspects of file manipulation from users.

- The basic services of a file system include:
  - keeping track of files (knowing location),
  - I/O support, especially the transmission mechanism to and from main memory,
  - management of secondary storage,
  - sharing of I/O devices,
  - providing protection mechanisms for information held on the system.

File concept

- Files are managed by operating system.

- How they are structured, named, accessed, used, protected, and implemented are major topics in operating system.

- As a whole that part of the operating system dealing with files known as file system.

- A file is a named collection of related information, usually as a sequence of bytes, with two views:

  – Logical (programmer's) view, as the users see it.

  – Physical (operating system) view, as it actually resides on secondary storage.

File concept...

- What is the difference between a file and a data structure in memory? Basically,

    - files are intended to be non-volatile; hence in principle, they are long lasting,

    - files are intended to be moved around (i.e., copied from one place to another),

    - accessed by different programs and users, and so on.

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 attributes

- Each file is associated with a collection of information, known as attributes:

  - NAME, owner, creator

  - type (e.g., source, data, binary)

  - location (e.g., I-node or disk address)

  - organization (e.g., sequential, indexed, random)

  - access permissions

  - time and date (creation, modification, and last accessed)

  - Size

  - variety of other (e.g., maintenance) information.

File access methods

➢ The information stored in a file can be accessed in a variety of methods:

  – Sequential: in order, one record after another.

  – Direct (random): in any order, skipping the previous records.

  – Keyed: in any order, but with particular value(s); e.g., hash table or dictionary.

  – TLB lookup is one example of a keyed search.

File Operations

- Create

- Write

- Read

- file seek – 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

Addressing levels

- There are three basic mapping levels (abstractions) from a logical to physical view of a file (contents):
- File relative:
  - <filename, offset> form is used at the higher levels, where the file system is viewed as a collection of files.
- Volume (partition) relative:
  - device-independent part of a file system use
  - <sector, offset> (e.g., a partition is viewed as an array of sectors.)
- Drive relative:
  - at the lowest level, <cylinder, head, sector> (also known as <track, platter, sector>) is used.

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	read 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, rrf, doc	various word-processor formats
library	lib, a, so, dll, mpeg, mov, rm	libraries of routines for programmers
print or view	arc, zip, tar	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	binary file containing audio or A/V information

File Organization

- One of the key elements of a file system is the way the files are organized.

- File organization is the "logical structuring" as well as the access method(s) of files.

- Common file organization schemes are:
  - Sequential
  - Indexed-sequential
  - Indexed
  - Direct (or hashed)
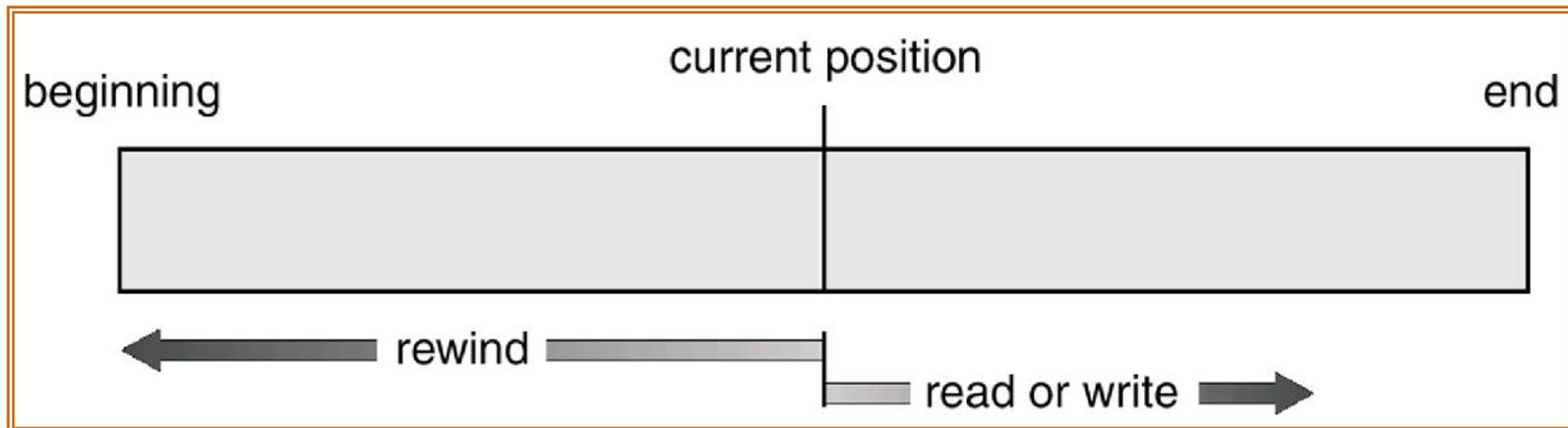
Access Methods

- 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 a Direct-access File

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

Example of Index and Relative Files
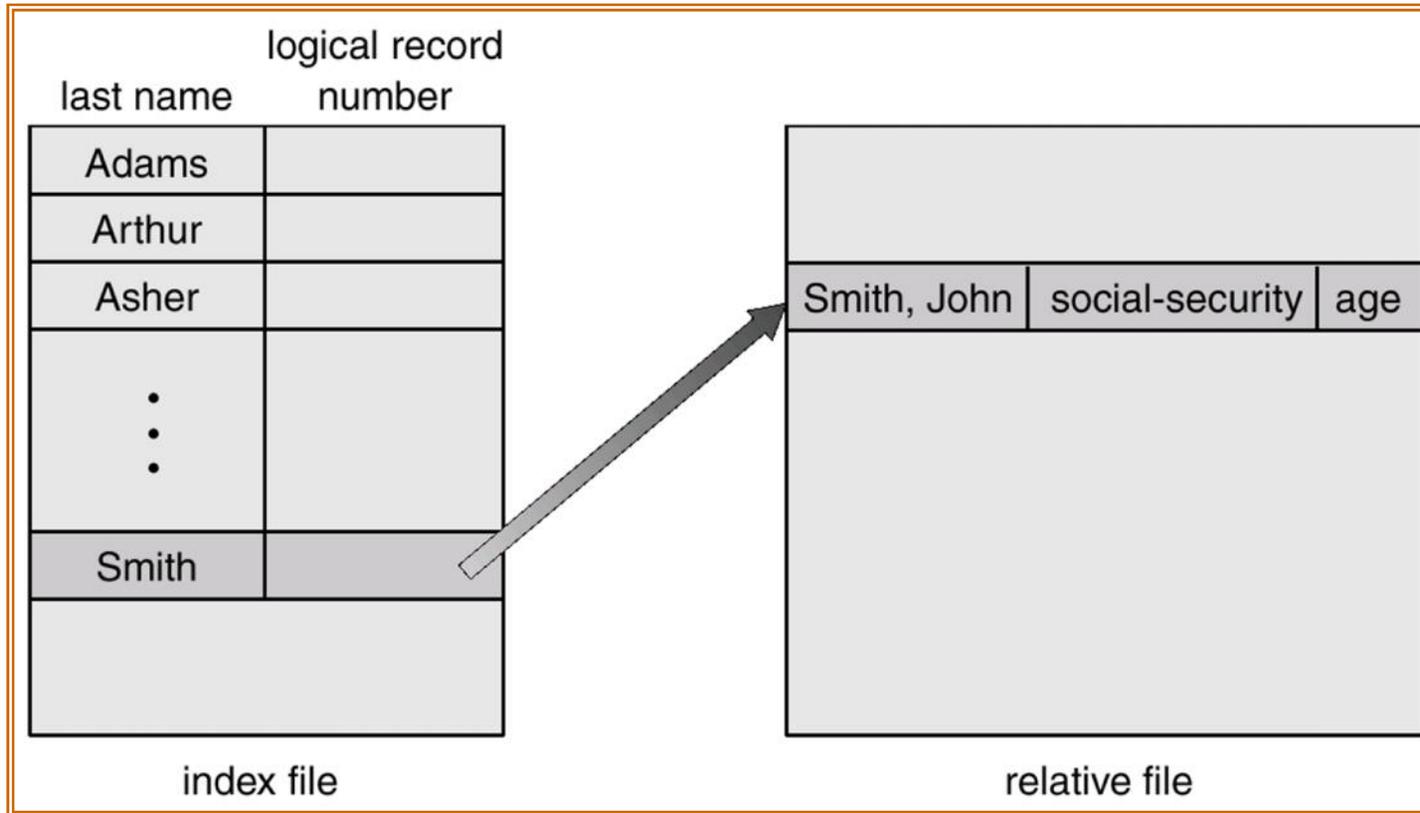
Directories

- A directory is a symbol table, which can be searched for information about the files.

- Also, it is the fundamental way of organizing files.

- Usually, a directory is itself a file.

- A typical directory entry contains information (attributes) about a file. Directory entries are added as files are created, and are removed when files are deleted.

- Common directory structures are:
  - Single-level (flat): shared by all users.
  - Two-level: one level for each user.
  - Tree: arbitrary (sub)-tree for each user.

Directory Structure

- A collection of nodes containing information about all files

Directory

Files

F 1 F 2 F 3 F 4 F n
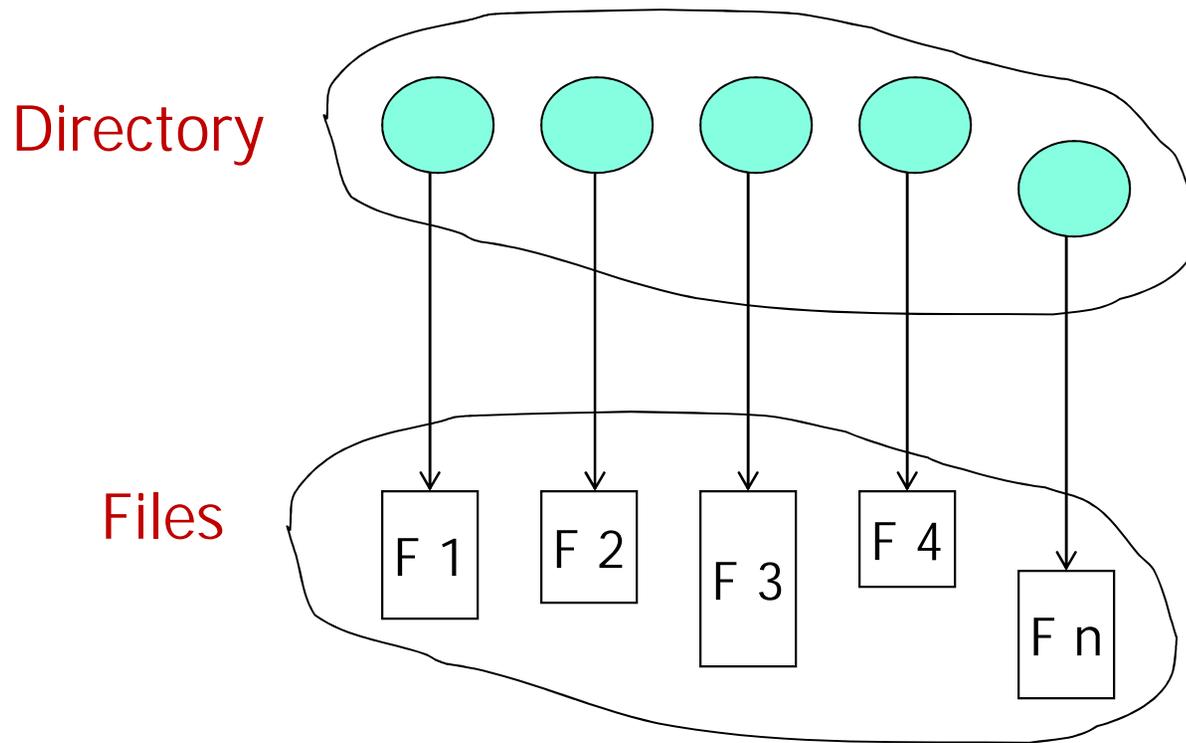
Both the directory structure and the files reside on disk

A Typical File-system Organization
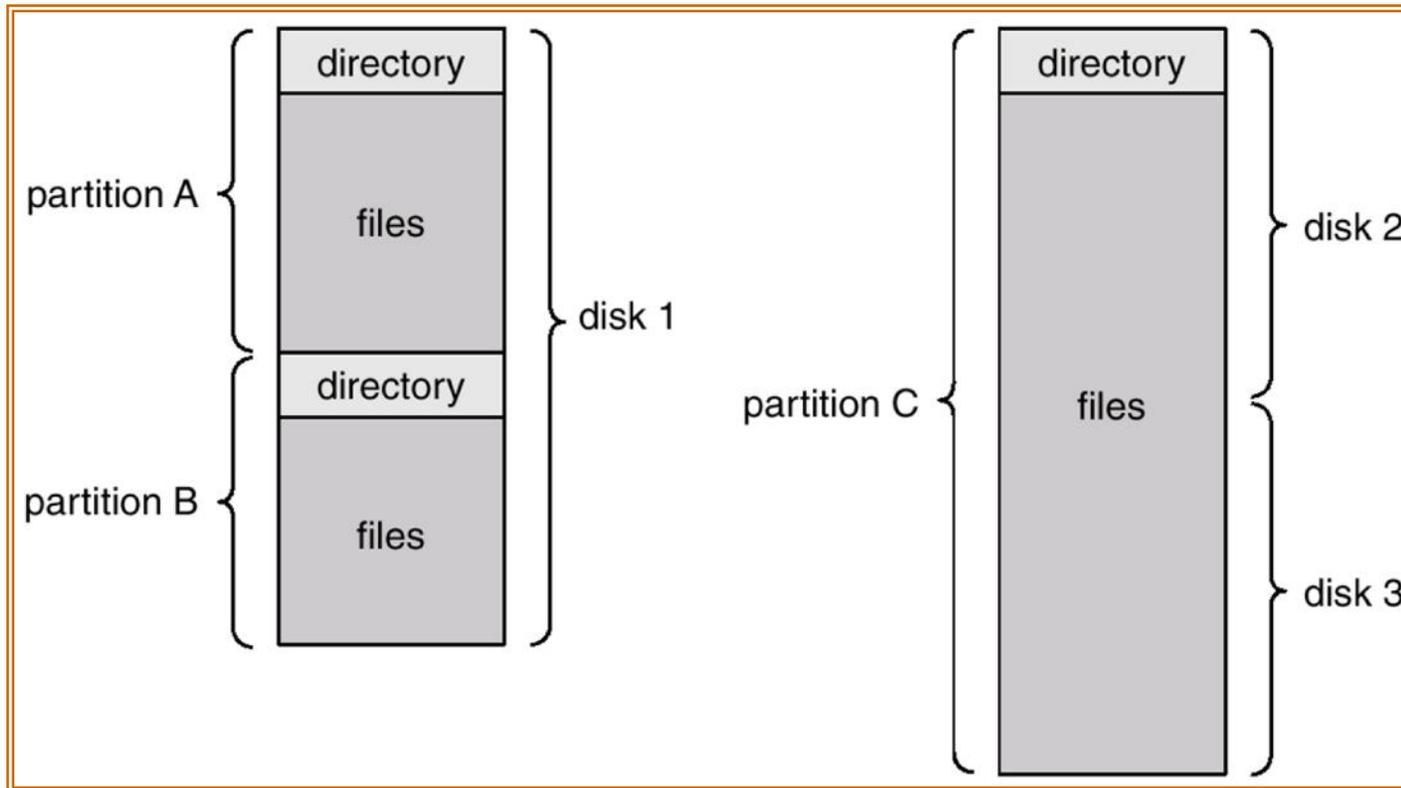
Information in a Device Directory

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

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
- Shared by all users.
- One directory containing all the files (root directory).

directory	cat	bo	a	test	data	mail	cont	hex	records
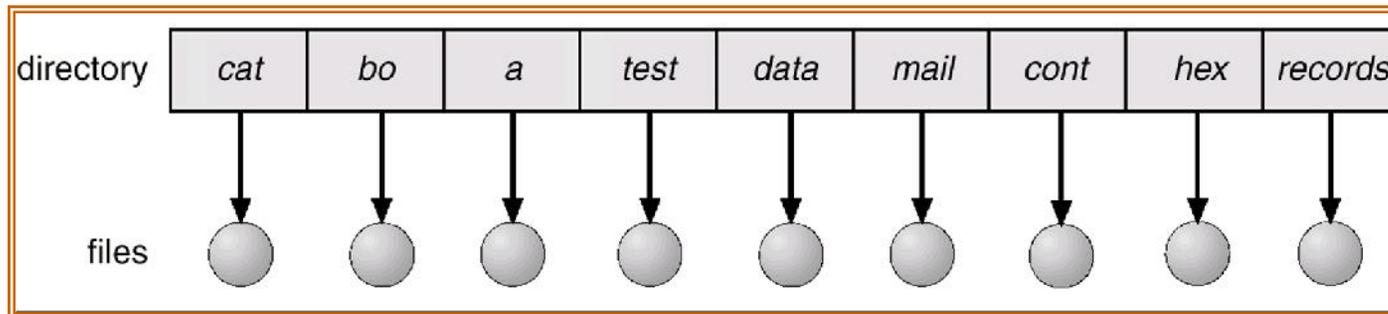
files

Naming problem

Grouping problem

Example:

Root directories

owners
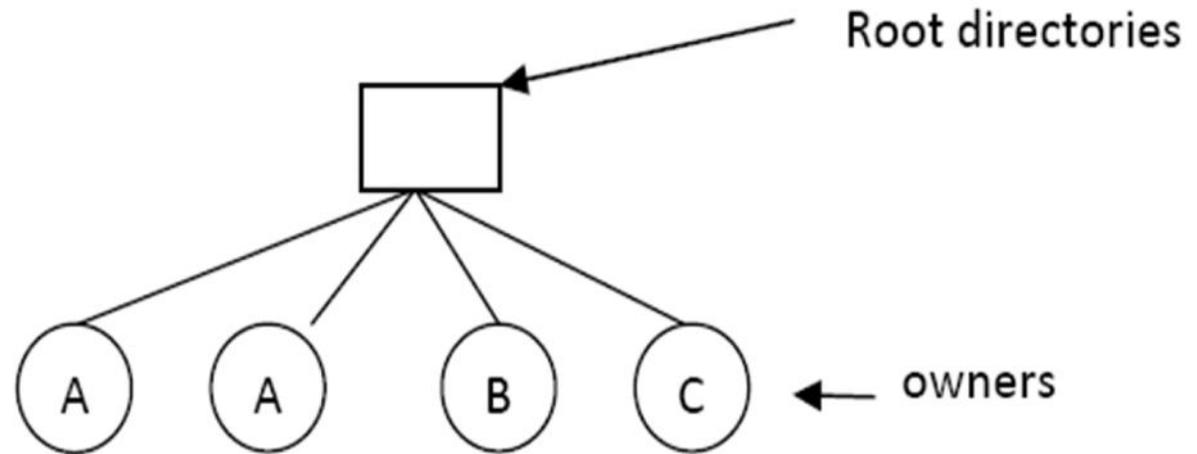
➤ Here the directory contains four files.

➤ Disadvantage:

  – Different users may accidentally use the same names for their files.

  – Example: User A creates file name mailbox. Latter user B creates file name mailbox. Then B's file will overwrite A's file.

Two-level directory system

- To overcome the above problem, just give each user a private directory.
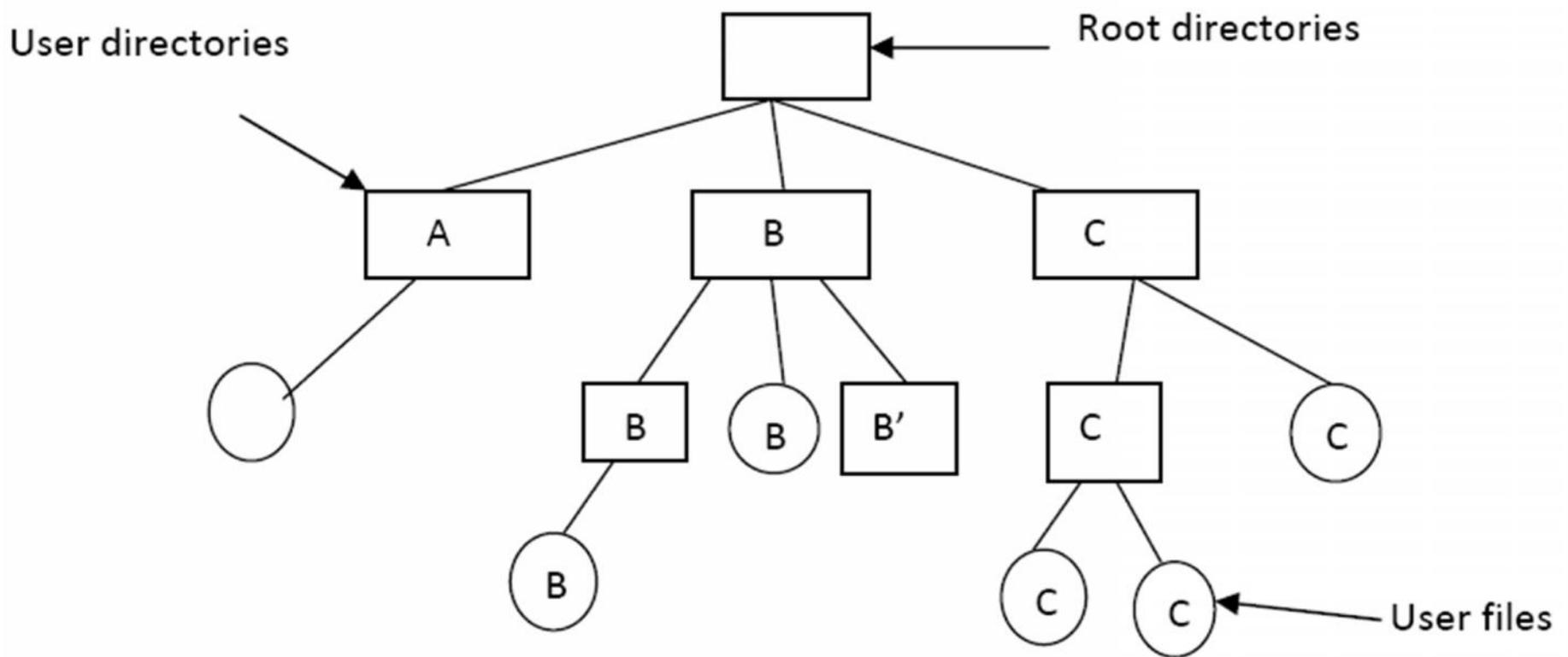- One level for each user.
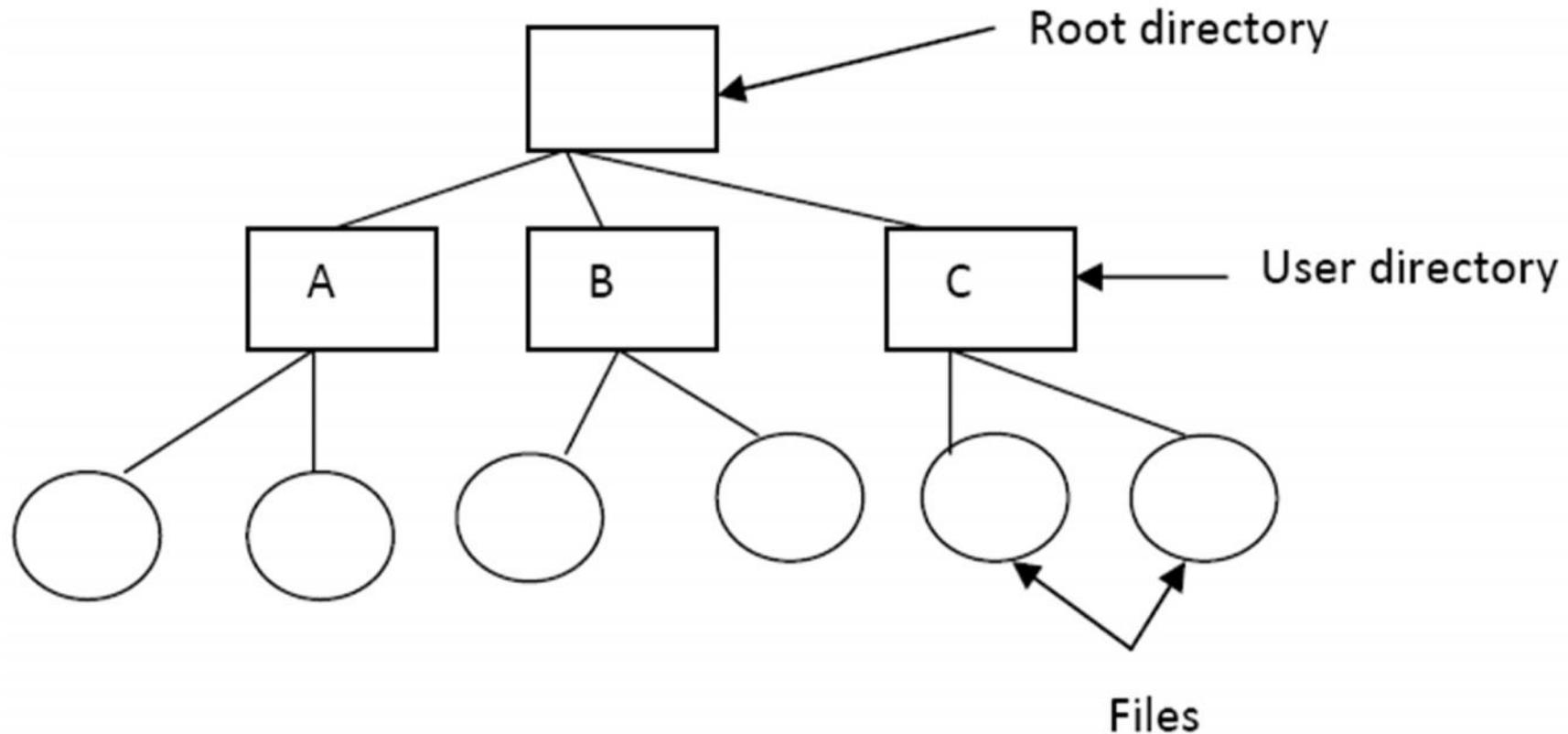
Two-Level Directory...

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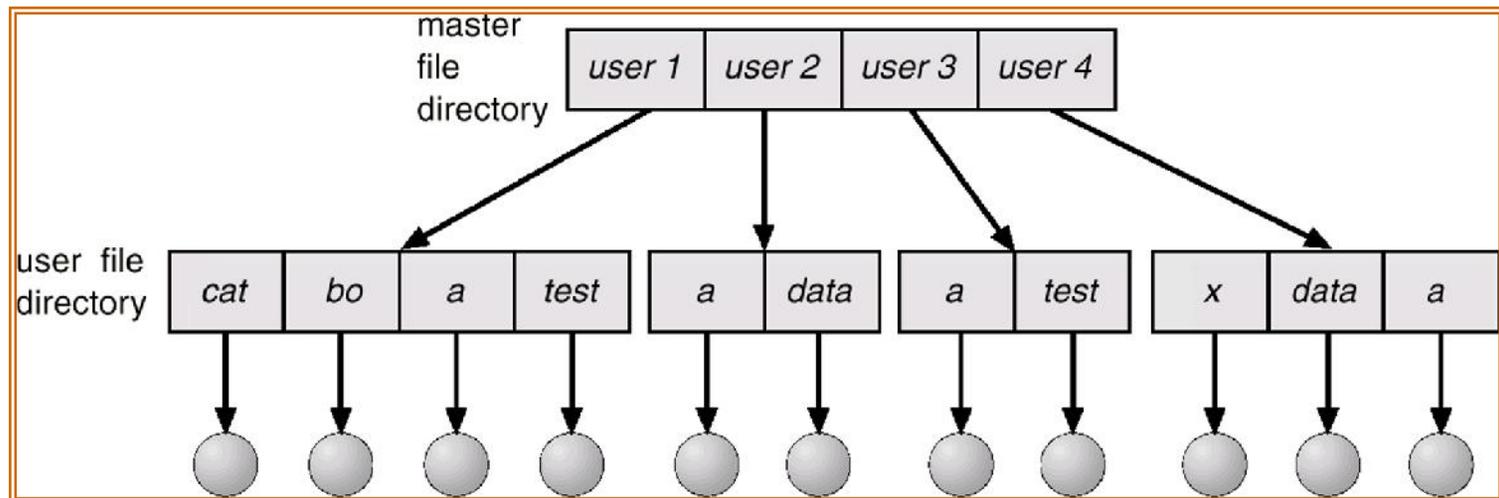
- Path name
- Can have the same file name for different user
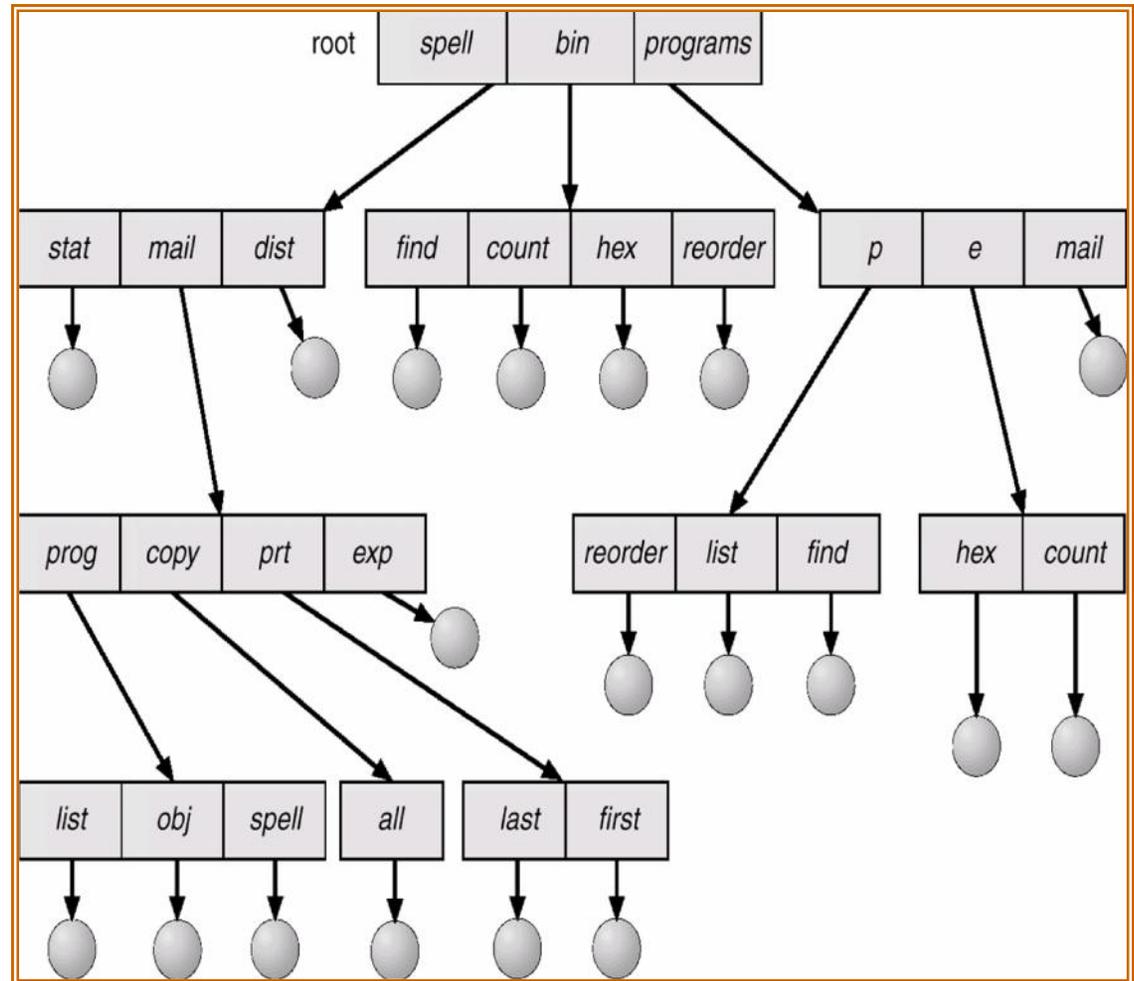- Efficient searching
- No grouping capability

Tree (Hierarchical directory system)

- Arbitrary (sub)-tree for each user.
- It is used for large number of files.
- Each user can have as many directories as possible.

Tree-Structured Directories...

- **Efficient searching**

- **Grouping Capability**

- **Current directory (working directory)**
  - cd /spell/mail/prog
  - type list

Tree-Structured Directories (Cont)

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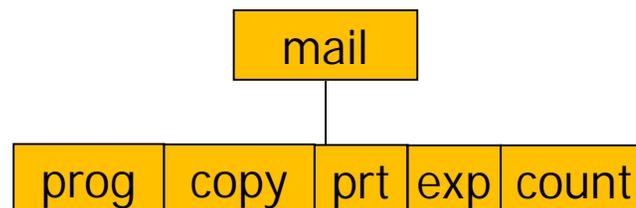- Creating a new file is done in current directory
- Delete a file

<p style="text-align:center;color:red;">rm <file-name></p>

- Creating a new subdirectory is done in current directory

<p style="text-align:center;color:red;">mkdir <dir-name></p>

Example: if in current directory   /mail

<p style="text-align:center;color:red;">mkdir count</p>

```
          mail
           |
 prog  copy  prt  exp  count
```

Deleting "mail" $\Rightarrow$ deleting the entire subtree rooted by "mail"

Acyclic-Graph Directories

- Have shared subdirectories and files

- Two different names (aliasing)

- If dict deletes list $\Rightarrow$ 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
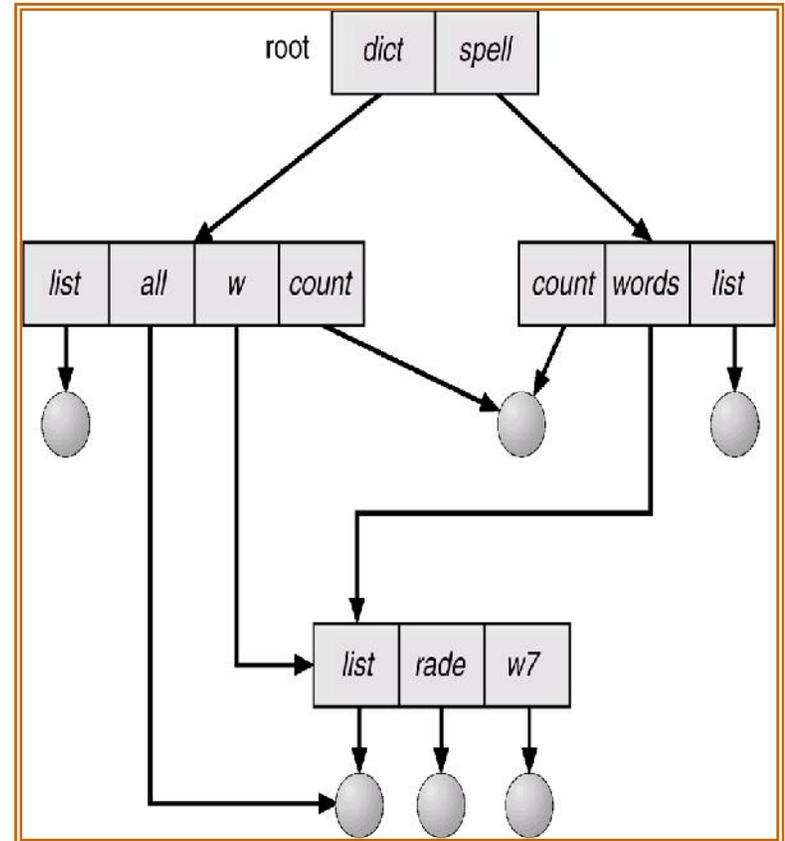
General Graph Directory

- 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
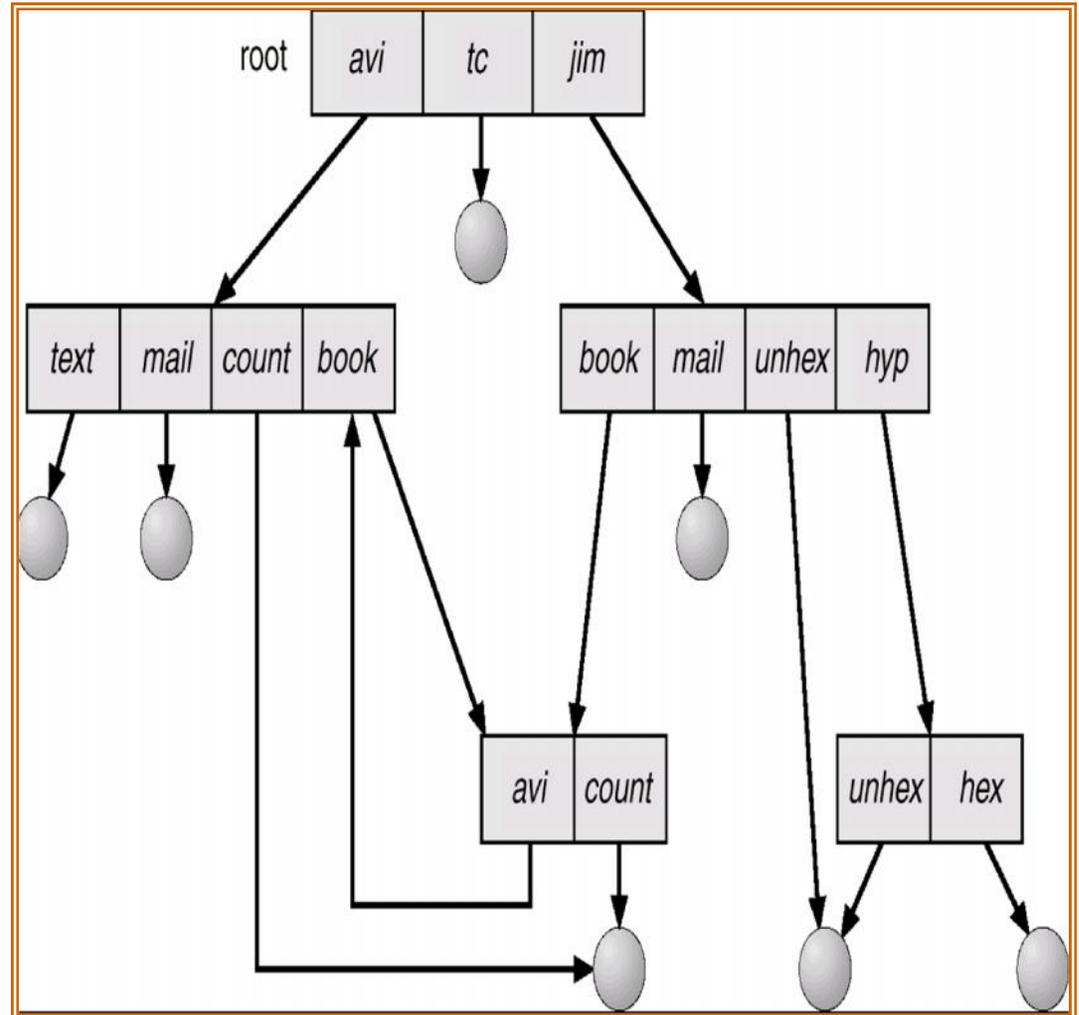
Directory operation

- Create- a directory is created.

- Delete- a directory is deleted.

- Opendir- directories can be read.

- Closedir- directory can be closed.

- Readdir- this call returns the next entry in an open directory.

File allocation

- The file system allocates disk space, when a file is created.

- With many files residing on the same disk, the main problem is how to allocate space for them.

- File allocation scheme has impact on the efficient use of disk space and file access time.

- Common file allocation techniques are:
  - Contiguous
  - Chained (linked)
  - Indexed

- All these techniques allocate disk space on a per block (smallest addressable disk units) basis.
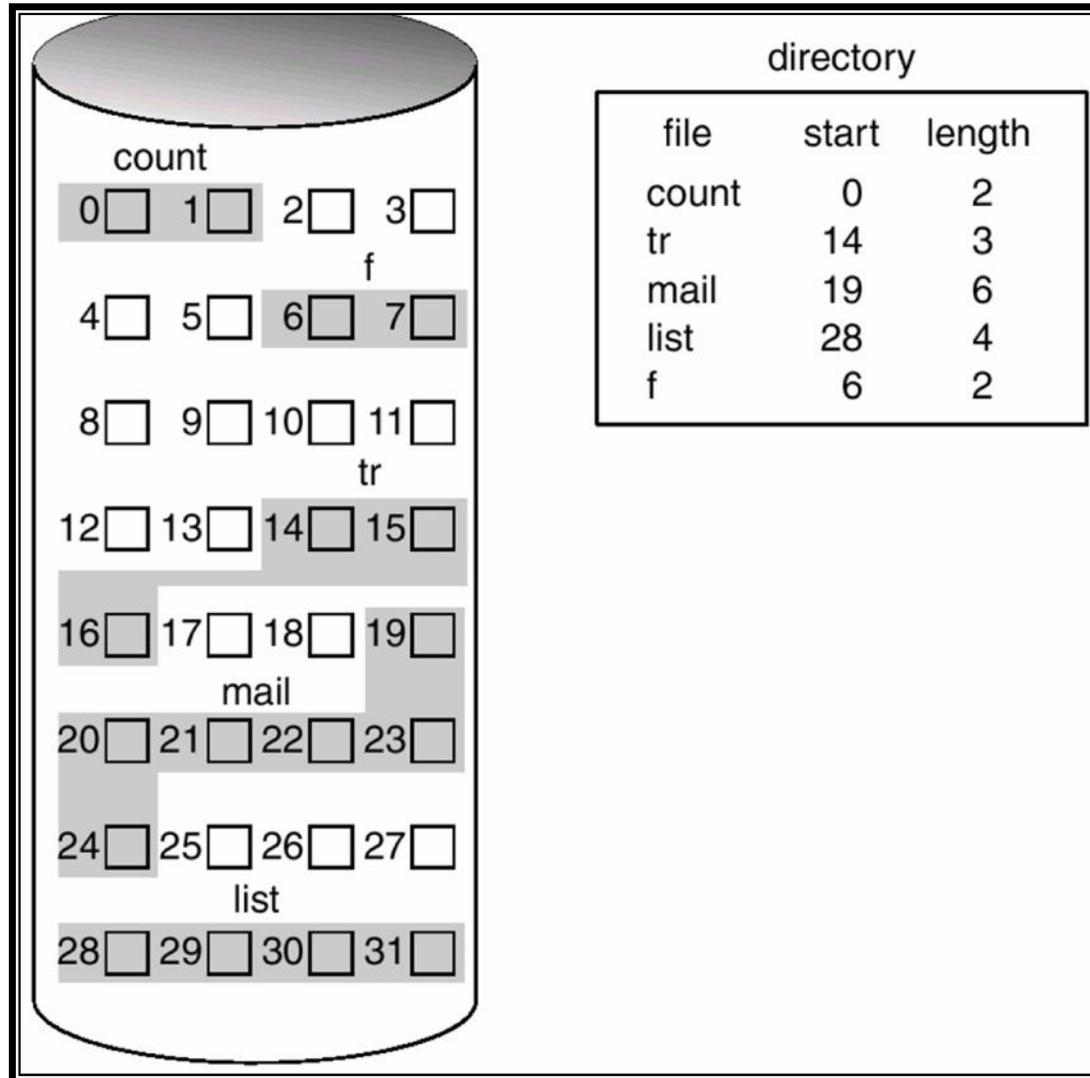
Contiguous allocation

- Allocate disk space like paged, segmented memory.

- Keep a free list of unused disk space.

- Advantages

  – Easy access, both sequential and random

  – Simple

  – Few seeks

- Disadvantages

  – External fragmentation

 – May not known the file size in advance
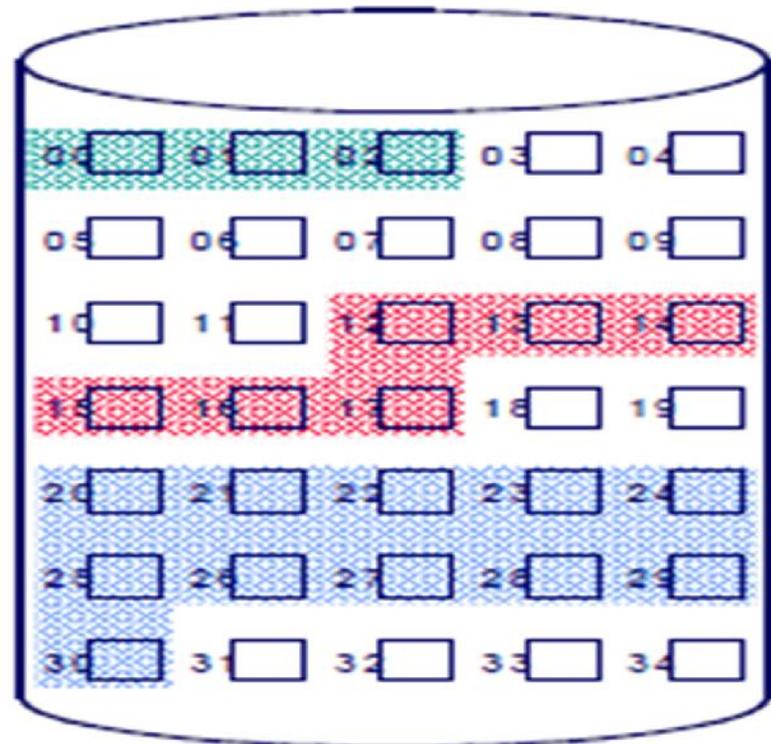
Directory

| name	start	len.
a.out	00	3
hw1.c	12	6
report.tex	20	11

Contiguous Allocation of Disk Space

Chained (linked) allocation

- Space allocation is similar to page frame allocation.
- Mark allocated blocks as in-use.
- Advantages:
  - no external fragmentation
  - files can grow easily
- Disadvantages
  - Lots of seeking
  - Random access difficult
- Example
  - MSDOS (FAT) file system
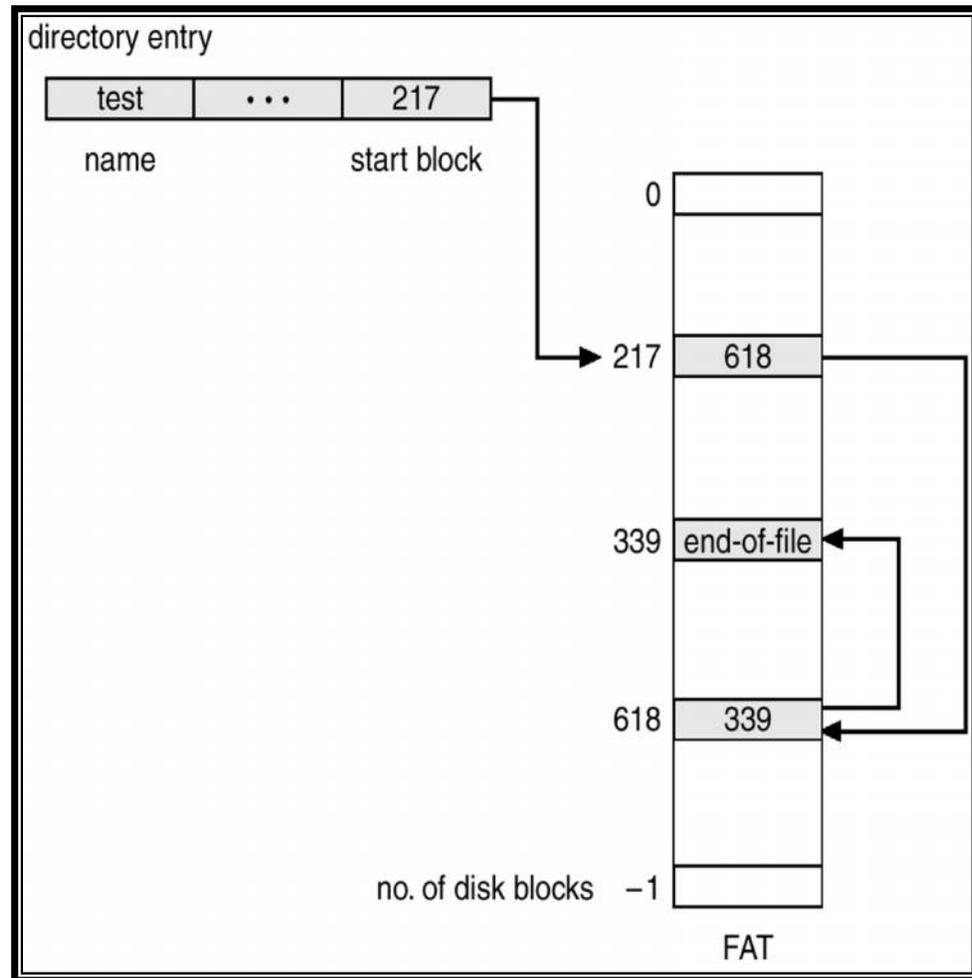

name	start	len.
a.out	01	3
hw1.c		
report.tex		

Directory
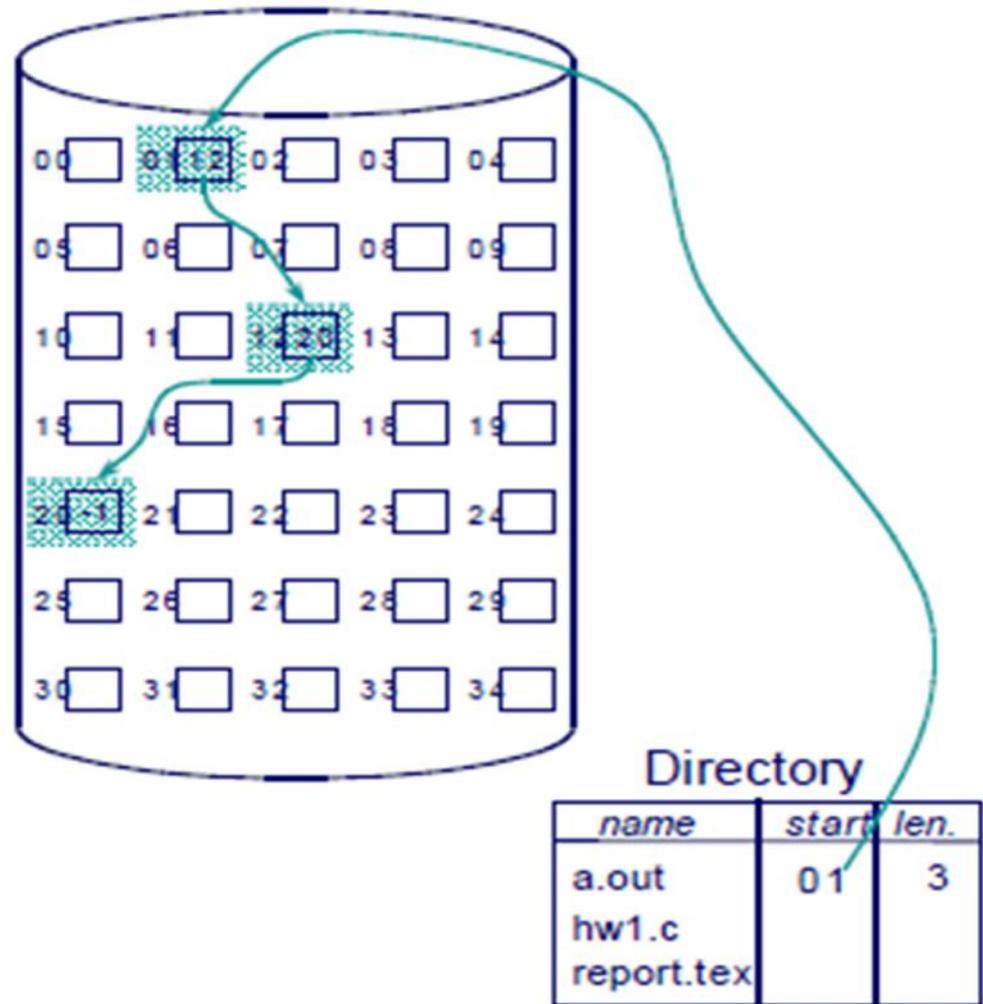
Linked Allocation…

- Each file is a linked list of disk blocks:

- blocks may be scattered anywhere on the disk.
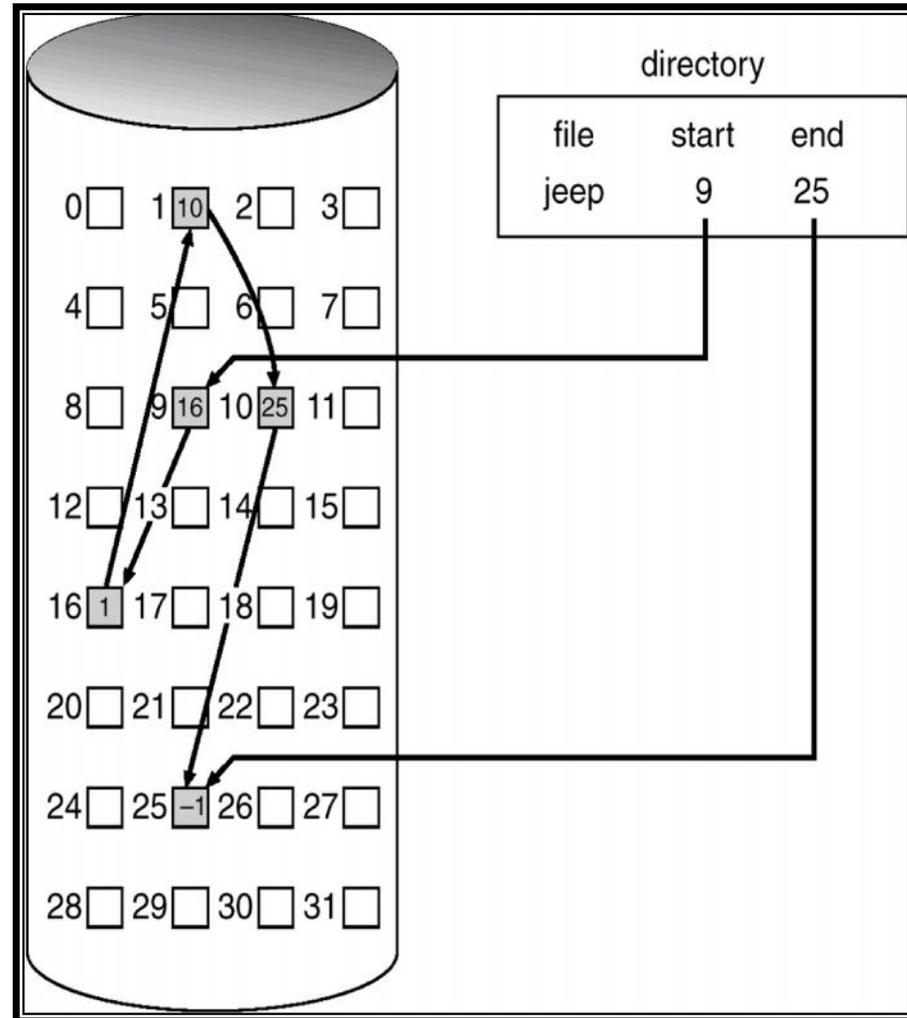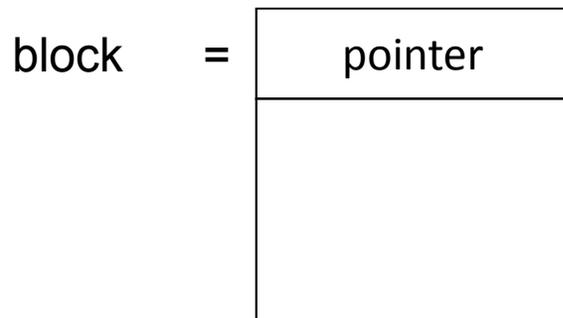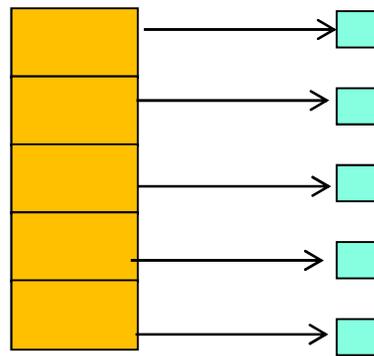
block = | pointer |

File-Allocation Table

- File-allocation table (FAT) – disk-space allocation used by MS-DOS and OS/2.
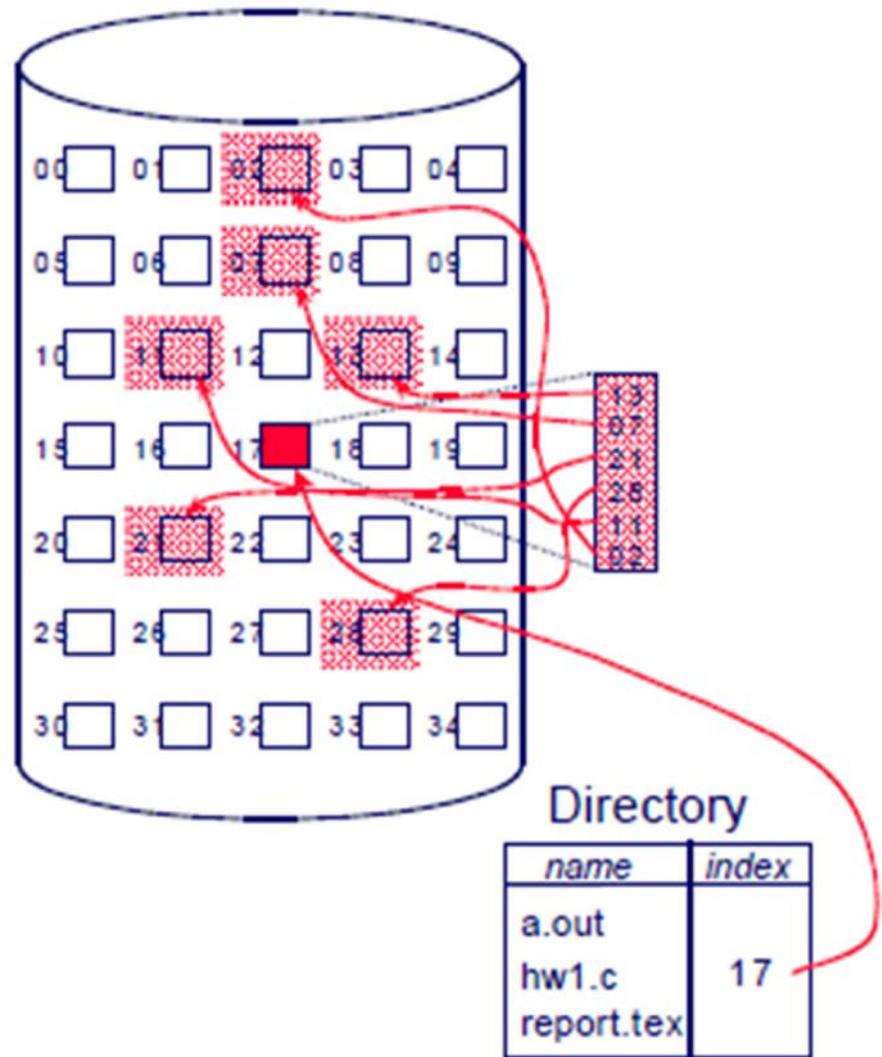
Indexed Allocation

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

index table

Indexed allocation...

- Allocate an array of pointers during file creation.

- Fill the array as new disk blocks are assigned.

- Advantages
  - Small internal fragmentation
  - Easy sequential and direct access

- Disadvantages
  - Lots of seeks if the file is big
  - Maximum file size is limited to the size of the block

- Example
  - Unix file system

Directory

| name	index
a.out	
hw1.c	17
report.tex	

Example of Indexed Allocation
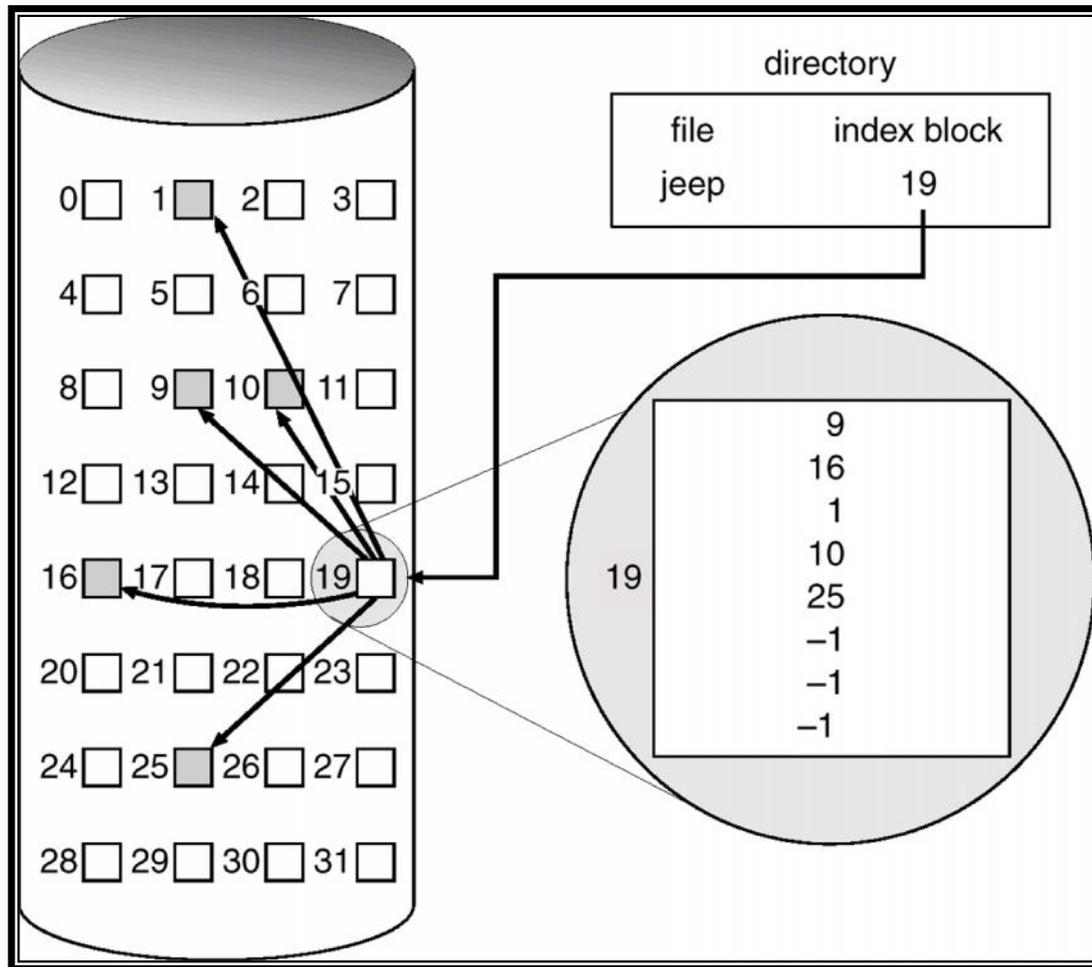
File Permissions

- Read, write, and execute privileges

- In Windows:

  – Change permission on the Security tab on a file's Properties dialog box

  – Allow indicates grant;

  – Deny indicates revoke

- In UNIX/Linux

  – Three permission settings:

    - owner; group to which owner belongs; all other users

  – Each setting consist of rwx

    - r for reading, w for writing, and x for executing

  – CHMOD command used to change file permissions

File Permissions
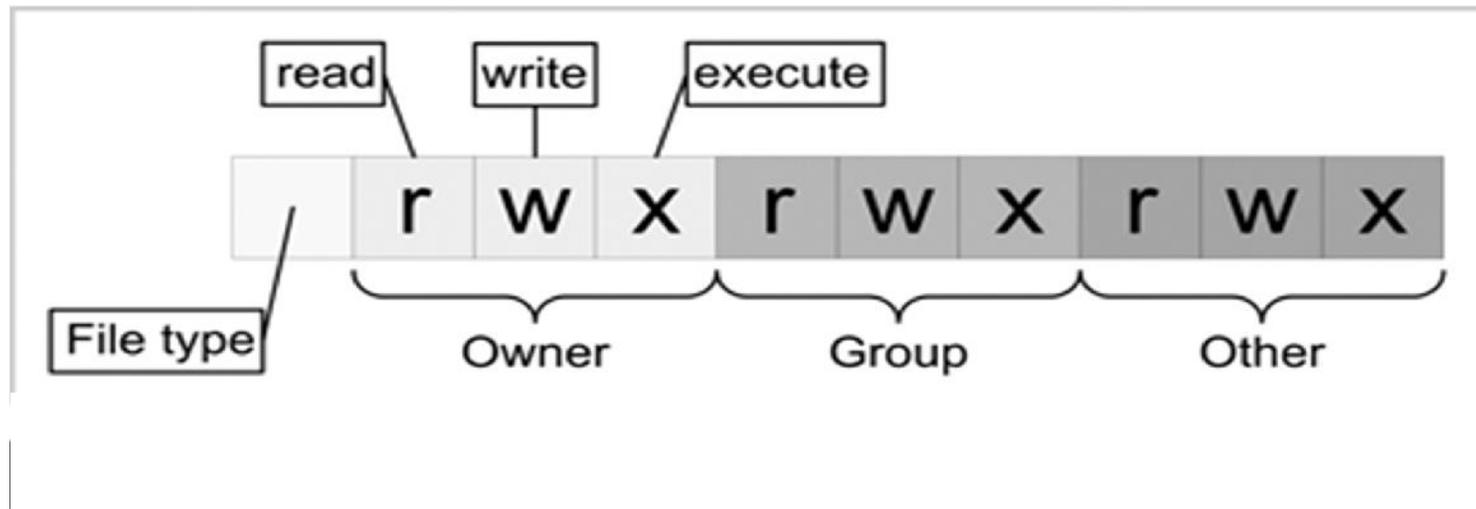
- One can easily view the permissions for a file by invoking a long format listing using the command ls -l.

- For instance, if the user Abe creates an executable file named test, the output of the command ls -l test would look like this:

  rwxrwxr-x Abe student Sep 26 12:25 test.l

Access Permissions

- This listing indicates that the file is readable, writable, and executable by the user who owns the file (user Abe)

- as well as the group owning the file (which is a group named student).

- The file is also readable and executable, but not writable by other users.

rwxrwxr-x Abe student Sep 26 12:25 test.l

Access Permission of File/Directory

- The ownership of the file or directory can be changed using the command

    – **chown <owner> <file/directory name>**

- The group of the file or directory can be changed using the command

    – **chgrp <group> <file/directory name>**

- The permissions of the file can be changed using chmod command

    – **chmod -R ### <filename or directory>**

- -R is optional and when used with directories will traverse all the sub-directories of the target directory changing ALL the permissions to ###.

Access Permission of File/Directory

The #'s can be:

0 = Nothing
1 = Execute
2 = Write
3 = Execute & Write (2 + 1)
4 = Read
5 = Execute & Read (4 + 1)
6 = Read & Write (4 + 2)
7 = Execute & Read & Write (4 + 2 + 1)