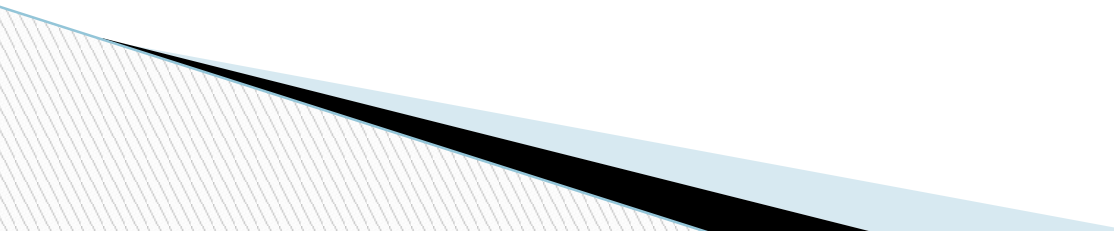


File Management Unit-5



Topics To Be Covered

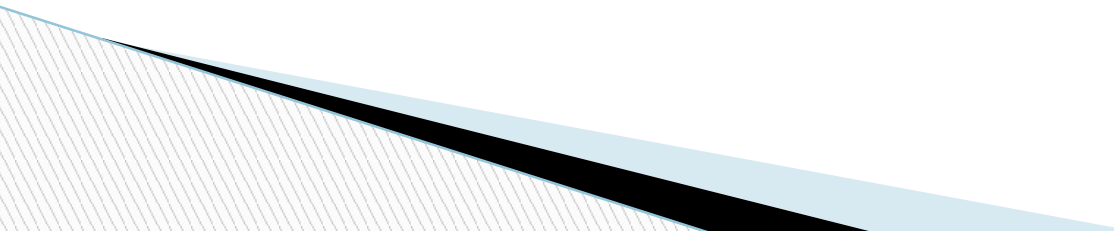
- File Concept
 - Access Methods
 - Disk and Directory Structure
 - File-System Mounting
 - File Sharing
 - Protection
- 

File Concept

- Contiguous logical address space
- Types:
 - Data
 - numeric
 - character
 - binary
 - Program
- Contents defined by file's creator
 - Many types
 - Consider **text file, source file, executable file**

File Structure

A File Structure should be according to a required format that the operating system can understand.

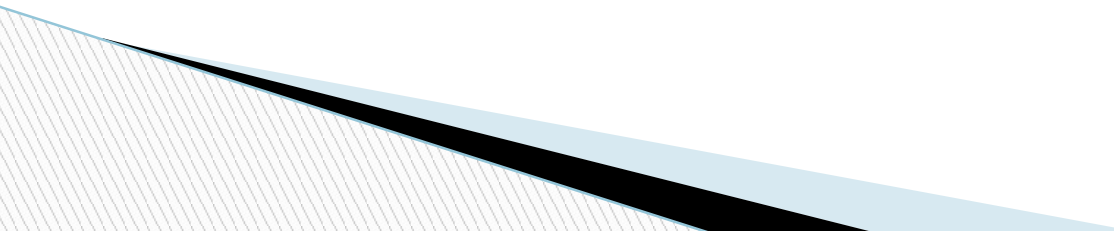
- A file has a certain defined structure according to its type.
 - A text file is a sequence of characters organized into lines.
 - A source file is a sequence of procedures and functions.
 - An object file is a sequence of bytes organized into blocks that are understandable by the machine.
 - When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure
- 

File Structure

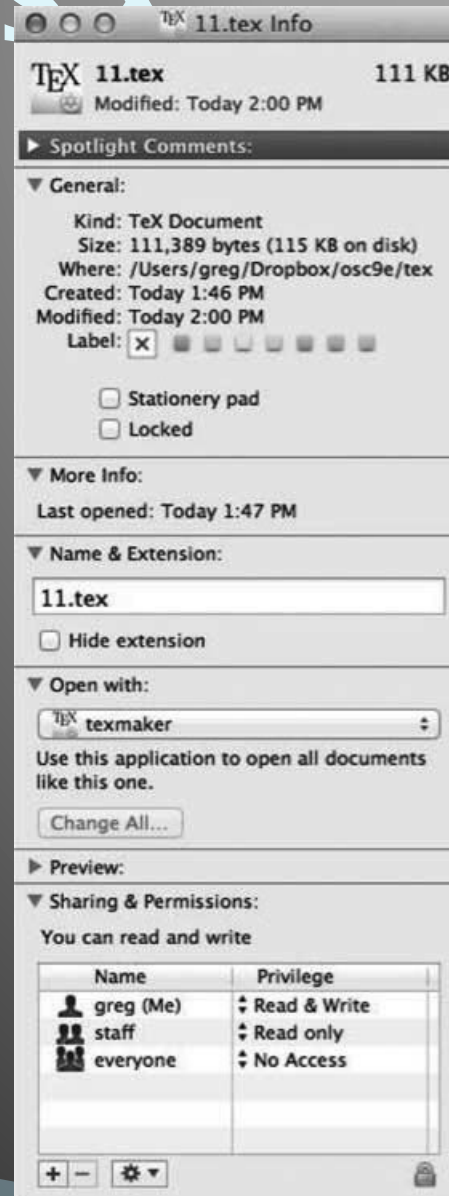
- None - sequence of words, bytes
- Simple record structure
 - Lines
 - Fixed length
 - Variable length
- Complex Structures
 - Formatted document
 - Relocatable load file
- Can simulate last two with first method by inserting appropriate control characters
- Who decides:
 - Operating system
 - Program

File Attribute

S

- **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
 - Many variations, including extended file attributes such as file checksum
 - Information kept in the directory structure
- 

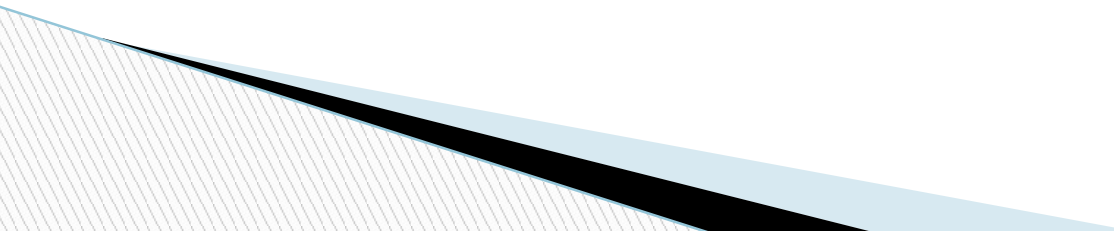
File info Window on Mac OS X



File Operation

- File is an **abstract data type**
- **Create**
- **Write** – at **write pointer** location
- **Read** – at **read pointer** location
- **Reposition within 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

Open File Locking

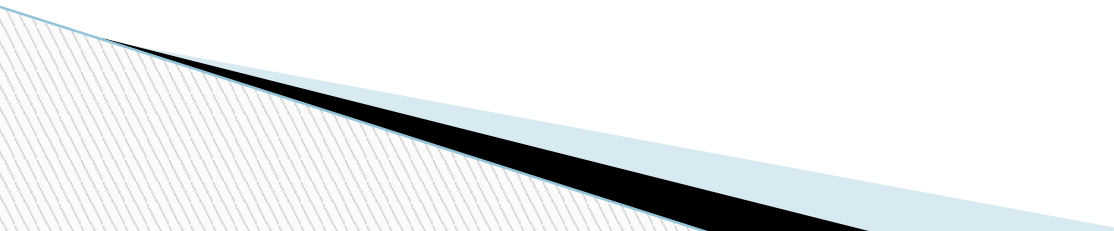
- Provided by some operating systems and file systems
 - Similar to reader-writer locks
 - **Shared lock** similar to reader lock – several processes can acquire concurrently
 - **Exclusive lock** similar to writer lock
 - Mediates access to a file
 - Mandatory or advisory:
 - **Mandatory** – access is denied depending on locks held and requested
 - **Advisory** – processes can find status of locks and decide what to do
- 

File Types – Name, Extension

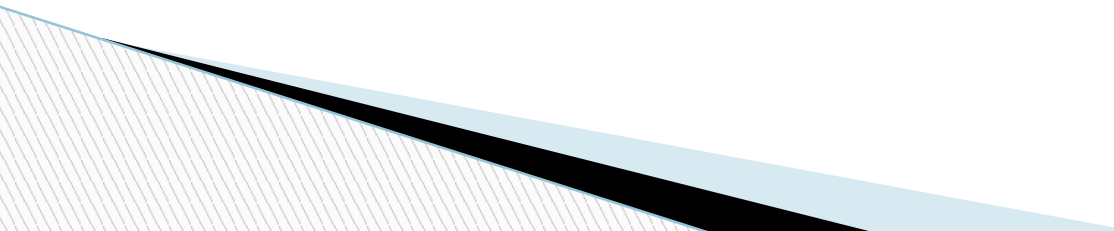
| file type | usual extension | function |
|----------------|--------------------------|---|
| executable | exe, com, bin or none | ready-to-run machine-language program |
| object | obj, o | compiled, machine language, not linked |
| source code | c, cc, java, pas, asm, a | source code in various languages |
| 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 |

File Access Mechanisms

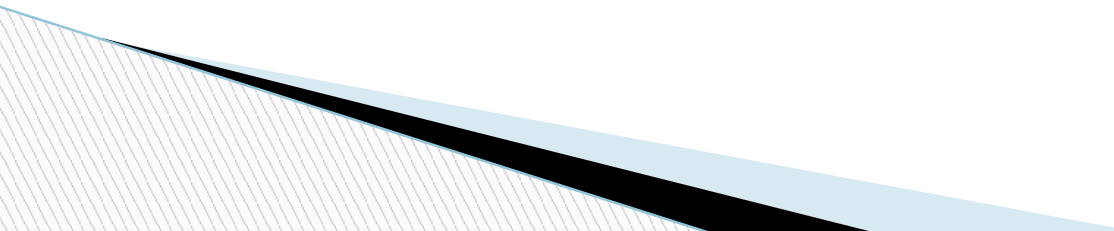
File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files –

- Sequential access
 - Direct/Random access
 - Indexed sequential access
- 

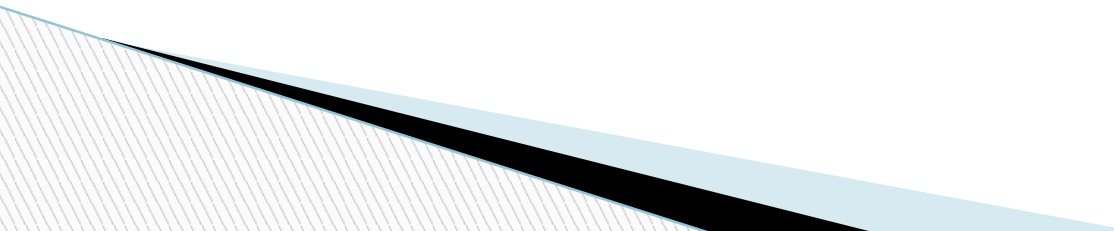
Sequential Access –

- ❑ It is the simplest access method.
 - ❑ Information in the file is processed in order, one record after the other. This mode of access is by far the most common; for example, editor and compiler usually access the file in this fashion.
 - ❑ Data is accessed one record right after another record in an order.
 - ❑ When we use read command, it move ahead pointer by one
 - ❑ When we use write command, it will allocate memory and move the pointer to the end of the file
 - ❑ Such method is reasonable for tape.
- 

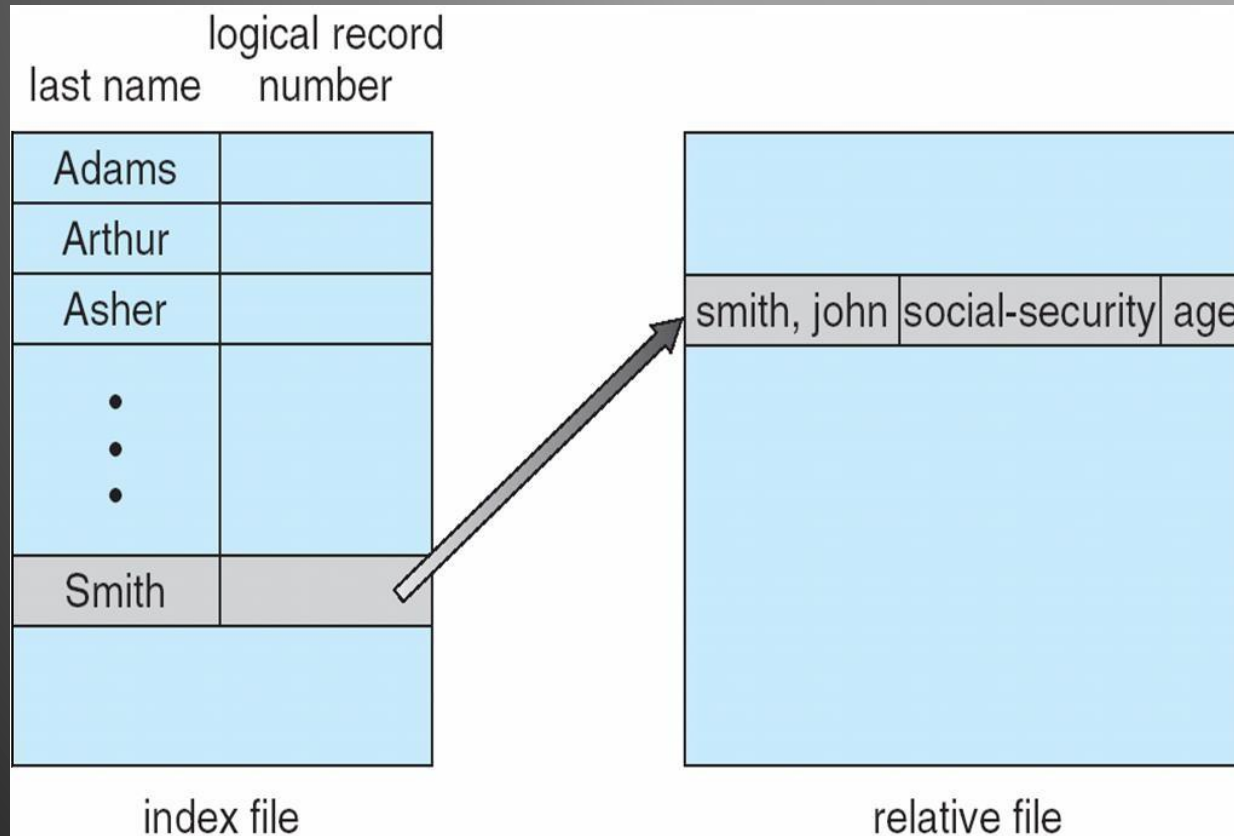
Direct Access –

- ❑ Another method is *direct access method* also known as *relative access method*.
 - ❑ A fixed-length logical record that allows the program to read and write record rapidly. in no particular order.
 - ❑ The direct access is based on the disk model of a file since disk allows random access to any file block.
 - ❑ For direct access, the file is viewed as a numbered sequence of block or record. Thus, we may read block 14 then block 59 and then we can write block 17.
 - ❑ There is no restriction on the order of reading and writing for a direct access file.
 - ❑ A block number provided by the user to the operating system is normally a *relative block number*, the first relative block of the file is 0 and then 1 and so on.
- 

Index sequential method –

- It is the other method of accessing a file which is built on the top of the direct access method.
 - These methods construct an index for the file.
 - The index, like an index in the back of a book, contains the pointer to the various blocks.
 - To find a record in the file, we first search the index and then by the help of pointer we access the file directly.
- 

Example of Index and Relative Files

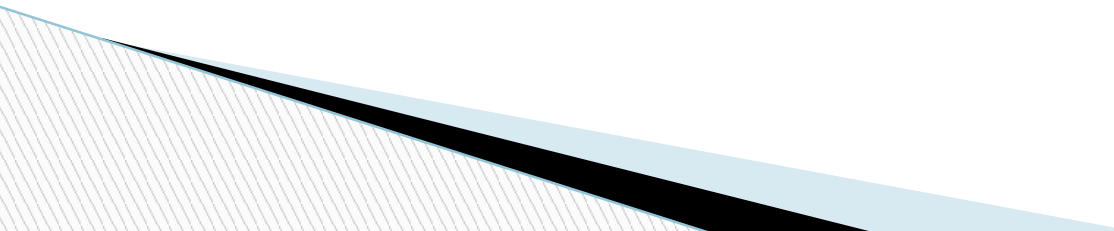


File Allocation Methods

The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

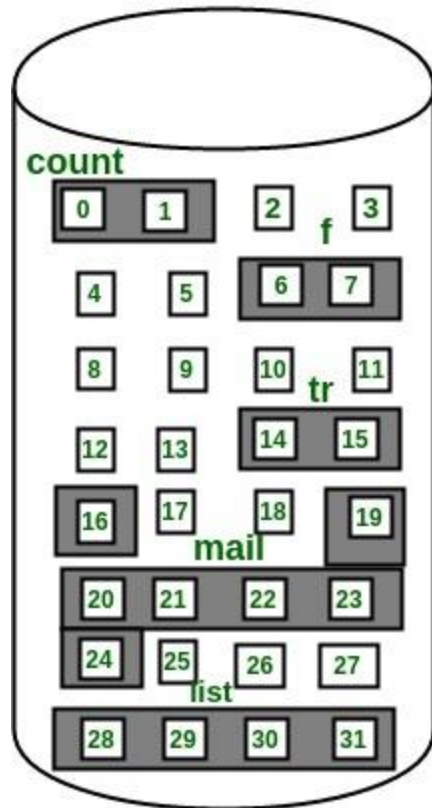
The main idea behind these methods is to provide:

- Efficient disk space utilization.
 - Fast access to the file blocks.
- 

1. Contiguous Allocation

- In this *scheme*, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: $b, b+1, b+2, \dots, b+n-1$.
- The directory entry for a file with contiguous allocation contains:
 - Address of starting block
 - Length of the allocated portion.

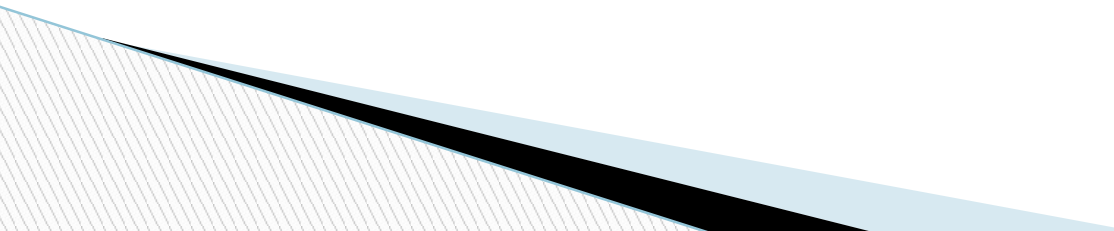
The *file* 'mail' in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



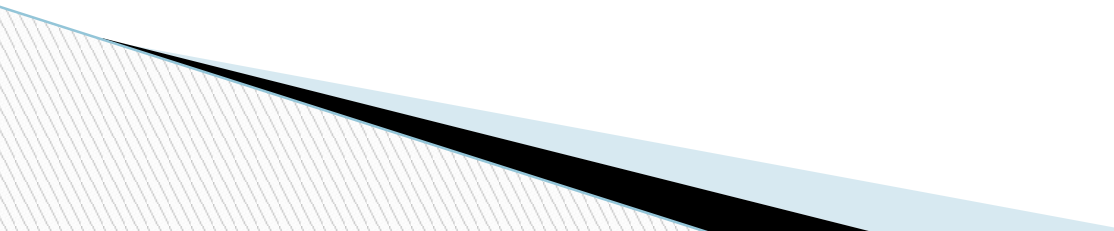
Directory

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

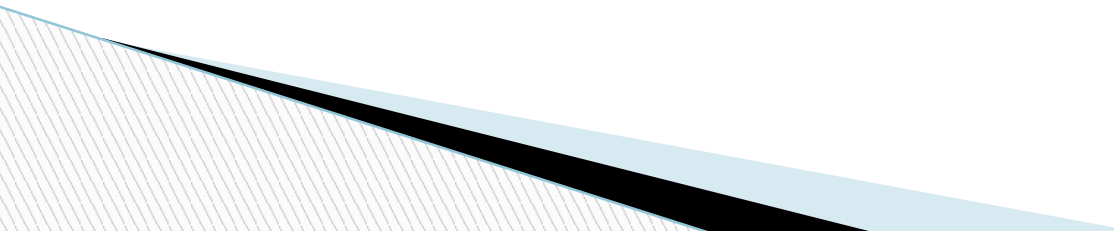
Advantages : Contiguous Allocation

- Both the Sequential and Direct Accesses are supported by this.
 - For direct access, the address of the k th block of the file which starts at block b can easily be obtained as $(b+k)$.
 - This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.
- 

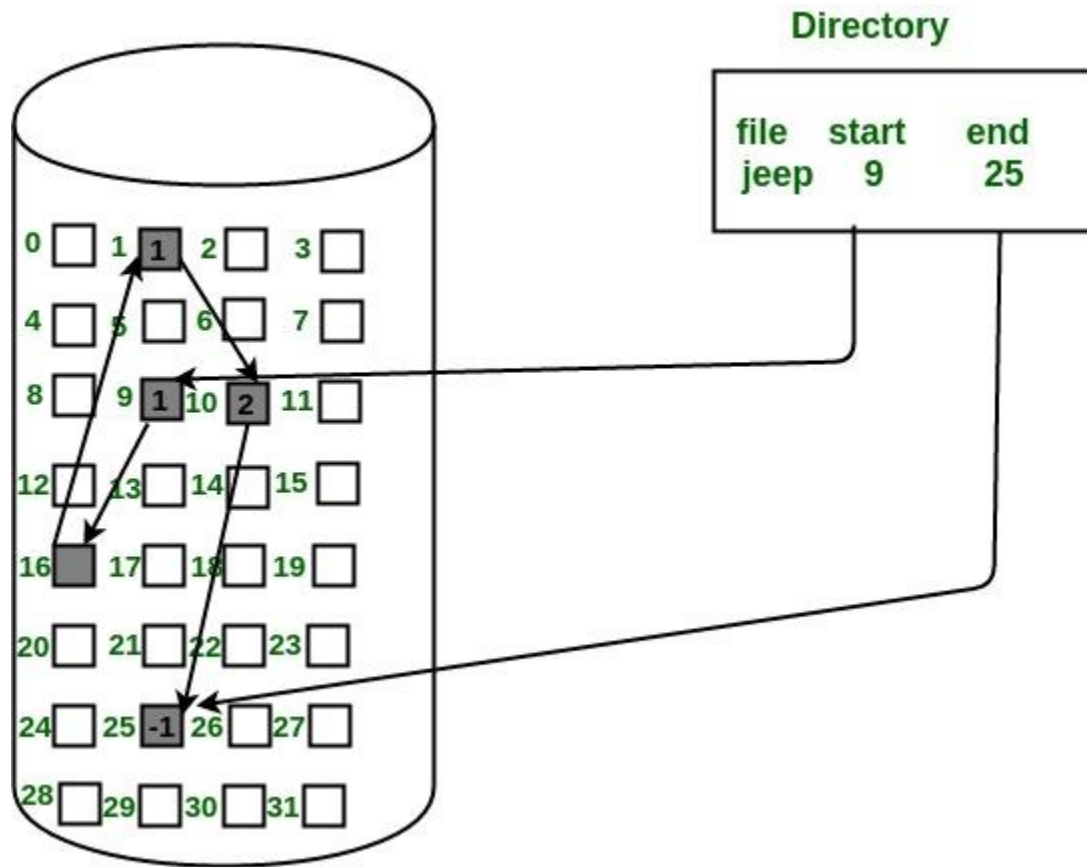
Disadvantages Contiguous Allocation

- This method suffers from both internal and external fragmentation.
 - This makes it inefficient in terms of memory utilization.
 - Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.
- 

2. Linked List Allocation

- In this scheme, each file is a linked list of disk blocks which **need not be** contiguous.
 - The disk blocks can be scattered anywhere on the disk.
 - The directory entry contains a pointer to the starting and the ending file block.
 - Each block contains a pointer to the next block occupied by the file.
- 

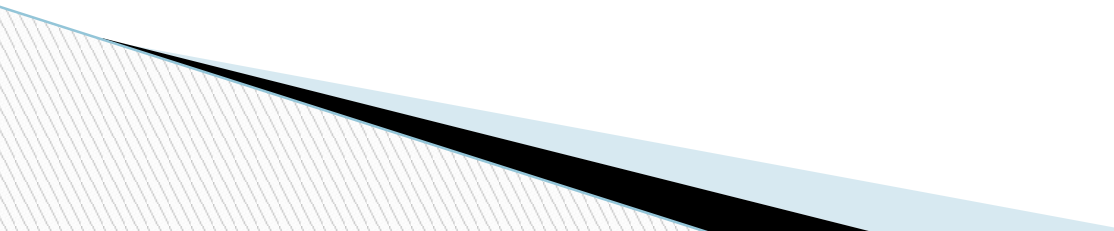
The file 'jeep' in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



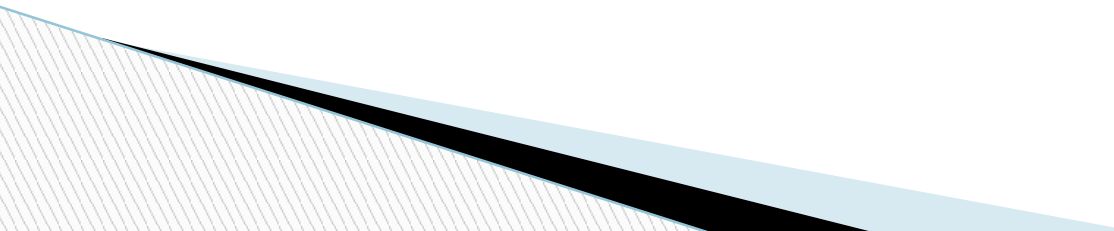
Advantages :Linked List Allocation

- This is very flexible in terms of file size.
- File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation.
This makes it relatively better in terms of memory utilization.

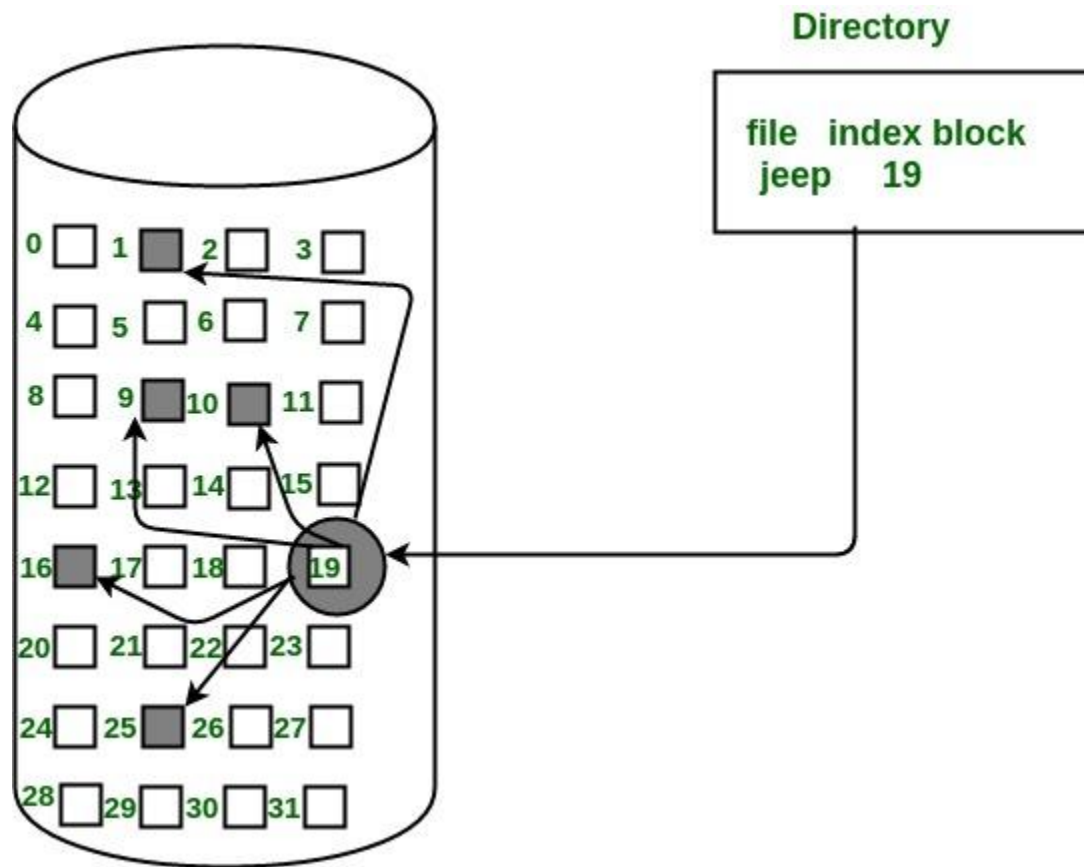
Disadvantages Linked List Allocation

- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually.
 - This makes linked allocation slower.
 - It does not support random or direct access. We can not directly access the blocks of a file.
 - A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
 - Pointers required in the linked allocation incur some extra overhead.
- 

3. Indexed Allocation

- In this scheme, a special block known as the **Index block** contains the pointers to all the blocks occupied by a file.
 - Each file has its own index block.
 - The i th entry in the index block contains the disk address of the i th file block.
 - The directory entry contains the address of the index block as shown in the image:
- 

Indexed Allocation



Advantages of Indexed Allocation

- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation

Disadvantages of Indexed Allocation

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization.
- However, in linked allocation we lose the space of only 1 pointer per block.

For files that are very large, single index block may not be able to hold all the pointers.

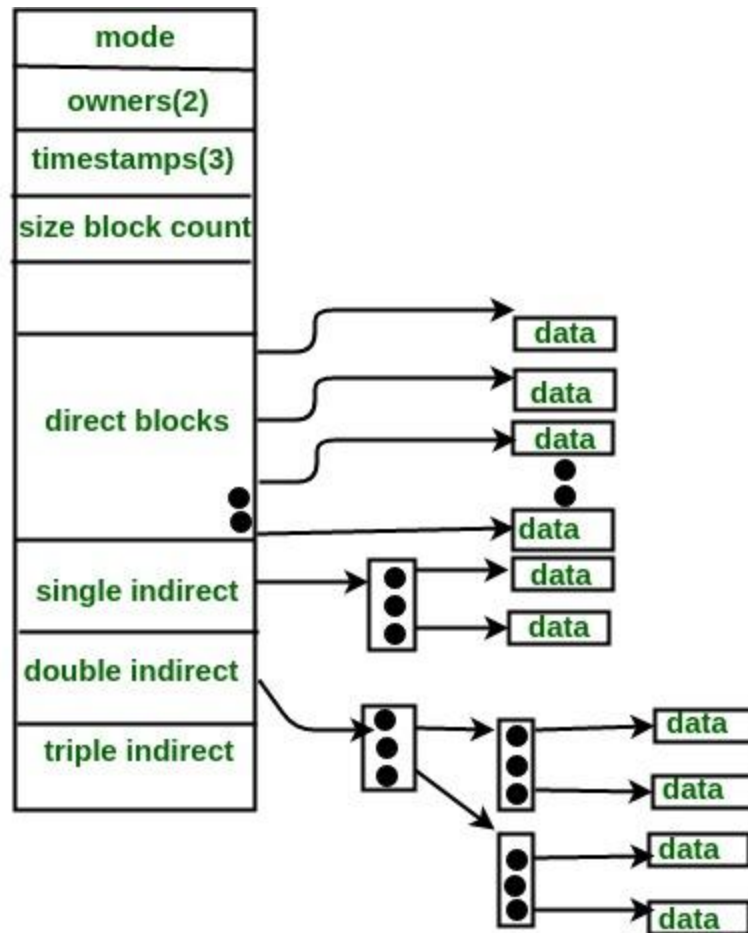
Following mechanisms can be used to resolve this:

Linked scheme: This scheme links two or more index blocks together for holding the pointers. Every index block would then contain a pointer or the address to the next index block.

Multilevel index: In this policy, a first level index block is used to point to the second level index blocks which in turn points to the disk blocks occupied by the file. This can be extended to 3 or more levels depending on the maximum file size.

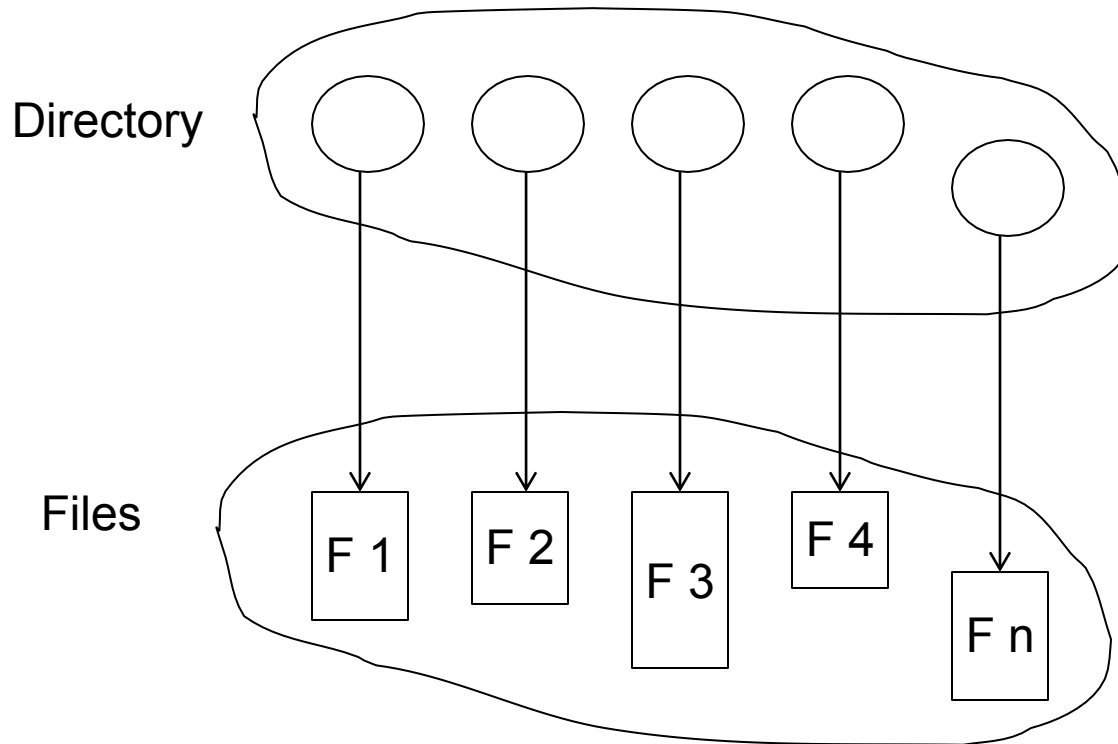
Combined Scheme: In this scheme, a special block called the **Inode (information Node)** contains all the information about the file such as the name, size, authority, etc and the remaining space of Inode is used to store the Disk Block addresses which contain the actual file





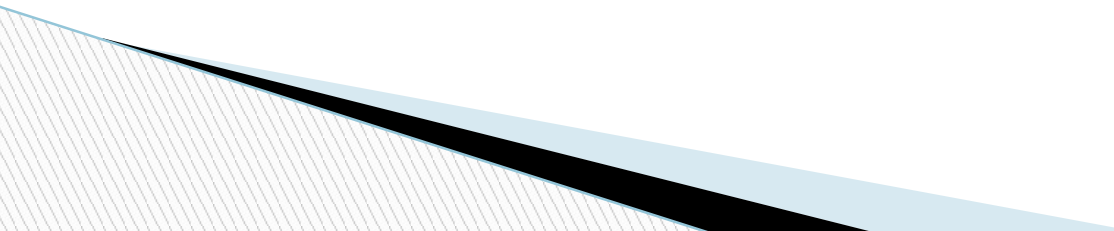
Directory Structure

- A collection of nodes containing information about all files

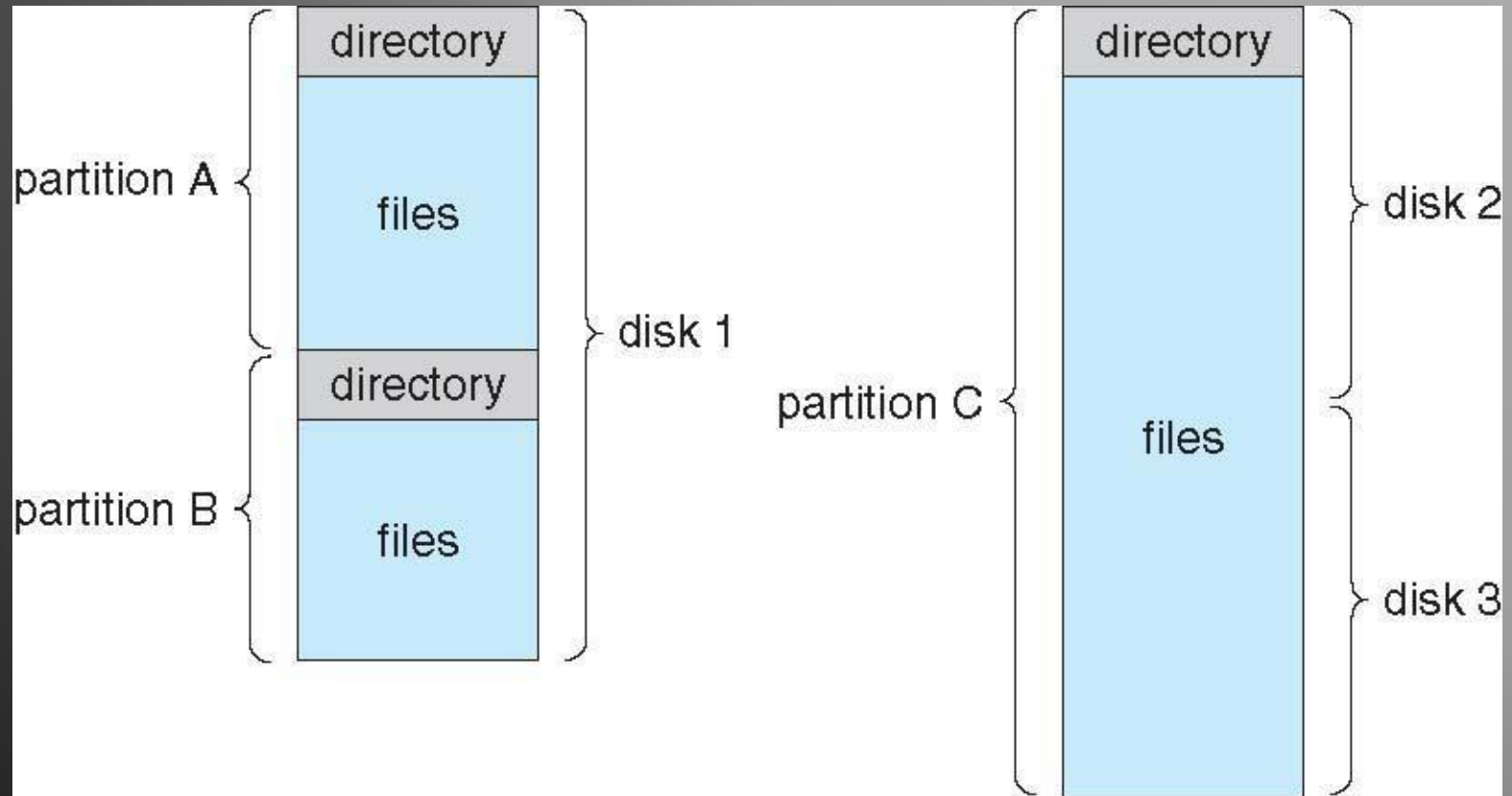


Both the directory structure and the files reside on disk

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

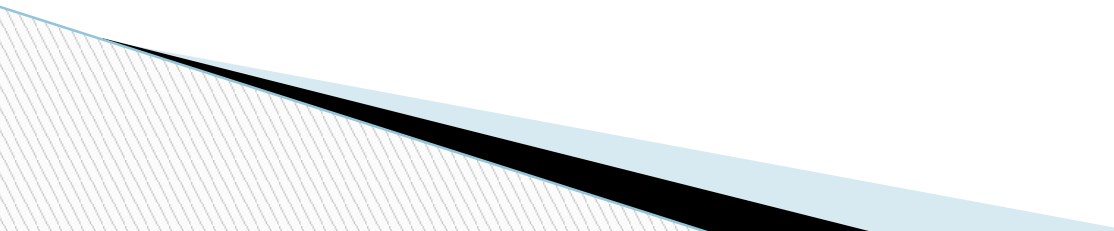




Types of File Systems

- We mostly talk of general-purpose file systems
- But systems frequently have many file systems, some general- and some special- purpose
- Consider Solaris has
 - tmpfs – memory-based volatile FS for fast, temporary I/O
 - objfs – interface into kernel memory to get kernel symbols for debugging
 - ctfb – contract file system for managing daemons
 - lofs – loopback file system allows one FS to be accessed in place of another
 - procfs – kernel interface to process structures
 - ufs, zfs – general purpose file systems

Operations Performed on Directory

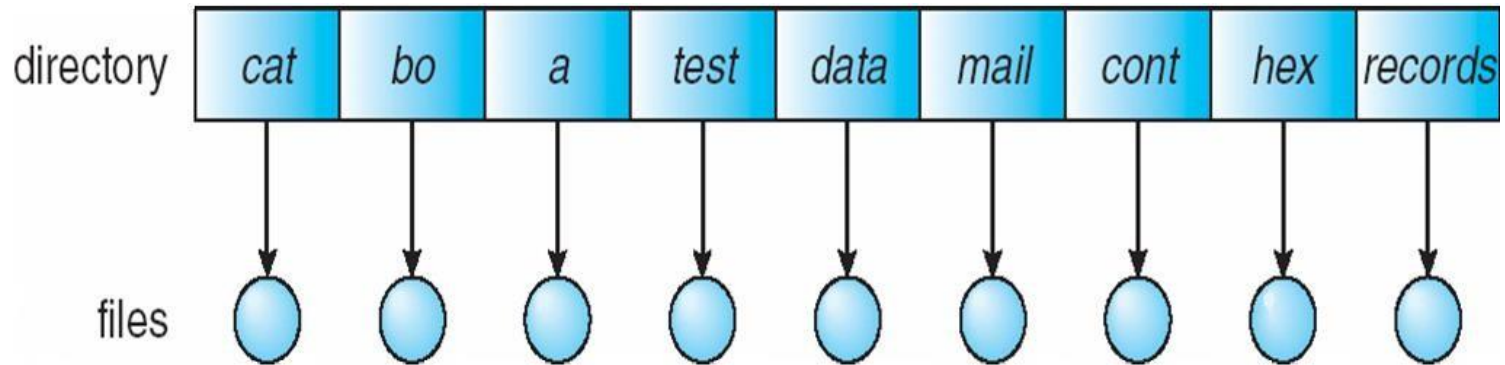
- Search for a file
 - Create a file
 - Delete a file
 - List a directory
 - Rename a file
 - Traverse the file system
- 

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

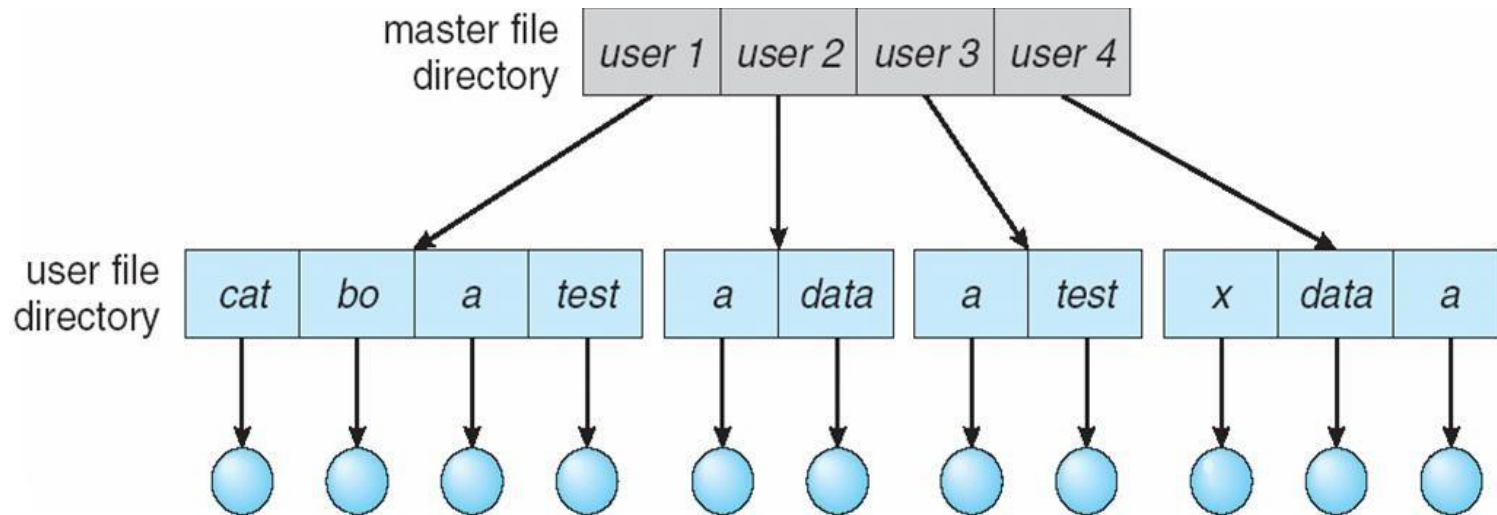


Naming problem

Grouping problem

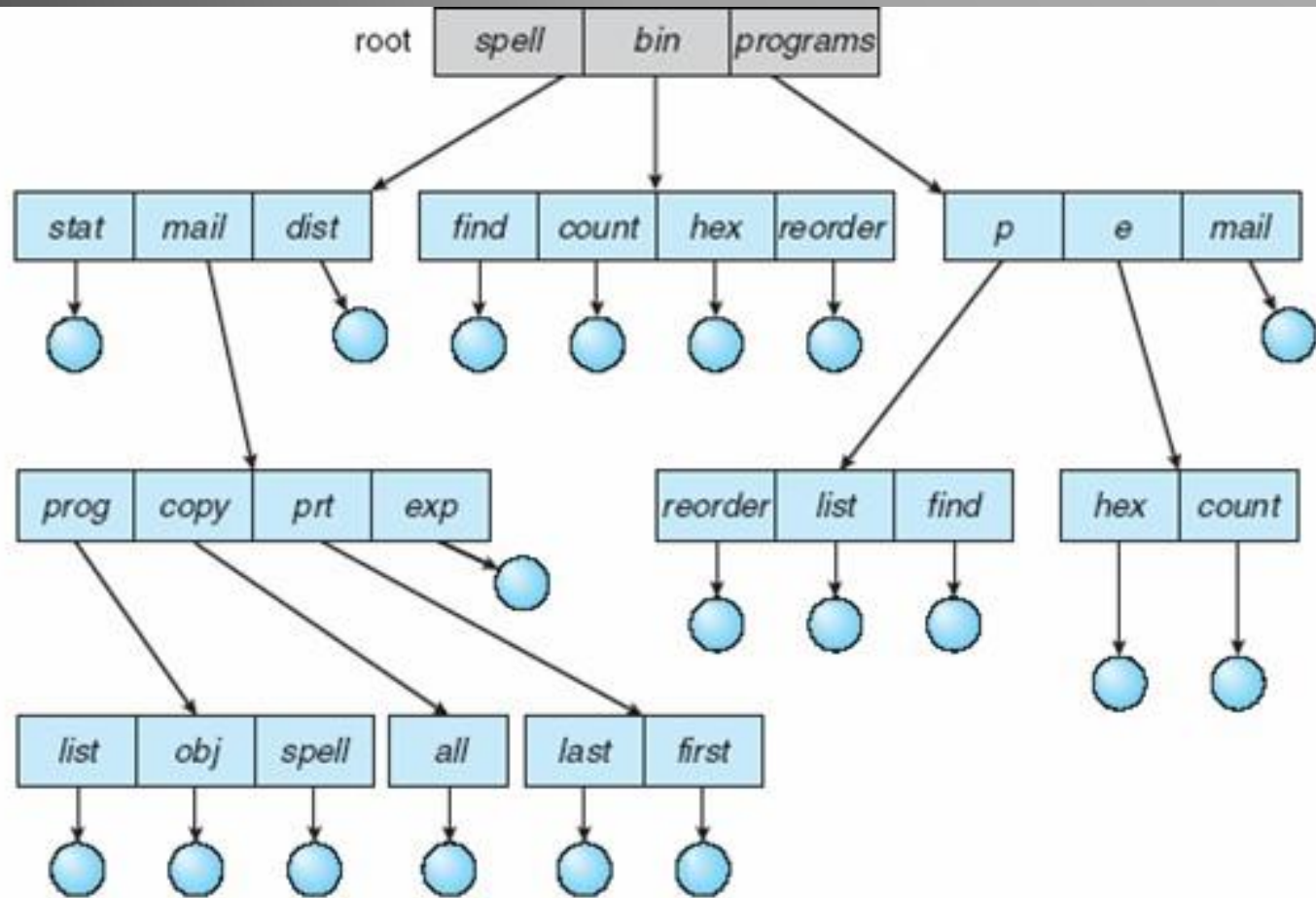
Two-Level Directory

- Separate directory for each user



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

Tree-Structured Directories



Tree-Structured Directories (Cont.)

- 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

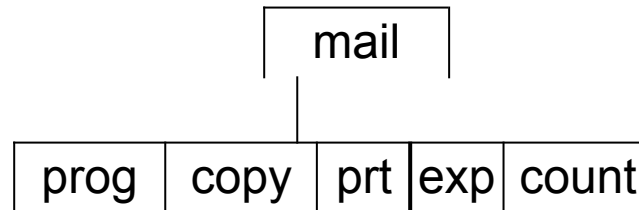
```
rm <file-name>
```

- Creating a new subdirectory is done in current directory

```
mkdir <dir-name>
```

Example: if in current directory `/mail`

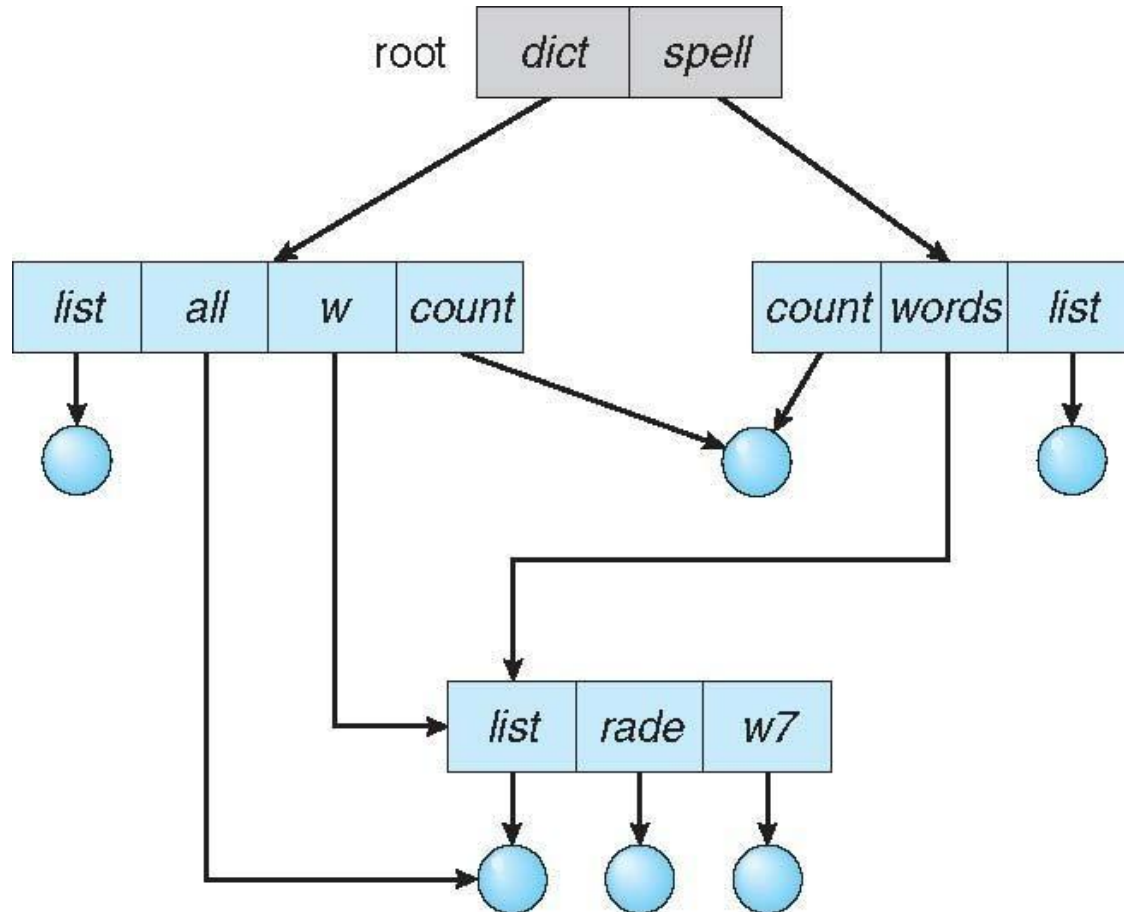
```
mkdir count
```



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

Acyclic-Graph Directories

- Have shared subdirectories and files



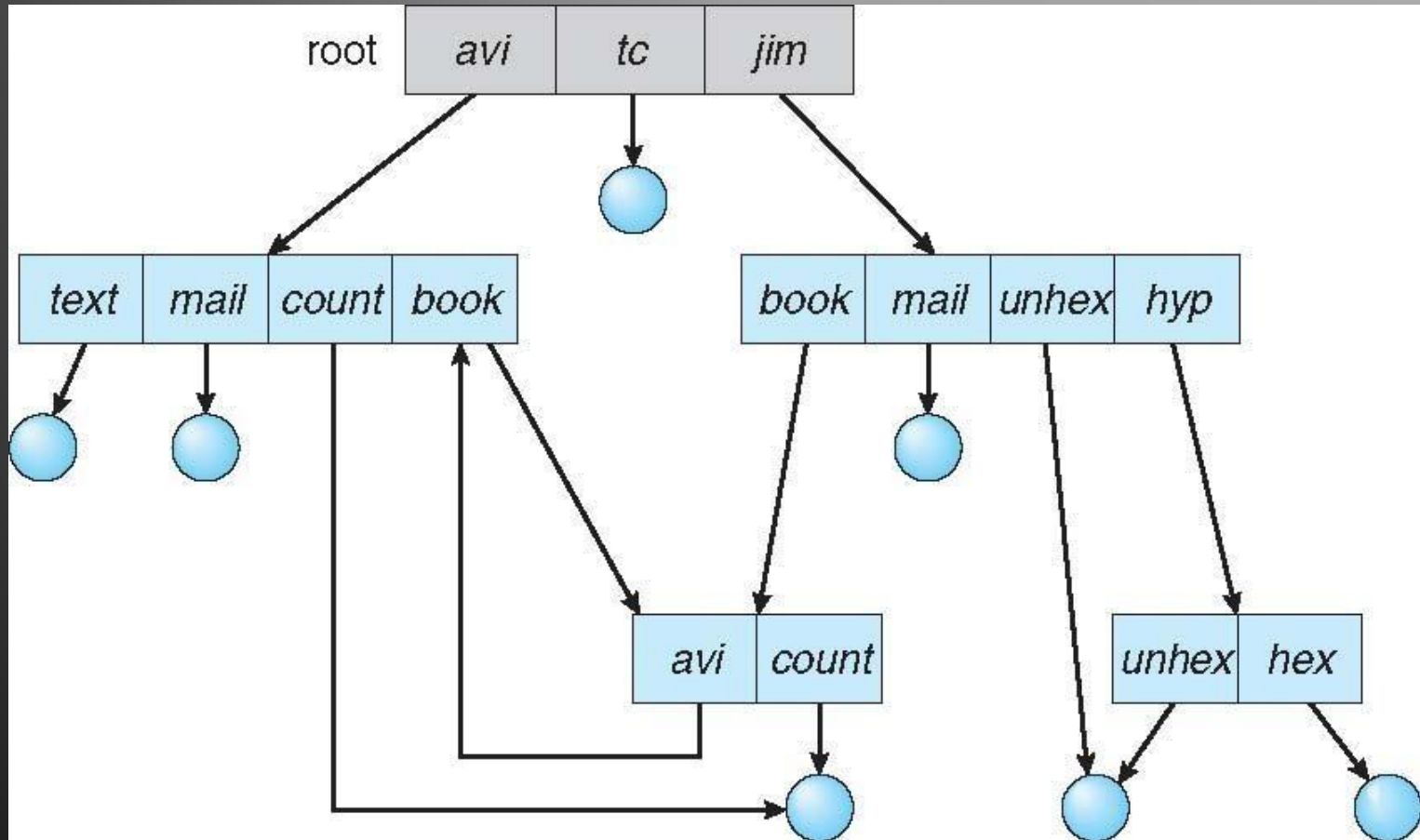
Acyclic-Graph Directories (Cont.)

- Two different names (aliasing)
- If **dict** deletes **list** \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
-
- New directory entry type
 - **Link** – another name (pointer) to an existing file
 - **Resolve the link** – follow pointer to locate the file

General Graph Directory

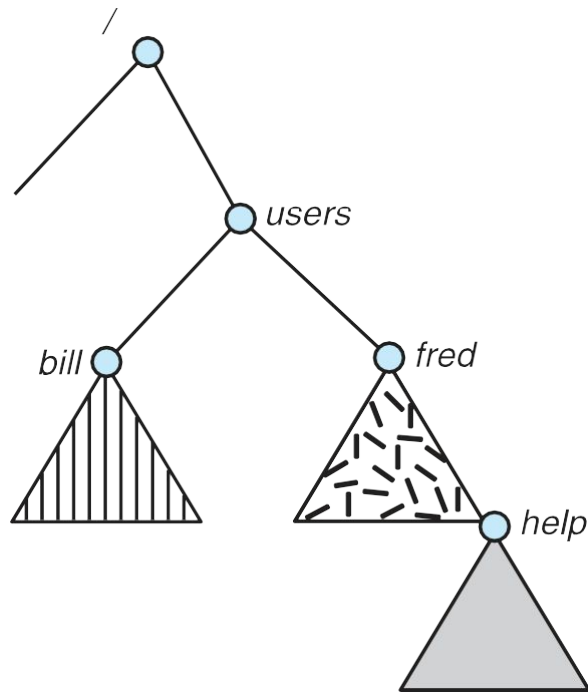


General Graph Directory (Cont.)

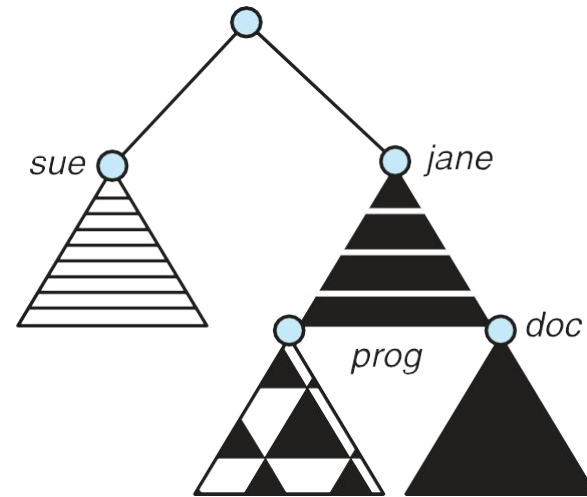
- How do we guarantee no cycles?
 - Allow only links to file not subdirectories
 - **Garbage collection**
 - Every time a new link is added use a cycle detection algorithm to determine whether it is OK

File System Mounting

- A file system must be **mounted** before it can be accessed
- A unmounted file system (i.e., Fig. 10-11(b)) is mounted at a **mount point**

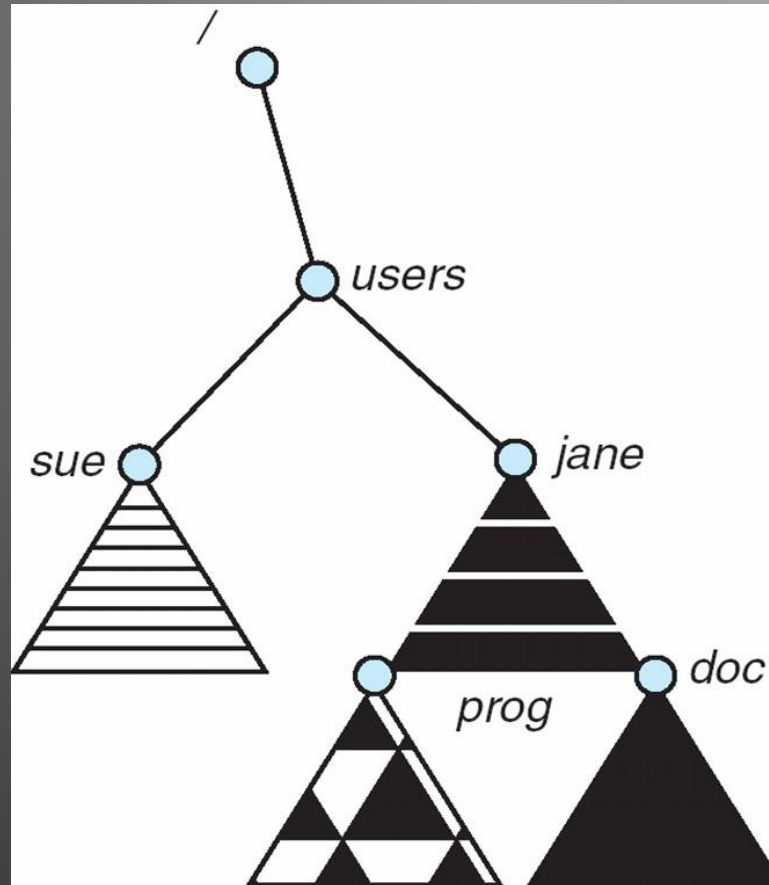


(a)



(b)

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
- If multi-user system
 - **User IDs** identify users, allowing permissions and protections to be per-user
 - **Group IDs** allow users to be in groups, permitting group access rights
 - Owner of a file / directory
 - Group of a file / directory

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

- All file systems have failure modes
 - For example corruption of directory structures or other non-user data, called **metadata**
- 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 v3 include all information in each request, allowing easy recovery but less security

File Sharing – Consistency Semantics

- Specify how multiple users are to access a shared file simultaneously
 - Similar to Ch 6 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

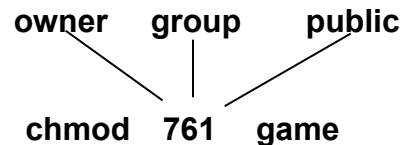
- Types of access
 - **Read**
 - **Write**
 - **Execute**
 - **Append**
 - **Delete**
 - **List**

Access Lists and Groups

- Mode of access: read, write, execute
- Three classes of users on Unix / Linux

| | | | |
|-------------------------|---|---|--------------|
| a) owner access | 7 | ⇒ | RWX 1 1 1 |
| b) group access | 6 | ⇒ | RWX 1 1 0 |
| c) public access | 1 | ⇒ | RWX 0 0 1 |

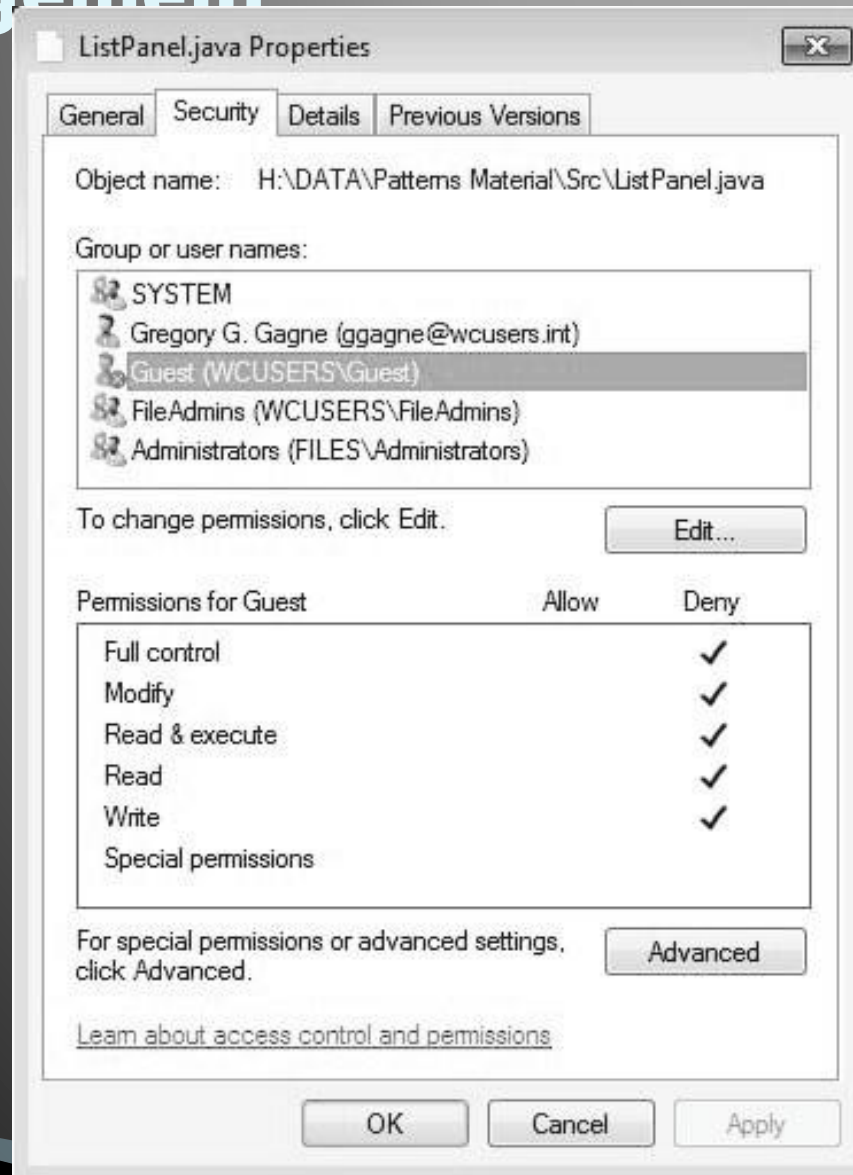
- Ask manager to create a group (unique name), say G, and add some users to the group.
- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp **G** **game**

Windows 7 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/ |