Operating Systems: Internals and Design Principles

Chapter 9
File Systems

Eighth Edition

By William Stallings

Files Concept

- **File** = collection of related information that is ***recorded on secondary storage***.
- Files are mapped by OS onto physical devices.
- We don't store file in a memory. We will only ***put file in a memory, ready for execution.***

Files Concept

- Types:
  - Data File
    - Numeric Data (0,1,2,3…)
    - Character (a, A, B, b, c…)
    - Binary (0,1)
  - Program File
- The information of the file is defined by its creator.
- Contents of file can be source program, graphic image, record, text and etc.

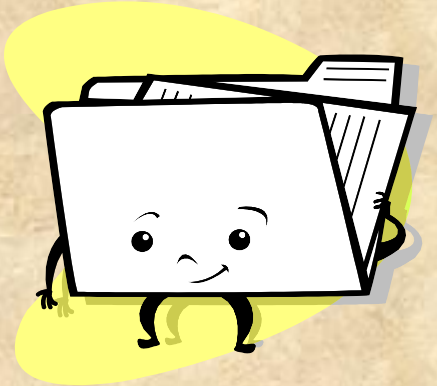
File System

File system

- provides mechanism for on-line storage and access to both data and program for OS and user’s program.
- Maintain a set of attributes associated with the file
  - These include owner, creation time, time last modified, access privileges, and so on.
- Consist of 2 parts:
  - Collection of files – storing related data
  - Directory structure – organizes and provides information about all files in system

File Operations

- Typical operations include:
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write



File Operations

- **Create:** A new file is defined and positioned within the structure of files.
- **Delete:** A file is removed from the file structure and destroyed.
- **Open:** An existing file is declared to be “opened” by a process, allowing the process to perform functions on the file.
- **Close:** The file is closed with respect to a process, so that the process no longer may perform functions on the file, until the process opens the file again.
- **Read:** A process reads all or a portion of the data in a file.
- **Write:** A process updates a file, either by adding new data that expands the size of the file or by changing the values of existing data items in the file.

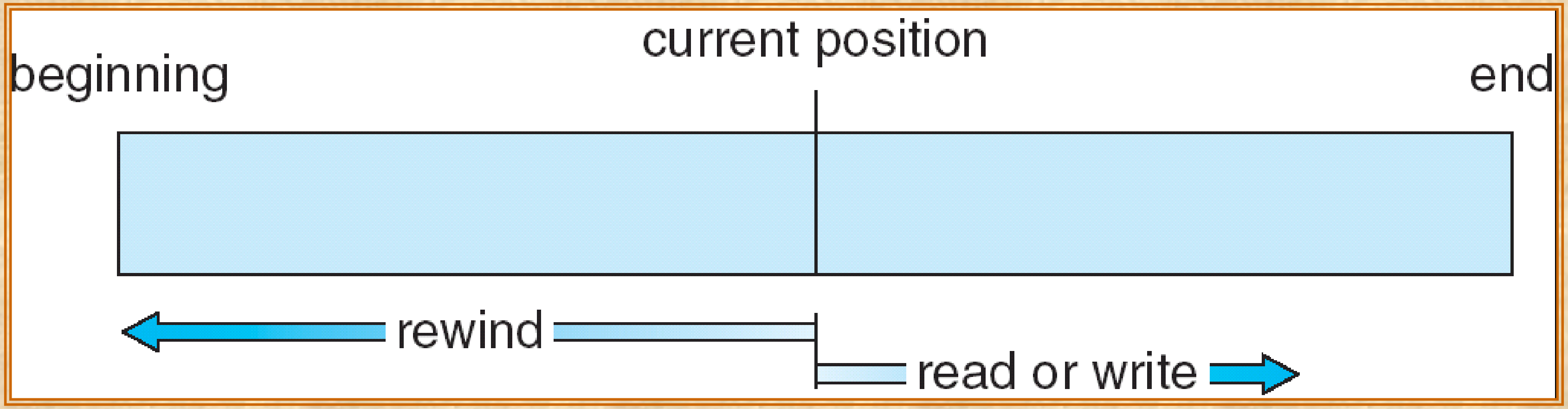
Access Methods

- File stores information.
- When it is used, this information must be accessed and read into computer memory.
- Access methods:
  - Sequential
  - Direct
  - indexed

Sequential Access

- Info in the file is processed **IN ORDER**, one record after another.
- Usually used by editors and compilers (have checked sequence)
- **Operations:**
  - Read next – reads next portion of file & automatically advances a pointer to I/O location
  - Write next – appends to end of the file & advances to new end of a file.
  - reset – reset to beginning
  - rewind/forward – skip some by OS.

Sequential Access



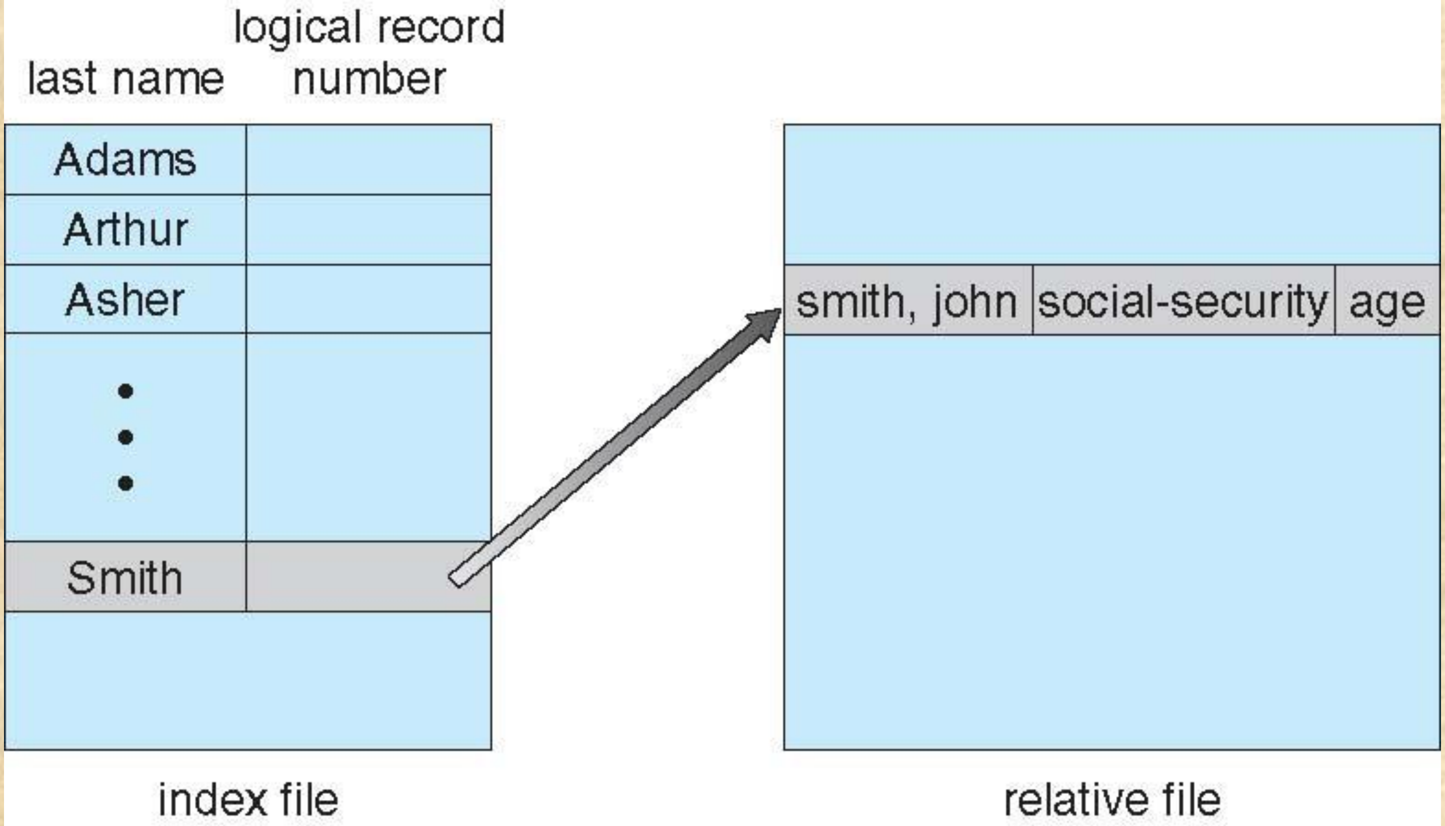
Direct/Relative Access Method

- A file is made up of a fixed-length logical records.
- **No particular order** for read or write
- File is viewed as a **numbered sequence of blocks/records**.
- Can begin from anywhere, **current position is changeable.**
- Is great use for immediate access to large amounts of information.
  - database.

INDEXED ACCESS METHOD

- Like **index on a book**
- Use **pointer** to point/locate to various blocks
- Steps:
  - search the index
  - use pointer to access the file

Index and Relative Files

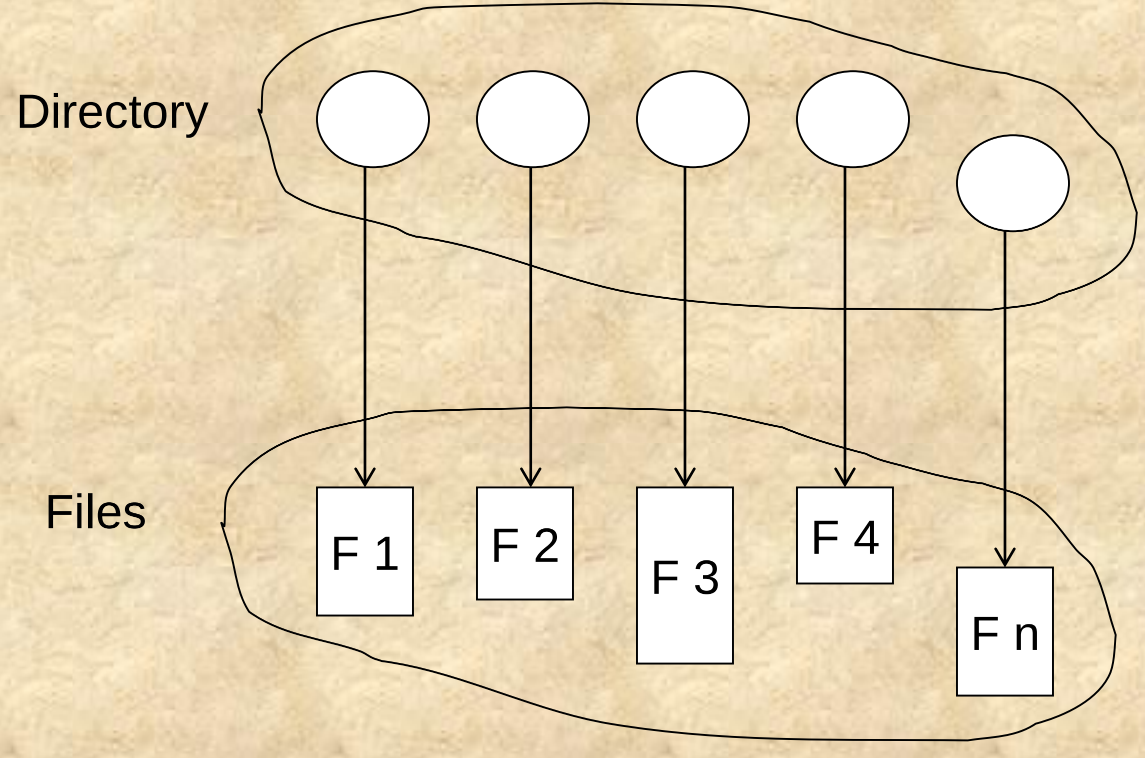


Directory Structure

- File system of computers can be extensive.
- Organization of all these files usually done in two parts:
  - Disks are split into one or more partitions.
  - Each partition contains information about file within it.
    - This information is kept in a device directory
- Device directory records information:
  - Name
  - Location
  - Size
  - type

Directory Structure

- A collection of nodes containing information about all files



Both the directory structure and the files *reside on disk*

Types of Directory

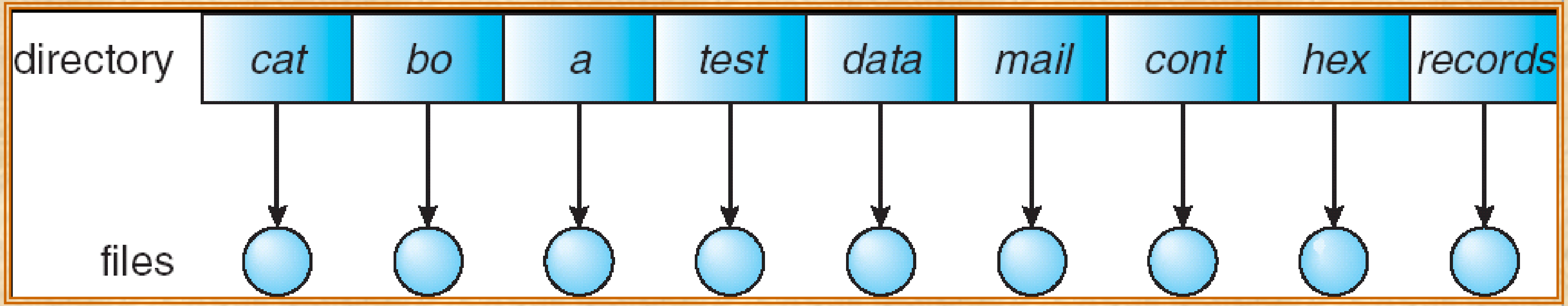
- SINGLE LEVEL DIRECTORY
- TWO LEVEL DIRECTORY

Single Level Directory

- **KEY WORD**:- All files contained in **same** directory.
- Advantage:-Is the **simplest directory structure** to support and understand
- Disadvantage:-
  - problem if **number of files increased** or system has more than one user.
  - **naming problem**-files must holding unique names since all are in same directory and it’s difficult to user to remember all file’s names.
  - **grouping problem**-it’s difficult to group the files since they are different in types.
  - brings **confusion of file names** among different users.

Single Level Directory

- A single directory for all users



Naming problem

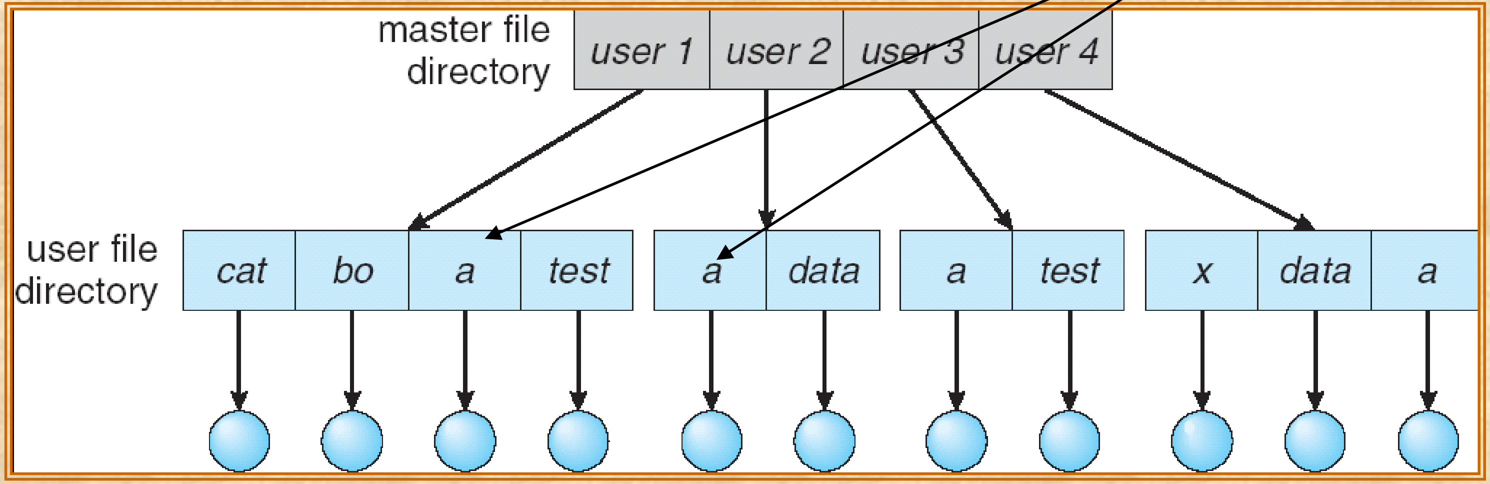
Grouping problem

Two Level Directory

- Solve confusion of file's names when many users in single-level directory.
- KEY WORD:- **Separate directory for each user**
- Each user has his own **UFD (User File Directory).** List only for single user
- When job's started or user logs in, **MFD (Master File Directory)** is searched. MFD is indexed by username/account number. Each entry points to UFD.
- Different users can have same file's name, but the file's name must be unique from others in the UFD.

Two Level Directory

- Separate directory for each user



Different users, same file's name

- Path name- ***from MFD to UFD***
- Can have the same file name for different user
- Efficient searching
- No grouping capability

File Sharing

- Sharing of files on multi-user systems is desirable to achieve computing goals.
- 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** (owner) identify users, allowing permissions and protections to be per-user, users who can change attributes and control the file

- **Group IDs** allow users to be in groups, permitting group access rights, is a subset of users who can share access to the file.

File Sharing- Client-Server Model

- **SERVER** = machine with the file, provide the file
- **CLIENT** = machine seeking for the file, request the file
- Operations:-
  - Server declares if resource/file is available to clients
  - Server specify exactly which file/resource and which client to allow for access
  - Server can allow multiple clients and client also can access multiple server

Protection

- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

File Allocation

- On secondary storage, a file consists of a collection of blocks
- The operating system or file management system is responsible for allocating blocks to files
- Space is allocated to a file as one or more ***portions*** (contiguous set of allocated blocks)
- ***File allocation table (FAT)***
    - data structure used to keep track of the portions assigned to a file

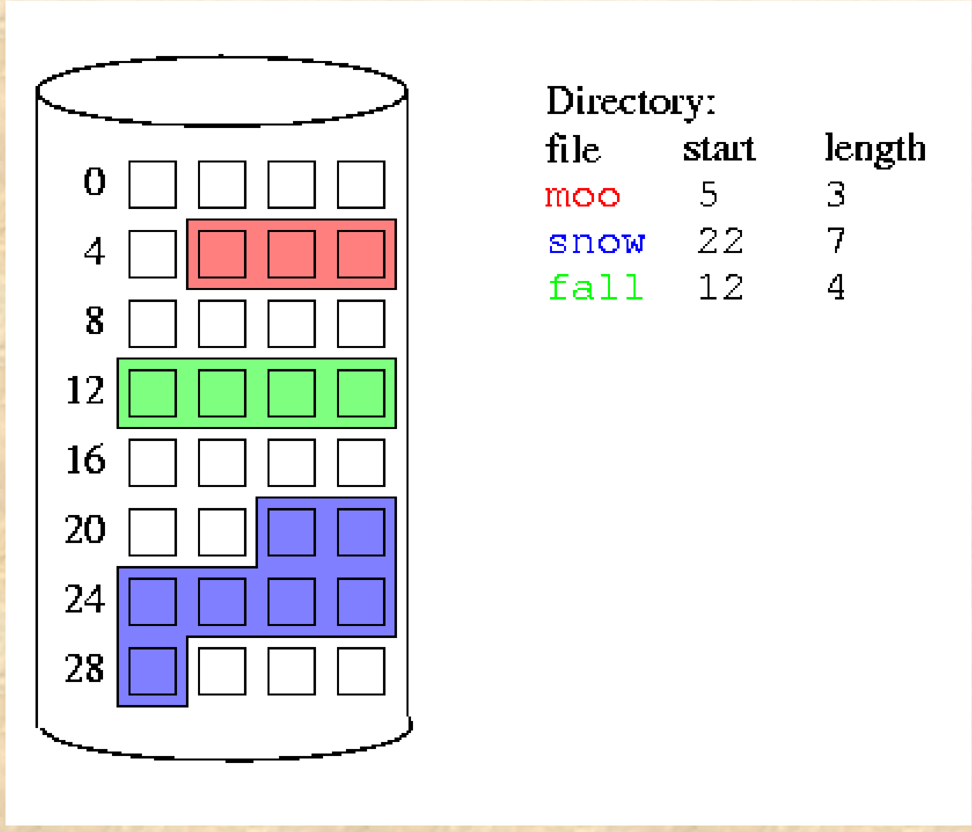
File Allocation methods

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

Contiguous Allocation

- Each file has to occupy contiguous blocks on the disk.
- The location of a file is defined by the disk address of the first block and its length.
- Both sequential access and direct access are supported by the contiguous allocation.
- Uses First fit and Best fit to select a free hole from the set of available holes
- The disadvantages:
  - it is often difficult to find free space for a new file.
  - one is often not sure of the space required while creating a new file.
- The various methods adopted to find space for a new file suffer from external fragmentation.

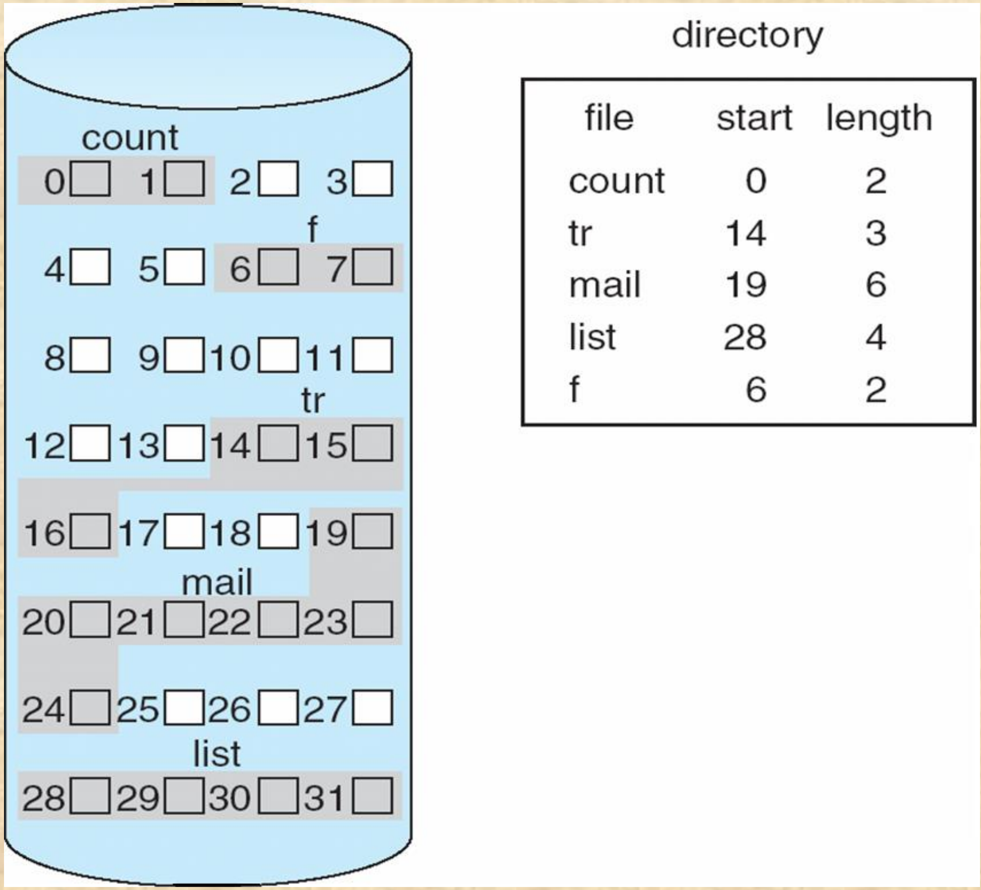
Contiguous Allocation



Directory:

file	start	length
moo	5	3
snow	22	7
fall	12	4

Contiguous Allocation



directory

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

Linked Allocation

- In linked allocation, each file is a linked list of disk blocks.
- The directory contains a pointer to the first and (optionally the last) block of the file.
- Solves all problems of contiguous allocation
- This pointer is initialized to nil (the end-of-list pointer value) to signify an empty file.
- Need to reserve part of each data block for a pointer

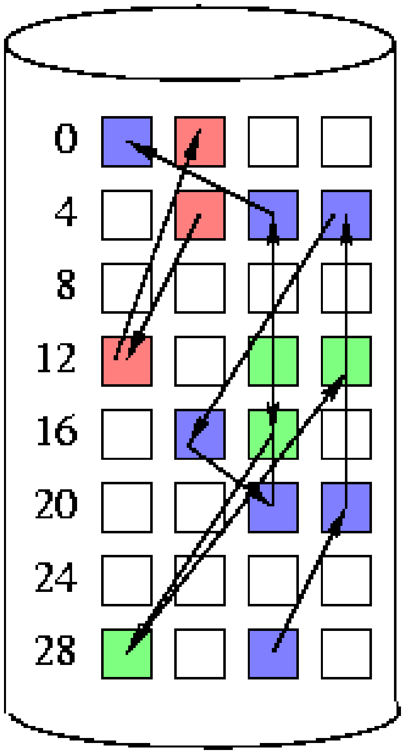
Linked Allocation

- To read a file, the pointers are just followed from block to block.
- For example:
  - A file of 5 blocks which starts at block 4, might continue at block 7, then block 16, block 10, and finally block 27.
  - Each block contains a pointer to the next block and the last block contains a NIL pointer.
  - The value -1 may be used for NIL to differentiate it from block 0.

Linked Allocation

- There is no external fragmentation with linked allocation.
- Any free block can be used to satisfy a request. There is no need to declare the size of a file when that file is created.
- A file can continue to grow as long as there are free blocks.
- Disadvantages:
  - It is inefficient to support direct-access; it is effective only for sequential-access files.
  - To find the *n*th block of a file, it must start at the beginning of that file and follow the pointers until the *n*th block is reached. Each access to a pointer requires a disk read

Linked Allocation



Directory

File	Start	End
moo	5	1
snow	30	0
fall	14	15

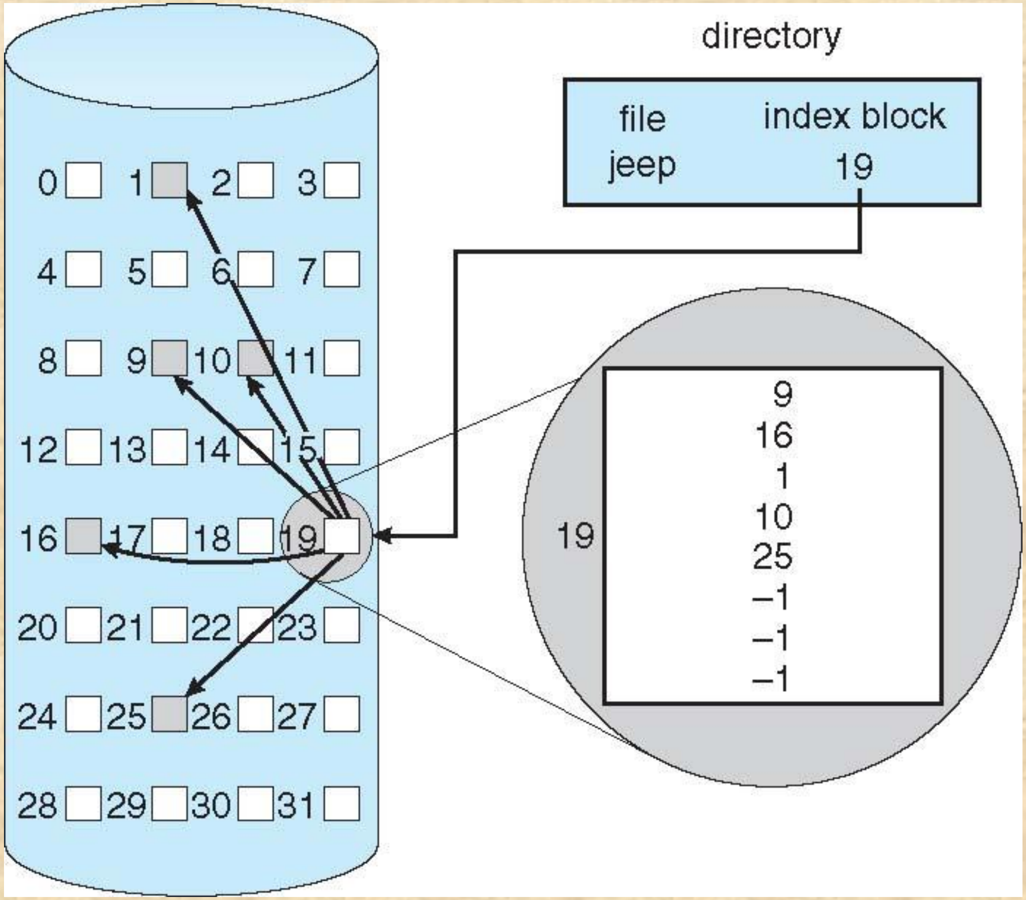
Indexed Allocation

- Linked allocation does not support random access of files, since each block can only be found from the previous.
- Indexed allocation solves this problem by bringing all the pointers together into one location: the index block.
- This type of allocation will have a pointer which has the address of all the blocks of a file.
- This method solves the problem of fragmentation as the blocks can be stored in any location.

Indexed Allocation

- Some disk space is wasted because an entire index block must be allocated for each file, regardless of how many data blocks the file contains.
- This leads to questions of how big the index block should be, and how it should be implemented.

Indexed Allocation



Summary

- File systems
- File operations
- File access methods
  - sequential
  - indexed
  - direct
- Directory Structure
  - Single level
  - Second level
- File Sharing

- Secondary storage management
  - File allocation