OPERATING SYSTEMS

FILE MANAGEMENT

Files

- The File System is one of the most important parts of the OS to a user
- Desirable properties of files
  - **Long-term existence**
    Files are stored on disk or other secondary storage and do not disappear when a user logs off
  - **Sharable between processes**
    Files have names and can have associated access permissions that permit controlled sharing
  - **Structure**
    Files can be organized into hierarchical or more complex structure to reflect the relationships among files

File Systems

- Provide a means to store data organized as files as well as a collection of functions that can be performed on files
- Maintain a set of attributes associated with the file
- Typical operations include
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write

File Concept

- Contiguous logical address space
- Types
  - Data
    - numeric
    - character
    - binary
  - Binary Executable
- Contents defined by file's creator
  - Many types
    - e.g., text file, source file, executable file, picture, sound, etc.

File Attributes

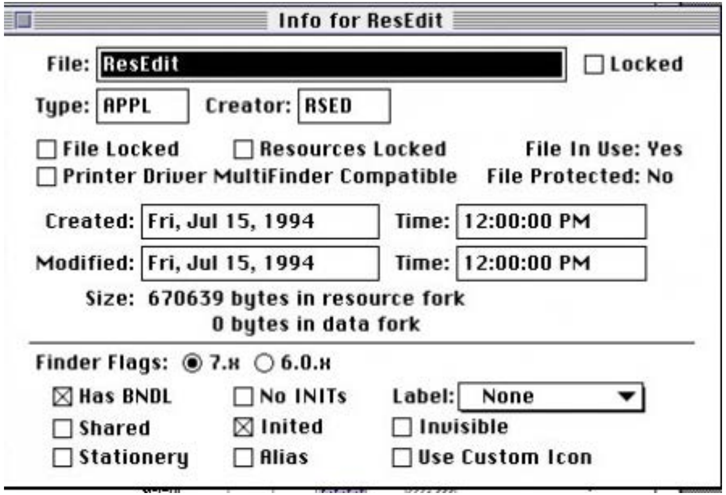
- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls 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
- Many variations, including extended file attributes such as file checksum

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 Type and File Creator (Classic Macintosh System)

https://www.macintoshrepository.org/1317-resedit-2-1-3



- File Types (4 characters)

code	*file type*
TEXT	text file
PDF	PDF
GIFI	GIF file
8BPS	Photoshop PSD
WDBN	MS Word

- File Creator (4 characters)

signature	*application*
8BIM	Photoshop
ttxt	Simple text
MSWD	MS WORD
XPR3	QuarkXpress
????	Unknown

http://livecode.byu.edu/helps/file-creatorcodes.php

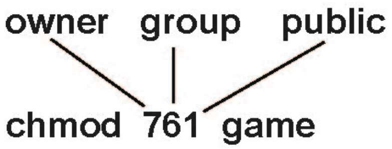
File Protection

- 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



Attach a group to a file **chgrp G game**

File Structure

- None - sequence of words, bytes
- Simple record structure
  - Lines
  - Fixed length
  - Variable length
- Complex Structures
  - Formatted document
  - Relocatable load file

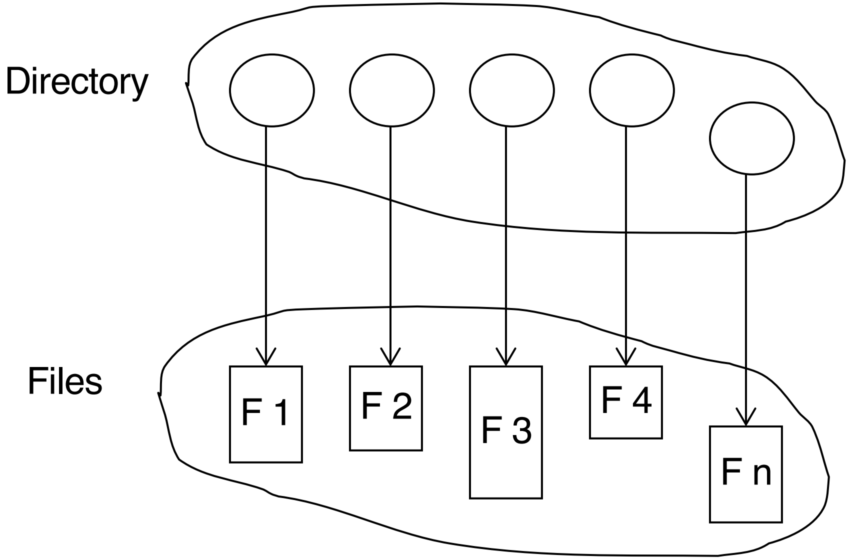
Open Files

- The OS needs the following information to manage open files
  - **Open-file table**: tracks open files
  - **File pointer**: pointer to last read/write location, per process that has the file open
  - **File-open count**: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
  - **Disk location of the file**: cache of data access information
  - **Access rights**: per-process access mode information

File Directories

Directory Structure

A collection of nodes containing information about all files



Both the directory structure and the files reside on disk

Information Elements of a File Directory

- **Basic Information**
  - **File Name**
    Name as chosen by creator (user or program). Must be unique within a specific directory.
  - **File Type**
    For example: text, binary, load module, etc.
  - **File Organization**
    For systems that support different organizations
- **Address Information**
  - **Volume**
    Indicates device on which file is stored
  - **Starting Address**
    Starting physical address on secondary storage (e.g., cylinder, track, and block number on disk)
  - **Size Used**
    Current size of the file in bytes, words, or blocks
  - **Size Allocated**
    The maximum size of the file

Information Elements of a File Directory

- **Access Control Information**
  - **Owner**
    User who is assigned control of this file. The owner may be able to grant/deny access to other users and to change these privileges.
  - **Access Information**
    A simple version of this element would include the user's name and password for each authorized user.
  - **Permitted Actions**
    Controls reading, writing, executing, transmitting over a network

Information Elements of a File Directory

- **Usage Information**
  - Date Created
  - Identity of Creator
  - Date Last Read Access
  - Identity of Last Reader
  - Date Last Modified
  - Identity of Last Modifier
  - Date of Last Backup
  - Current Usage
    Information about current activity on the file, such as process or processes that have the file open, whether it is locked by a process, etc.

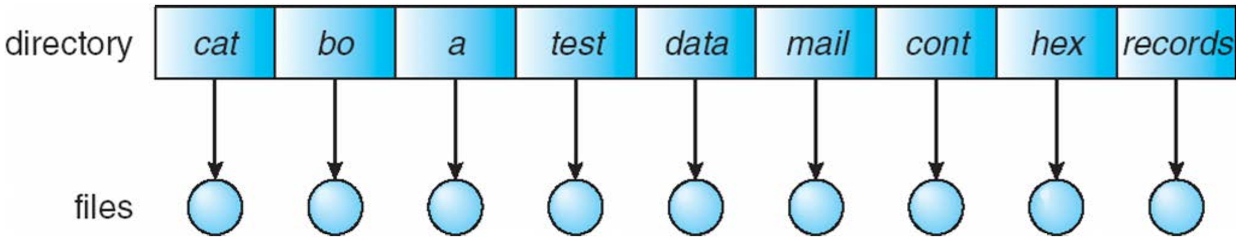
Operations Performed on Directory

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

Directory Organization

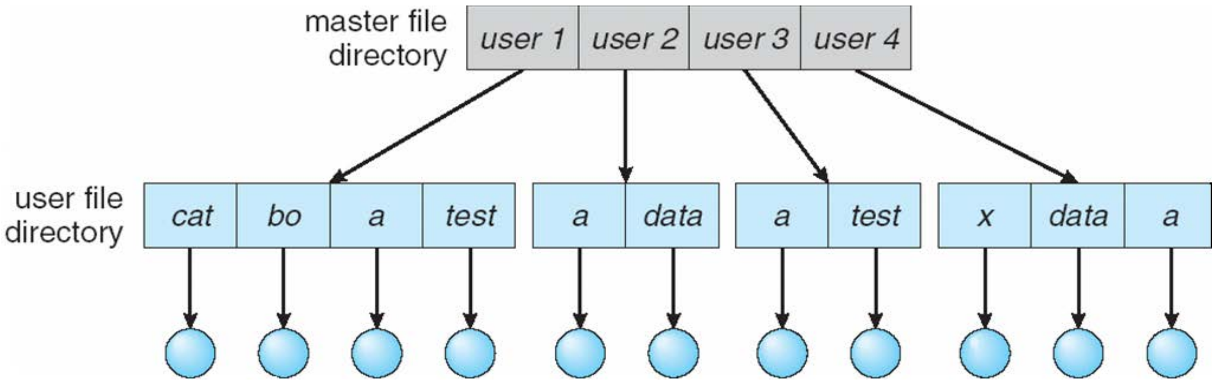
- The directory organization aims at attaining
  - **Efficiency**
    locating a file quickly
  - **Naming**
    convenient to users
    - Two users can have same name for different files
    - The same file can have several different names
  - **Grouping**
    logical grouping of files by properties, e.g., all Java programs, all games, …

Single-Level Directory



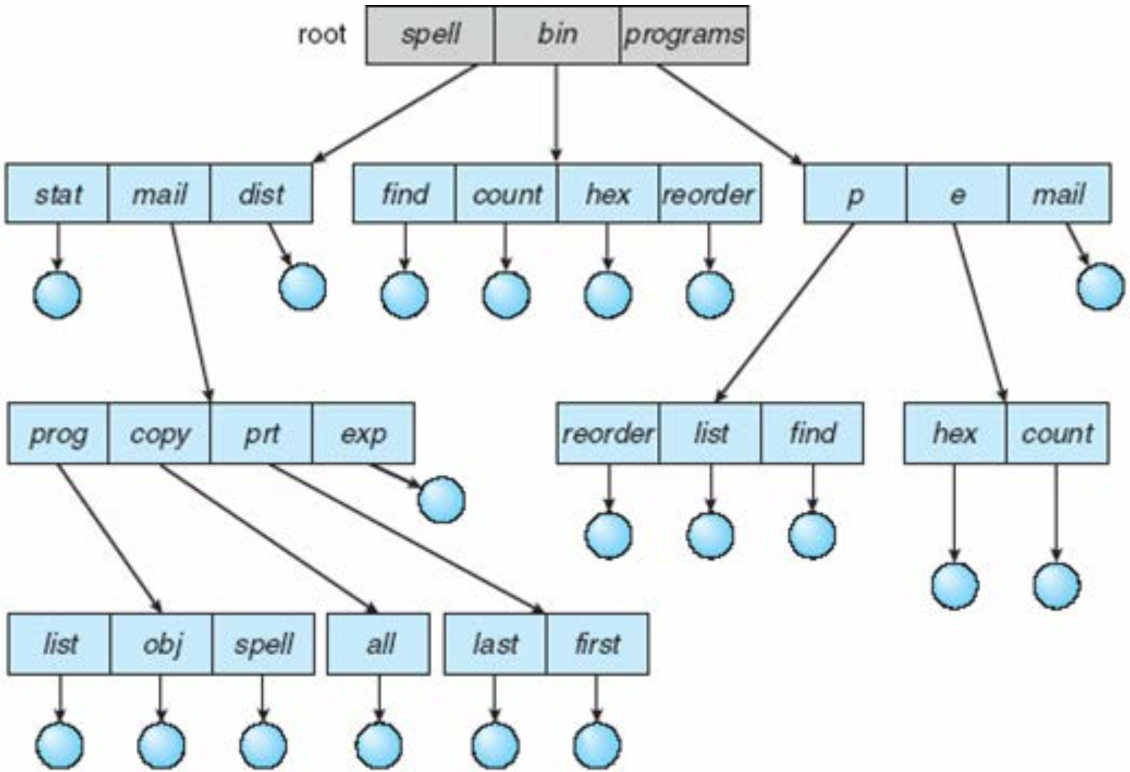
- Naming problem and Grouping problem with multiple users
- Solution used to store shared content among users
  - Filenames assigned automatically to avoid conflicts
  - Useful for documents and media in repositories, and streaming services

Two-Level Directory



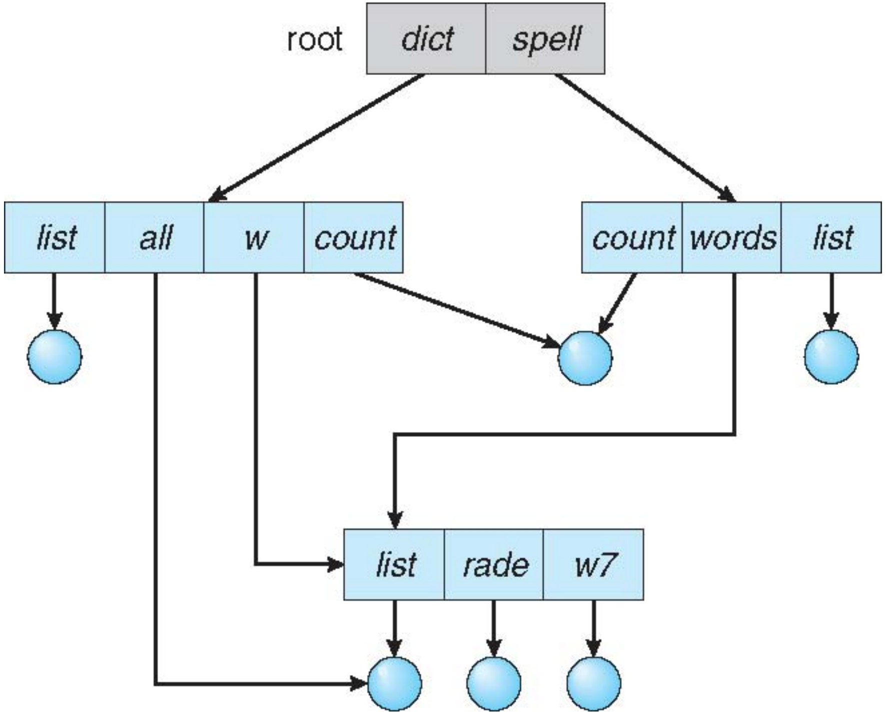
- Separate directory for each user
  - Need to define the Path name
  - Can have the same file name for different user
  - Efficient searching
  - No grouping capability

Tree-Structured Directories

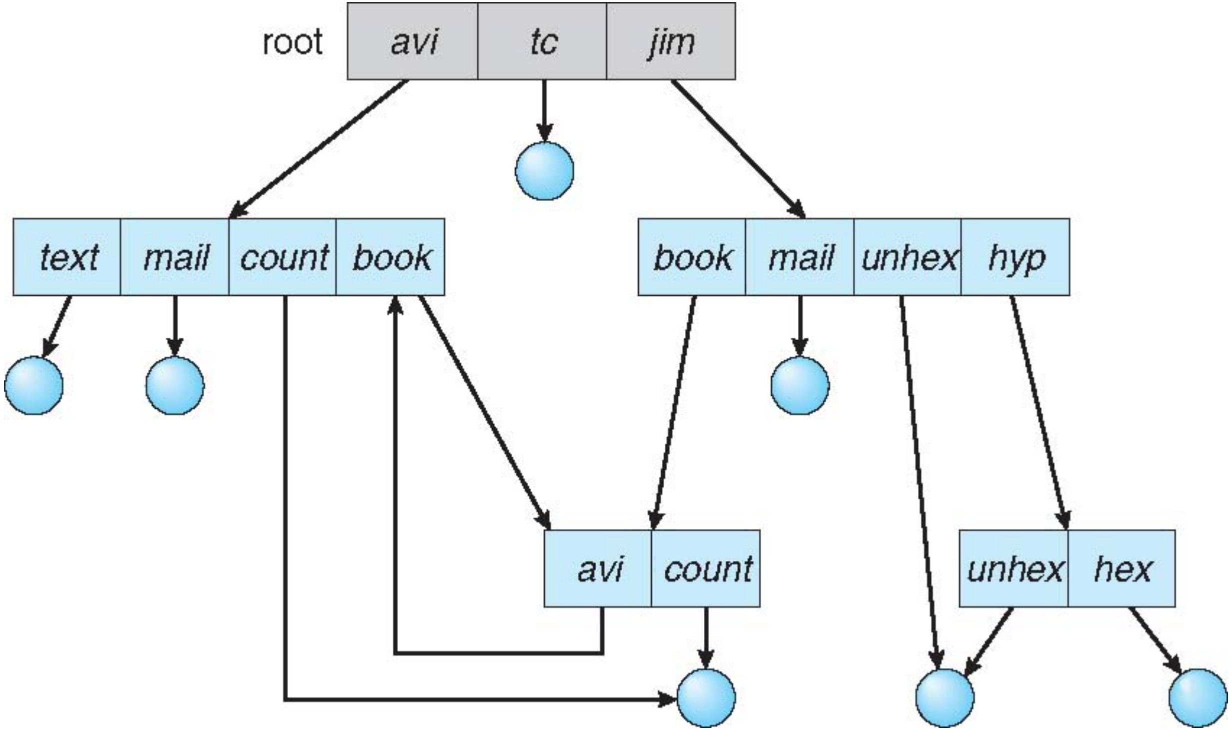


Acyclic-Graph Directories

- Have shared subdirectories and files

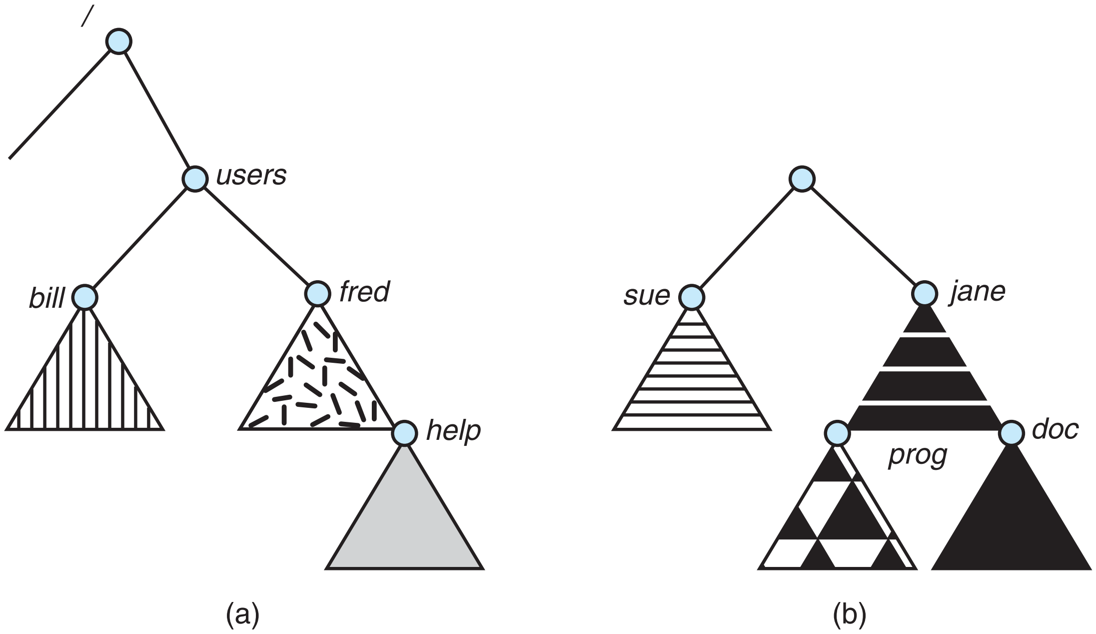


General Graph Directory



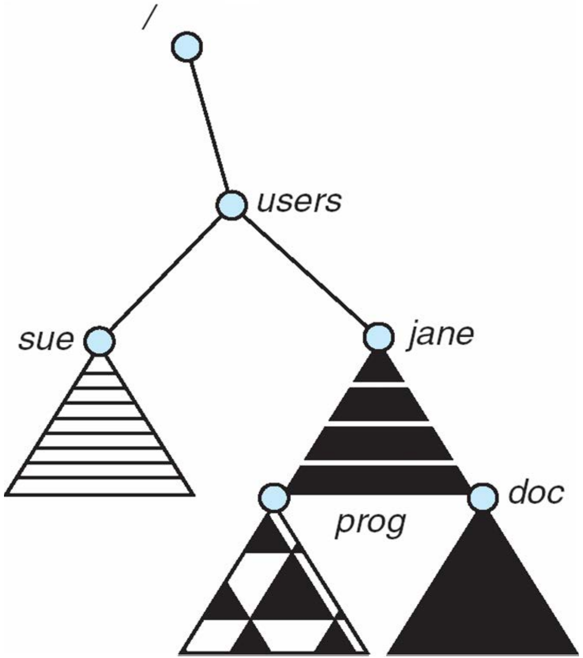
File System Mounting

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



Mount Point

- If the filesystem b) is mounted in filesystem a) at the `users` mount point, the resulting file system is

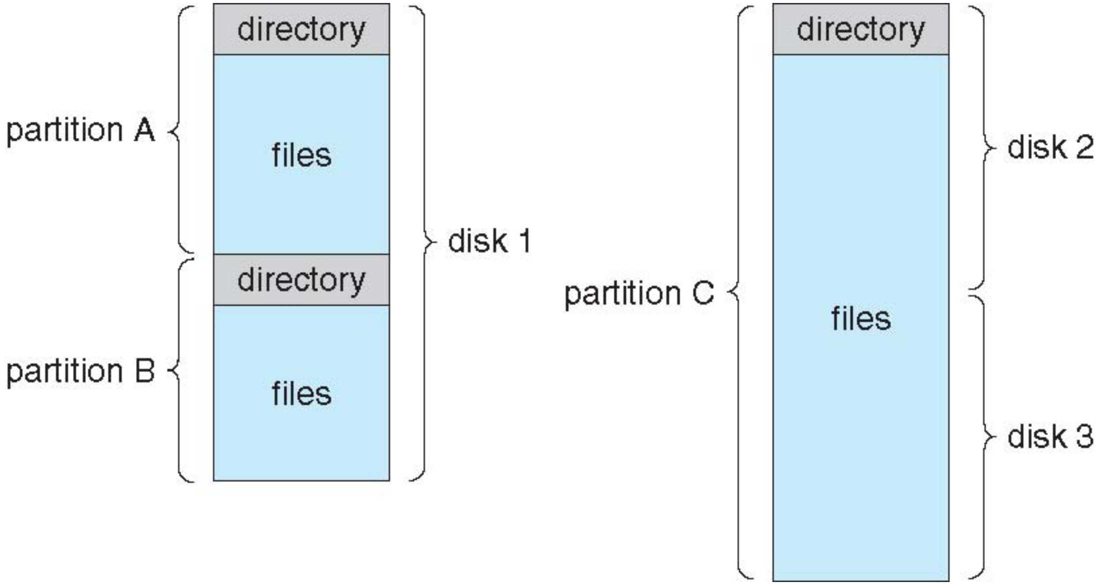


File Systems

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
- 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**
  - e.g, videosurveillance systems

A Typical File-system Organization



File Management System

- Meet the data management needs of the user
- Guarantee that the data in the file are valid
- Optimize performance
- Provide I/O support for a variety of storage device types
- Minimize the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines to user processes
- Provide I/O support for multiple users in the case of multiple-user systems

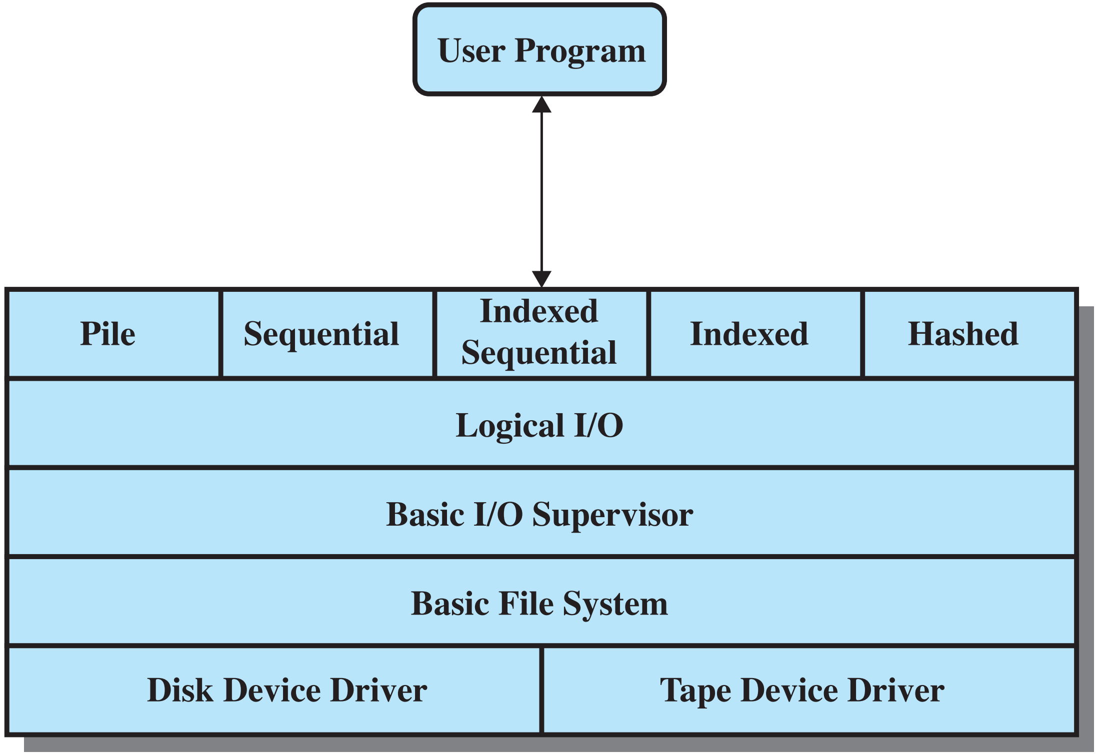
Minimal User Requirements

1. Should be able to create, delete, read, write and modify files
2. May have controlled access to other users' files
3. May control what type of accesses are allowed to the users' files
4. Should be able to move data between files
5. Should be able to back up and recover files in case of damage
6. Should be able to access his or her files by name rather than by numeric identifier

File System Layers

- Many file systems (FS) , sometimes many within an operating system
- Each FS with its own format
  - CD-ROM is ISO 9660
  - Unix has UFS, FFS
    Windows has FAT, FAT32, NTFS as well as floppy, CD, DVD Blu-ray
    Linux supports more than 40 types of FS, with extended file system ext2 and ext3 leading; plus distributed file systems
  - New ones still arriving
    ZFS, GoogleFS, Oracle ASM, FUSE

File System Software Architecture



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
- Considered to be part of the operating system

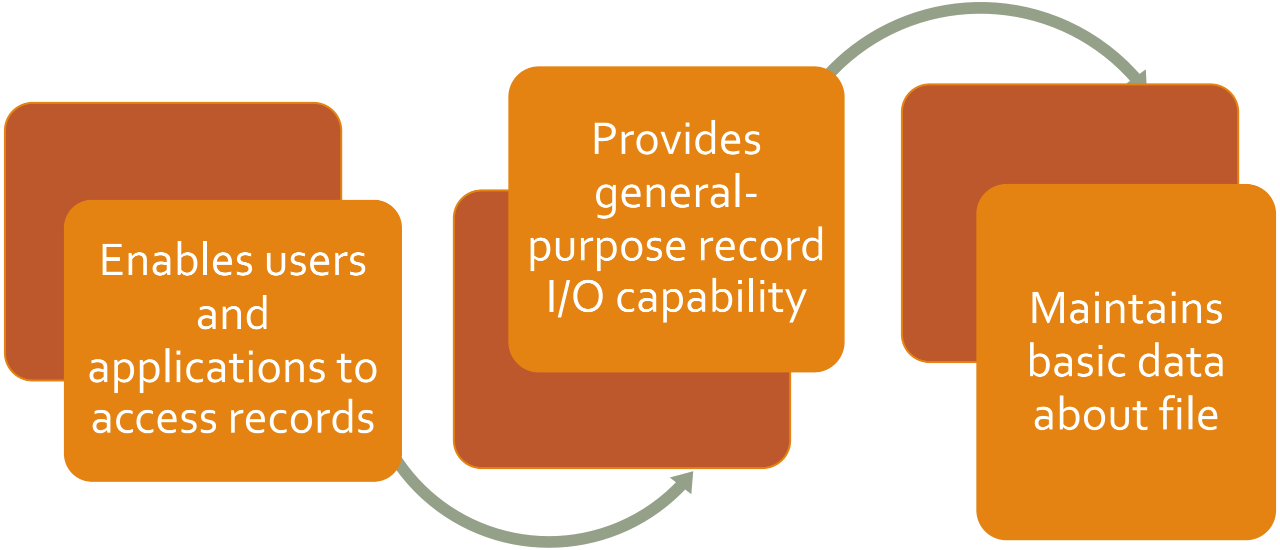
Basic File System

- Also referred to as the physical I/O level
- Primary interface with the environment outside the computer system
- Deals with blocks of data that are exchanged with disk or tape systems
- Concerned with the placement of blocks on the secondary storage device
- Concerned with buffering blocks in main memory
- Does not understand the content of the data or the structure of the files involved
- Considered part of the operating system

Basic I/O Supervisor

- Responsible for all file I/O initiation and termination
- At this level, control structures are maintained that deal with device I/O, scheduling, and file status
- Selects the device on which I/O is to be performed
- Concerned with scheduling disk and tape accesses to optimize performance
- I/O buffers are assigned and secondary memory is allocated at this level
- Part of the operating system

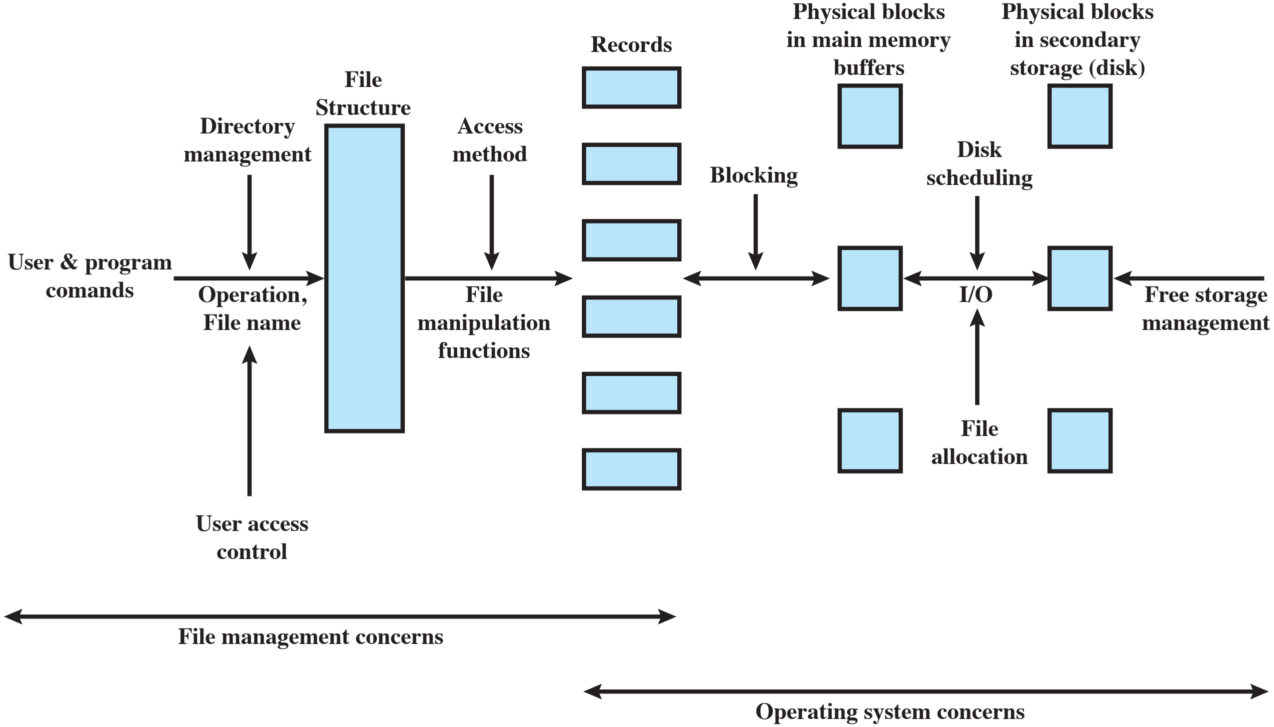
Logical I/O


Enables users and applications to access records
Provides general-purpose record I/O capability
Maintains basic data about file

Access Method

- Level of the file system closest to the user
- Provides a standard interface between applications and the file systems and devices that hold the data
- Different access methods reflect different file structures and different ways of accessing and processing the data

Elements of File Management



File Organization and Access

- File organization is the logical structuring of the records as determined by the way in which they are accessed
- In choosing a file organization, several criteria are important
  - Short access time
  - Ease of update
  - Economy of storage
  - Simple maintenance
  - Reliability
- Priority of criteria depends on the application that will use the file

File Sharing

Two issues arise when allowing files to be shared among a number of users:

- Access rights
- Management of simultaneous access

Access Rights

- None
  - The user would not be allowed to read the user directory that includes the file
- Knowledge
  - The user can determine that the file exists and who its owner is
- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- Appending
  - The user can add data to the file but cannot modify or delete any of the file's contents
- Updating
  - The user can modify, delete, and add to the file's data
- Changing protection
  - The user can change the access rights granted to other users
- Deletion
  - The user can delete the file from the file system

User Access Rights

Owner

- Usually the initial creator of the file
- Has full rights
- May grant rights to others

Specific Users

- Individual users who are designated by user ID

User Groups

- A set of users who are not individually defined

All

- All users who have access to this system
- These are public files

File allocation

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
- The approach taken for file allocation may influence the approach taken for free space management
- 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

Preallocation vs Dynamic Allocation

- A preallocation policy requires that the maximum size of a file be declared at the time of the file creation request
- For many applications it is difficult to estimate reliably the maximum potential size of the file
  - Tends to be wasteful because users and application programmers tend to overestimate size
  - Dynamic allocation allocates space to a file in portions as needed

Portion Size

- In choosing a portion size there is a trade-off between efficiency from the point of view of a single file versus overall system efficiency
- Items to be considered
  - Contiguity of space increases performance, especially for Retrieve_Next operations, and greatly for transactions running in a transaction-oriented operating system
  - Having a large number of small portions increases the size of tables needed to manage the allocation information
  - Having fixed-size portions simplifies the reallocation of space
  - Having variable-size or small fixed-size portions minimizes waste of unused storage due to overallocation

Alternatives

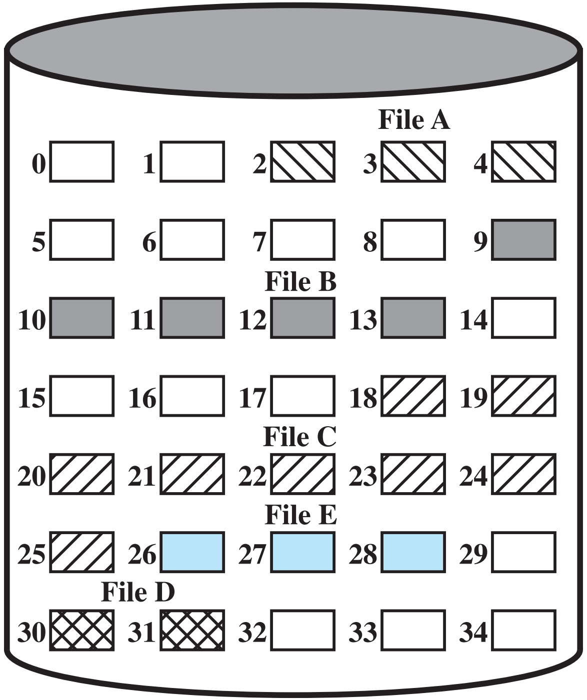
Variable, large contiguous portions

- Provides better performance
- The variable size avoids waste
- The file allocation tables are small

Blocks

- Small fixed portions provide greater flexibility
- They may require large tables or complex structures for their allocation
- Contiguity has been abandoned as a primary goal
- Blocks are allocated as needed

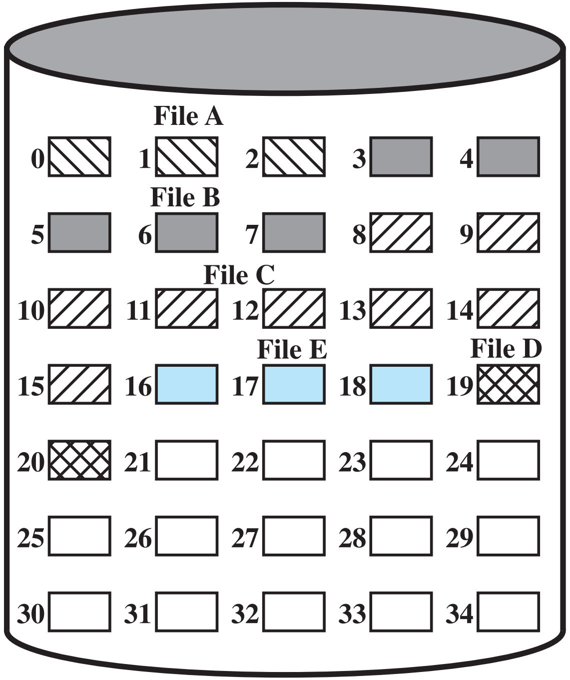
Contiguous File Allocation



File Allocation Table

File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3

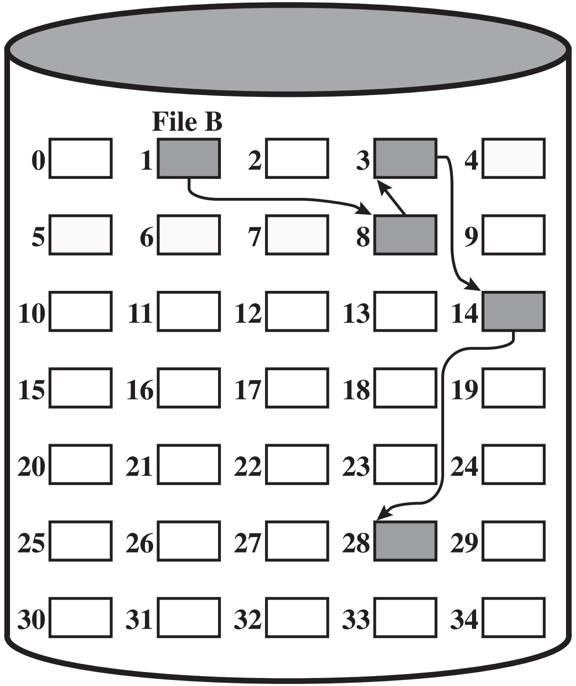
Contiguous File Allocation (after compaction)



File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

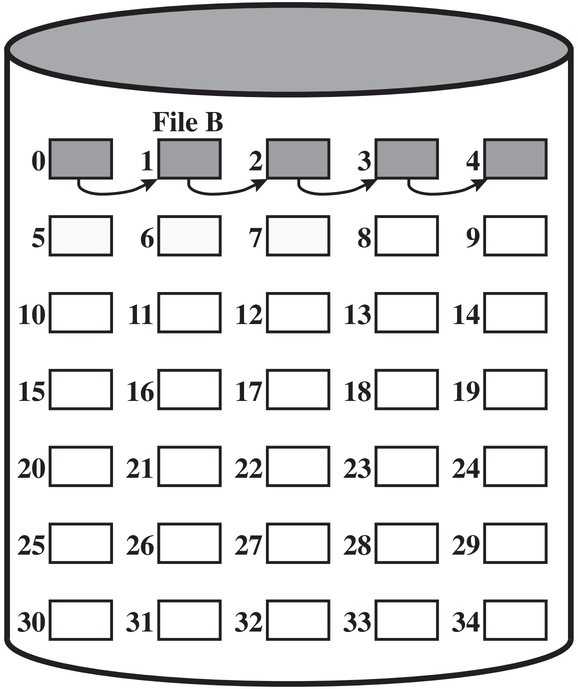
Chained Allocation



File Allocation Table

File Name	Start Block	Length
• • •	• • •	• • •
File B	1	5
• • •	• • •	• • •

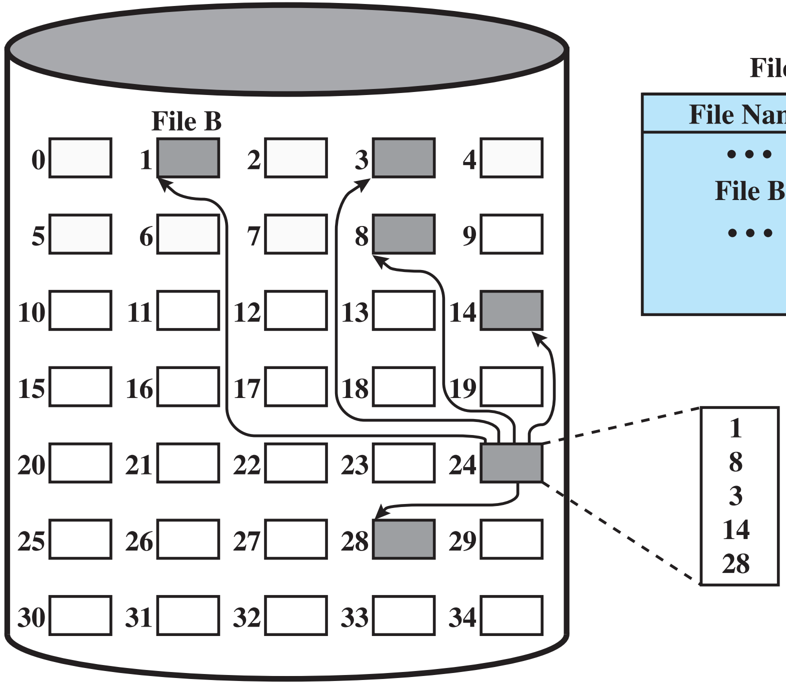
Chained Allocation (after consolidation)



File Allocation Table

File Name	Start Block	Length
• • •	• • •	• • •
File B	0	5
• • •	• • •	• • •

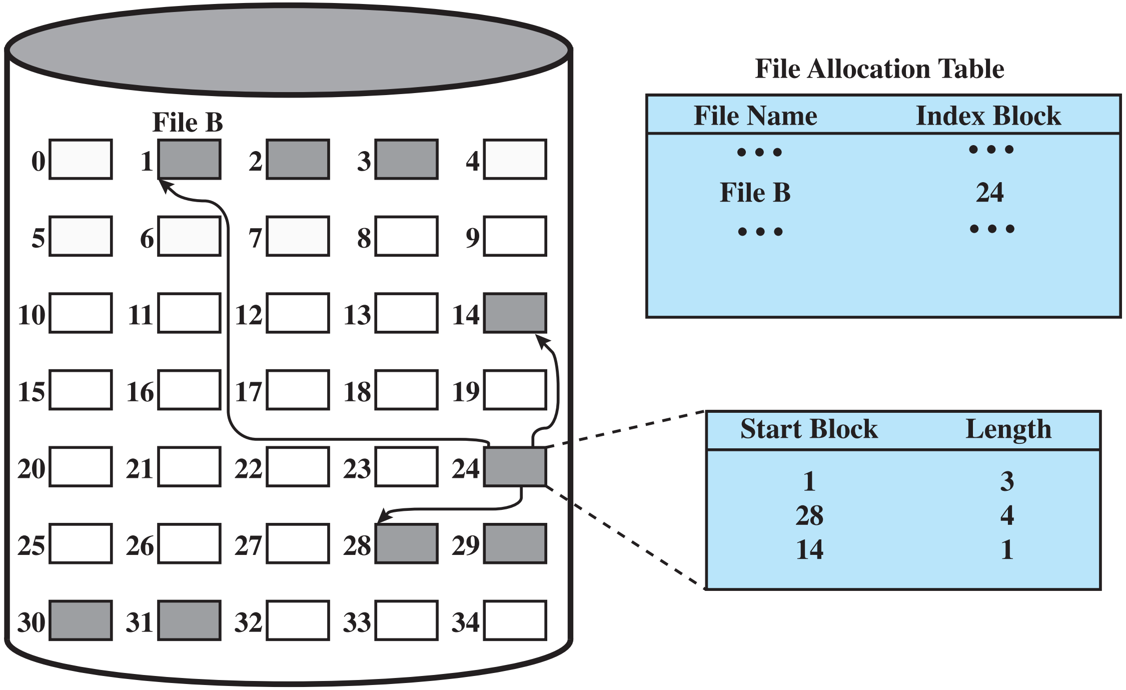
Indexed Allocation with Block Portions



File Allocation Table

File Name	Index Block
• • •	• • •
File B	24
• • •	• • •

Indexed Allocation with Variable-Length Portions



Summary on File Allocatin Methods

	Contiguous	Chained	Indexed	
Preallocation?	Necessary	Possible	Possible	
Fixed or variable size portions?	Variable	Fixed blocks	Fixed blocks	Variable
Portion size	Large	Small	Small	Medium
Allocation frequency	Once	Low to high	High	Low
Time to allocate	Medium	Long	Short	Medium
File allocation table size	One entry	One entry	Large	Medium

Free space management

Free Space Management

- To perform file allocation, it is necessary to know which blocks are available
- A disk allocation table is needed in addition to a file allocation table

Bit Tables

- This method uses a vector containing one bit for each block on the disk
- Each entry of a 0 corresponds to a free block, and each 1 corresponds to a block in use

Advantages

- Works well with any file allocation method
- It is as small as possible

Chained Free Portions

- The free portions may be chained together by using a pointer and length value in each free portion
- Negligible space overhead because there is no need for a disk allocation table
- Suited to all file allocation methods

Disadvantages:

- Leads to fragmentation
- Every time you allocate a block you need to read the block first to recover the pointer to the new first free block before writing data to that block

Indexing

Treats free space as a file and uses an index table as it would for file allocation

For efficiency, the index should be **on the basis of variable-size portions** rather than blocks

This approach provides efficient support for **all of the file allocation methods**

Free Block List

Each block is assigned a number sequentially

The list of the numbers of all free blocks is maintained in a reserved portion of the disk

Depending on the size of the disk, either 24 or 32 bits will be needed to store a single block number

The size of the free block list is 24 or 32 times the size of the corresponding bit table

Two effective techniques for storing a small part of the free block list in main memory:

The list can be treated either **as a push-down stack** or **as a FIFO queue** with a few thousand elements in main memory

Volumes

- A collection of addressable sectors in secondary memory that an OS or application can use for data storage
- The sectors in a volume need not be consecutive on a physical storage device
  - They need only appear that way to the OS or application
- A volume may be the result of assembling and merging smaller volumes

File management examples

UNIX File Management

- In the UNIX file system, six types of files are distinguished:

Regular, or ordinary
Contains arbitrary data in zero or more data blocks

Directory
Contains a list of file names plus pointers to associated inodes (index nodes)

Special
Contains no data but provides a mechanism to map physical devices to file names

Named pipes
An interprocess communications facility

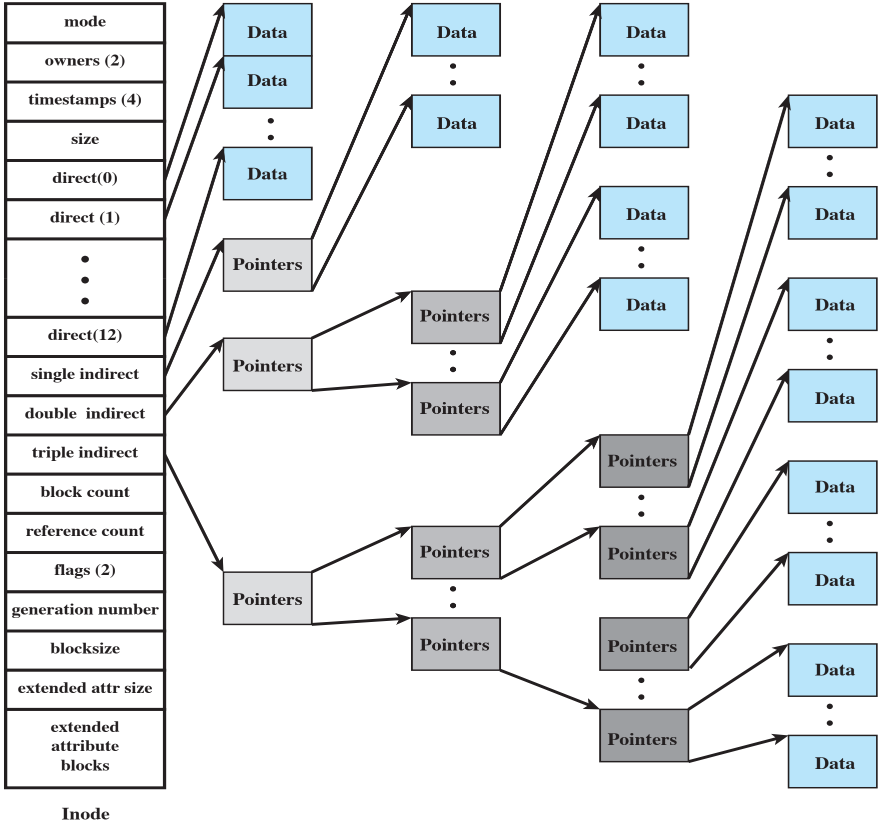
Links
An alternative file name for an existing file

Symbolic links
A data file that contains the name of the file to which it is linked

Inodes

- All types of UNIX files are administered by the OS by means of inodes
- An inode (index node) is a control structure that contains the key information needed by the operating system for a particular file
- Several file names may be associated with a single inode
  - An active inode is associated with exactly one file
  - Each file is controlled by exactly one inode

FreeBSD inode and Files



File Allocation in UNIX

- File allocation is done on a block basis
- Allocation is dynamic, as needed, rather than using preallocation
- An indexed method is used to keep track of each file, with part of the index stored in the inode for the file
- In all UNIX implementations the inode includes a number of direct pointers and three indirect pointers (single, double, triple)

Capacity of a FreeBSD file with 4KB block size

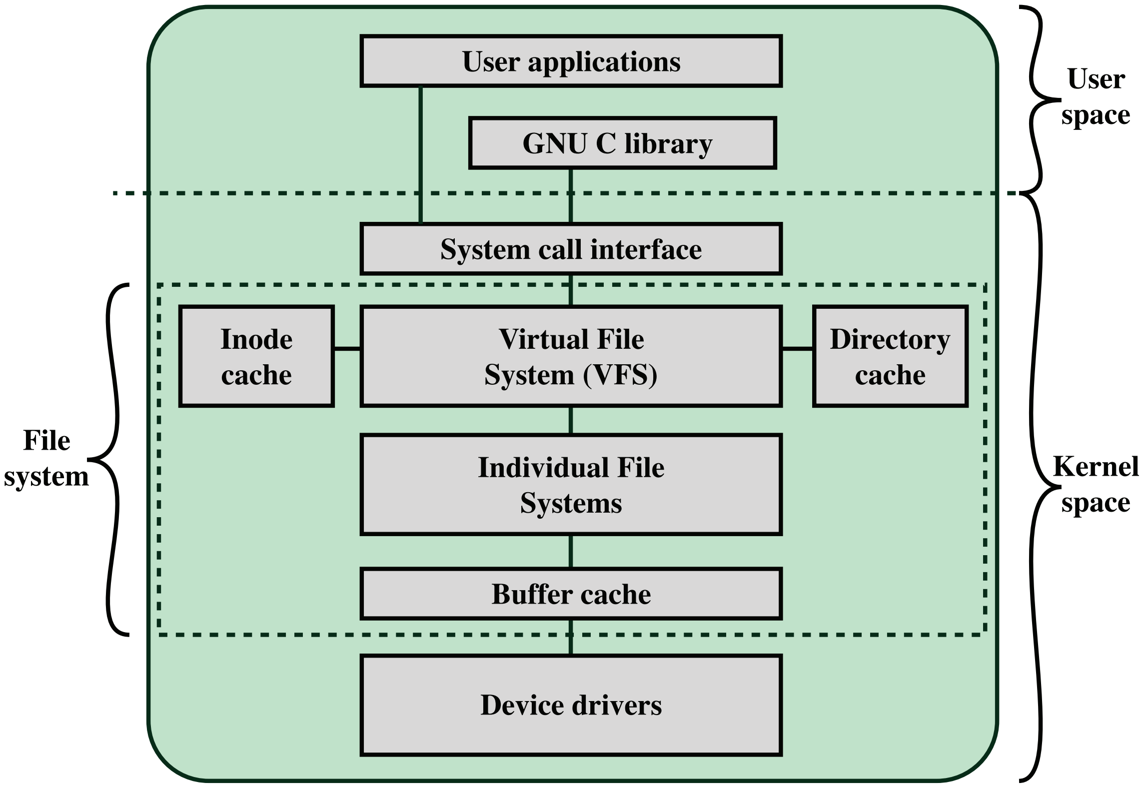
Level	Number of Blocks	Number of Bytes
Direct	12	48K
Single Indirect	512	2M
Double Indirect	512 x 512 = 256K	1G
Triple Indirect	512 x 256K = 128M	512G

Volume Structure

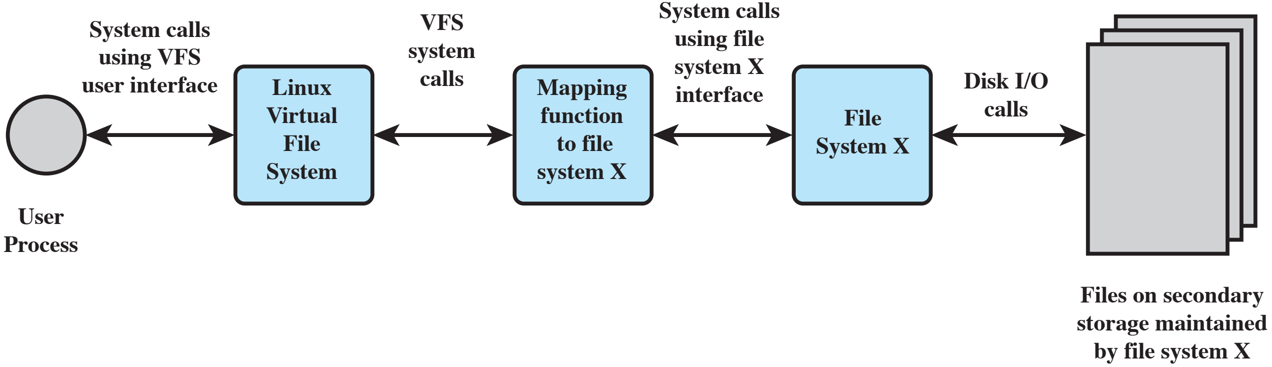
A UNIX file system resides on a single logical disk or disk partition and is laid out with the following elements

- **Boot block**: Contains code required to boot the operating system
- **Superblock**: Contains attributes and information about the file system
- **Inode table**: Collection of inodes for each file
- **Data blocks**: Storage space available for data files and subdirectories

Linux Virtual File System



Linux Virtual File System



Caches

- Inode cache
- Directory cache
- Buffer cache
  - The buffer cache is independent of the file systems and is integrated into the mechanisms that the Linux kernel uses to allocate and read and write data buffers
  - As the real file systems read data from the underlying physical disks, this results in requests to the block device drivers to read physical blocks from the device that they control

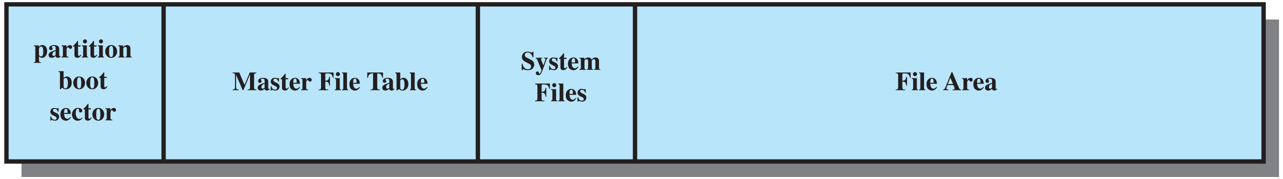
Windows File System

- The developers of Windows NT designed a new file system, the New Technology File System (NTFS) to meet high-end requirements for workstations and servers
- Key features of NTFS
  - Recoverability
  - Security
  - Large disks and large files
  - Multiple data streams
  - Journaling
  - Compression and encryption
  - Hard and symbolic links

NTFS Volume and File Structure

- **Sector**
  - The smallest physical storage unit on the disk
  - The data size in bytes is a power of 2 (almost always 512B)
- **Cluster**
  - One or more contiguous sectors
  - The cluster size in sectors is a power of 2
- **Volume**
  - A logical partition on a disk, consisting of one or more clusters and used by a file system to allocate space
  - Can be all or a portion of a single disk or it can extend across multiple disks
  - The maximum volume size for NTFS is 264 clusters

NTFS Volume Layout



Master File Table (MFT)

- It is the core of the Windows file system
- The MFT is organized as a table of 1,024-byte rows, called records
- Each row describes a file on this volume, including the MFT itself, which is treated as a file
- Each record in the MFT consists of a set of attributes that serve to define the file (or folder) characteristics and the file contents

NTFS Components