File Directories

- Contains information about files
  - Attributes
  - Location
  - Ownership
- Directory itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

Simple Structure for a Directory

- List of entries, one for each file
- Sequential file with the name of the file serving as the key
- Provides no help in organizing the files
- Forces user to be careful not to use the same name for two different files

Table 12.2 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

	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

Usage Information	
Date Created	When file was first placed in directory
Identity of Creator	Usually but not necessarily the current owner
Date Last Read Access	Date of the last time a record was read
Identity of Last Reader	User who did the reading
Date Last Modified	Date of the last update, insertion, or deletion
Identity of Last Modifier	User who did the modifying
Date of Last Backup	Date of the last time the file was backed up on another storage medium
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, and whether the file has been updated in main memory but not yet on disk

Two-level Scheme for a Directory

- One directory for each user and a master directory
- Master directory contains entry for each user
  - Provides address and access control information
- Each user directory is a simple list of files for that user
- Still provides no help in structuring collections of files

Hierarchical, or Tree-Structured Directory

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries

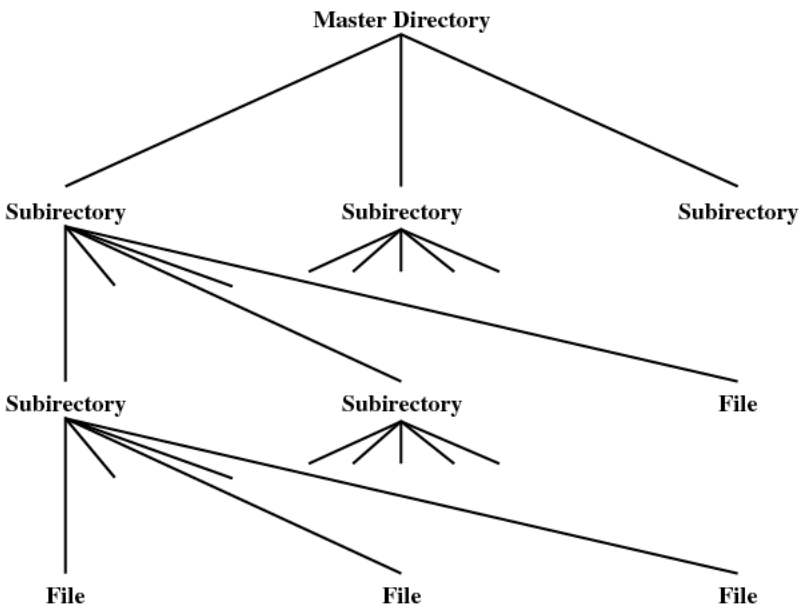


Figure 12.4 Tree-Structured Directory

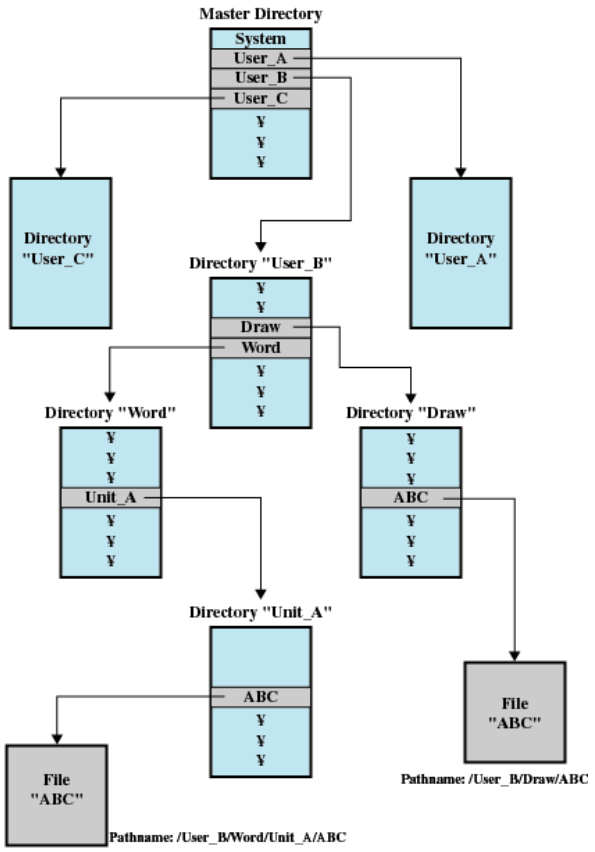


Figure 12.5 Example of Tree-Structured Directory

Hierarchical, or Tree-Structured Directory

- Files can be located by following a path from the root, or master, directory down various branches
  - This is the pathname for the file
- Can have several files with the same file name as long as they have unique path names

Hierarchical, or Tree-Structured Directory

- Current directory is the working directory
- Files are referenced relative to the working directory

File Sharing

- In multiuser system, allow files to be shared among users
- Two issues
  - Access rights
  - Management of simultaneous access

Access Rights

- None
  - User may not know of the existence of the file
  - User is not allowed to read the user directory that includes the file
- Knowledge
  - User can only determine that the file exists and who its owner is

Access Rights

- 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

Access Rights

- Updating
  - The user can modify, deleted, and add to the file's data. This includes creating the file, rewriting it, and removing all or part of the data
- Changing protection
  - User can change access rights granted to other users
- Deletion
  - User can delete the file

Access Rights

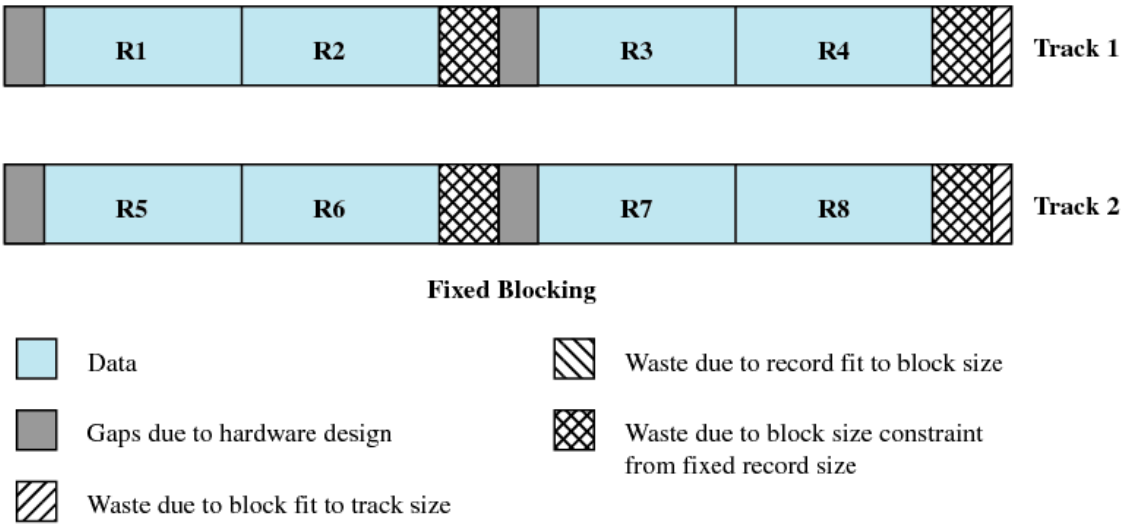
- Owners
  - Has all rights previously listed
  - May grant rights to others using the following classes of users
    - Specific user
    - User groups
    - All for public files

Simultaneous Access

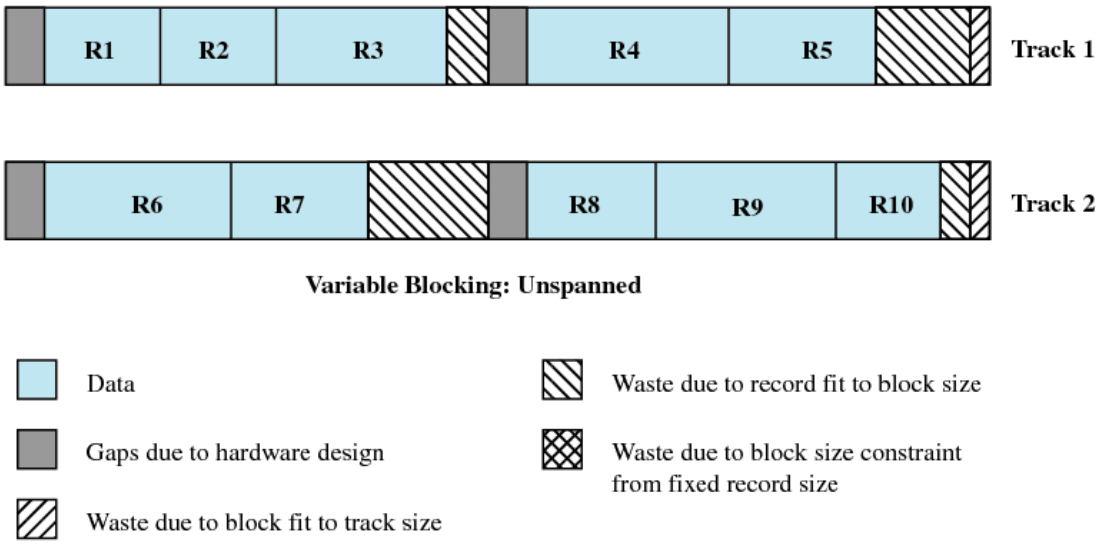
- User may lock entire file when it is to be updated
- User may lock the individual records during the update
- Mutual exclusion and deadlock are issues for shared access

Fixed Blocking

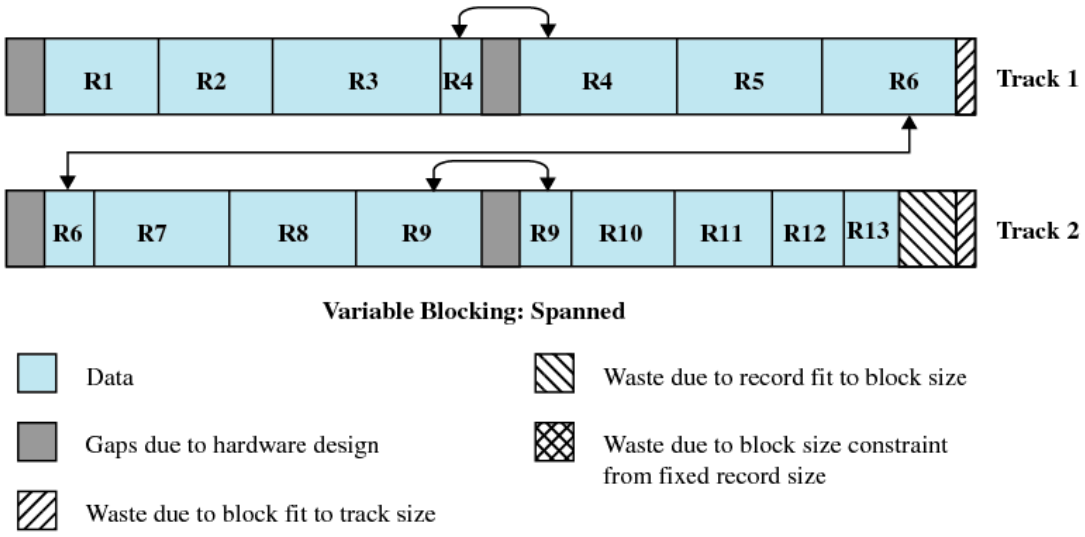
Units of I/O with secondary storage are called **blocks**



Variable Blocking Unspanned



Variable Blocking: Spanned



Secondary Storage Management

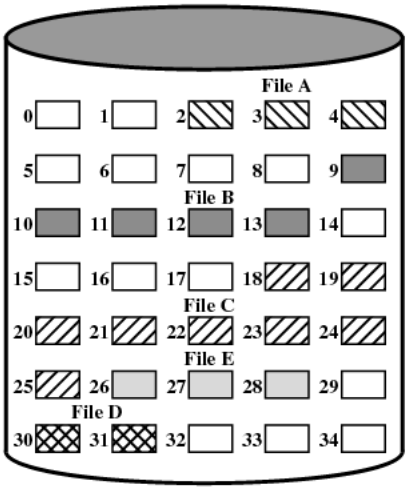
- Space must be allocated to files
- Must keep track of the space available for allocation

Preallocation

- Need the maximum size for the file at the time of creation
  - Difficult to reliably estimate the maximum potential size of the file
  - Tend to overestimated file size so as not to run out of space

Methods of File Allocation

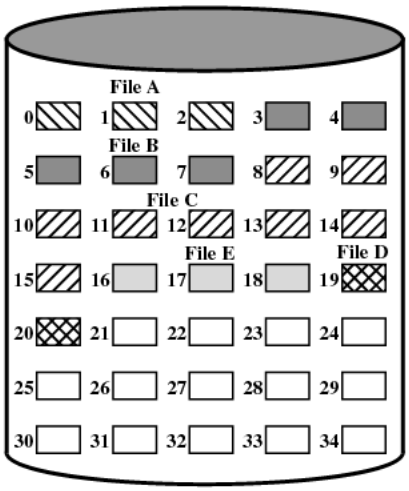
- Contiguous allocation
  - Single set of blocks is allocated to a file at the time of creation
  - Only a single entry in the file allocation table
    - Starting block and length of the file
- External fragmentation will occur
  - Need to perform compaction



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

Figure 12.7 Contiguous File Allocation



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

Figure 12.8 Contiguous File Allocation (After Compaction)

Methods of File Allocation

- Chained allocation
  - Allocation on individual block basis
  - Each block contains a pointer to the next block in the chain
  - Only single entry in the file allocation table
    - Starting block and length of file
- No external fragmentation
- Best for sequential files
- No accommodation of the principle of locality

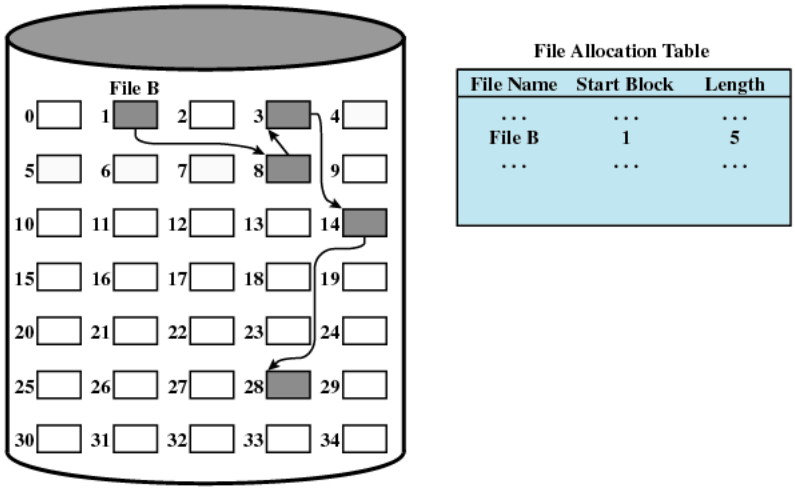


Figure 12.9 Chained Allocation

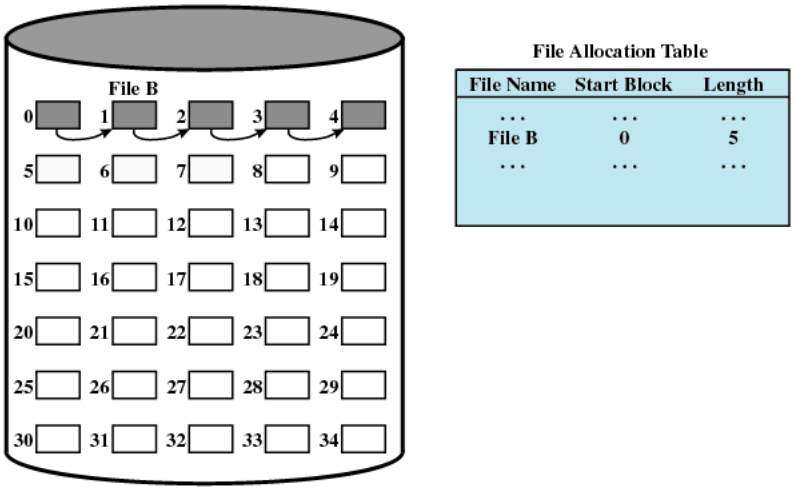


Figure 12.10 Chained Allocation (After Consolidation)

Methods of File Allocation

- Indexed allocation
  - File allocation table contains a separate one-level index for each file
    - The index has one entry for each portion allocated to the file
    - The file allocation table contains block number for the index

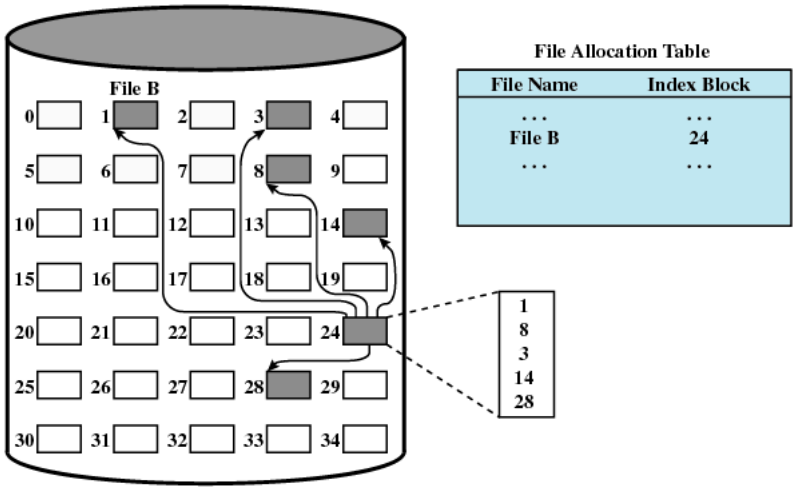


Figure 12.11 Indexed Allocation with Block Portions

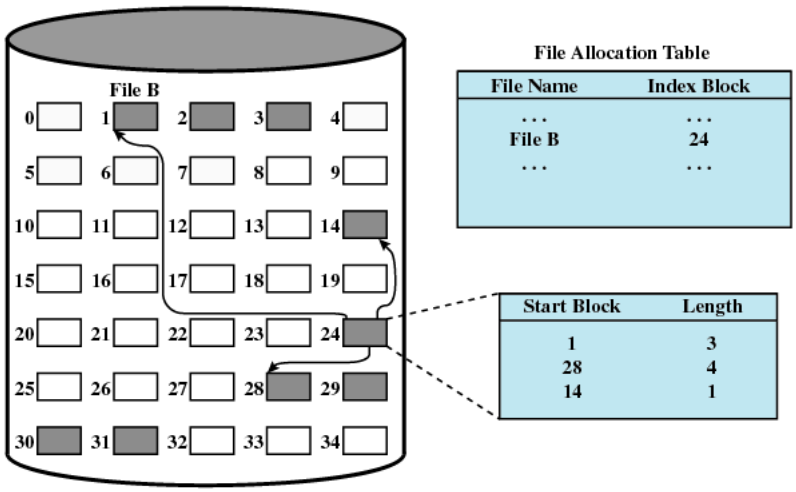


Figure 12.12 Indexed Allocation with Variable-Length Portions

UNIX File Management

- Types of files
  - Regular, or ordinary
  - Directory
  - Special
  - Named pipes
  - Links
  - Symbolic links

Inodes

- Index node
- Control structure that contains key information for a particular file

Table 12.4 Information in a UNIX Disk-Resident Inode

File Mode	16-bit flag that stores access and execution permissions associated with the file. 12-14 File type (regular, directory, character or block special, FIFO pipe 9-11 Execution flags 8 Owner read permission 7 Owner write permission 6 Owner execute permission 5 Group read permission 4 Group write permission 3 Group execute permission 2 Other read permission 1 Other write permission 0 Other execute permission
Link Count	Number of directory references to this inode
Owner ID	Individual owner of file
Group ID	Group owner associated with this file
File Size	Number of bytes in file
File Addresses	39 bytes of address information
Last Accessed	Time of last file access
Last Modified	Time of last file modification
Inode Modified	Time of last inode modification

File Allocation

- Allocation on block basis
- Dynamic allocation (not preallocation)
- Index method used to keep track of each file
  - part of which is stored in file's inode

File Allocation

- inode contains 39 bytes address
  - thirteen 3-byte addresses (pointers)
    - first 10 addresses point to first 10 data blocks
  - if file is longer than that (i.e. 10 blocks)
    - 11th address points to next portion of index
      - single indirect block
  - file is larger than that
    - 12th address points double indirect block
      - i.e. contains list of addresses of single indirect blocks, each of which contains pointers to file blocks
    - 13th address points to triple indirect block

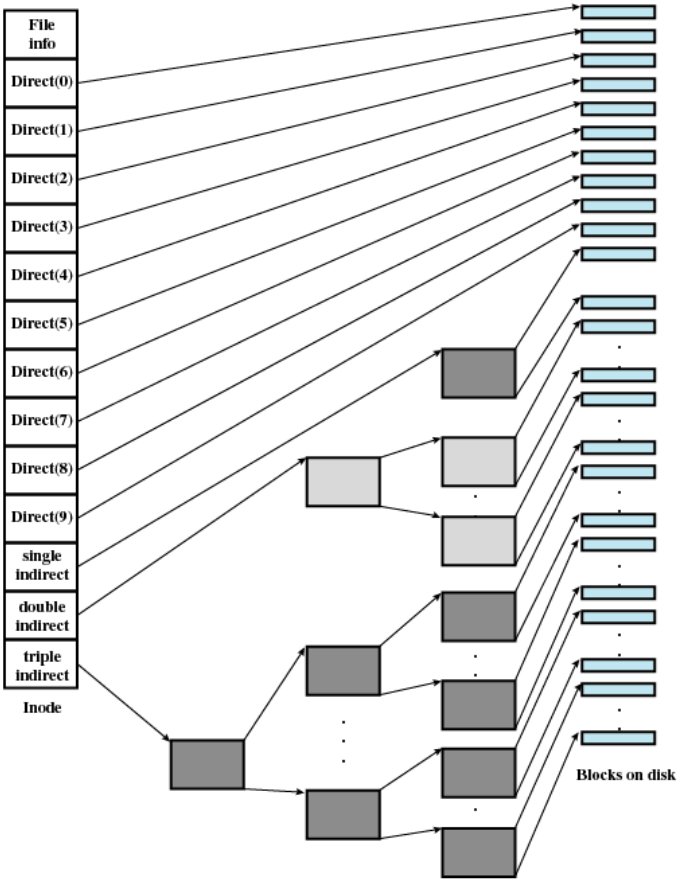


Figure 12.13 Layout of a UNIX File on Disk

Directories

- Hierarchical tree
  - root
  - subdirectories
  - directory is simply a file that contains list of file names and a pointer to their inodes
    - these entries are called *dentry* (directory entry)
    - note that these files can themselves be directories

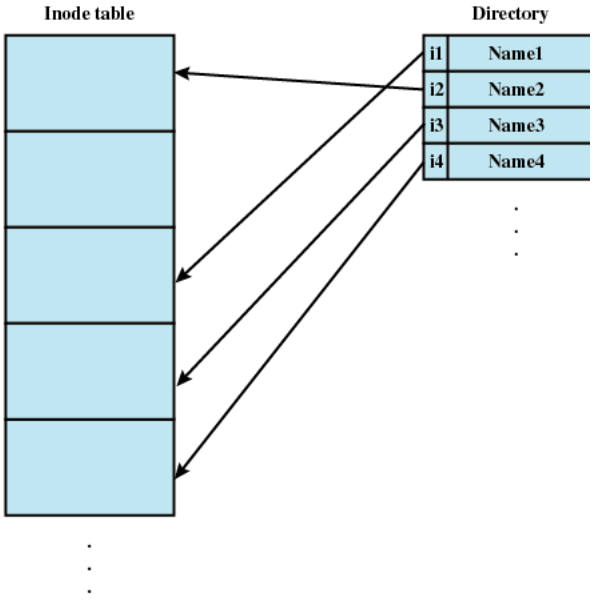


Figure 12.14 UNIX Directories and Inodes

Linux Virtual File System

- Uniform file system interface to user processes
- Represents any conceivable file system's general feature and behavior
- Assumes files are objects that share basic properties regardless of the target file system

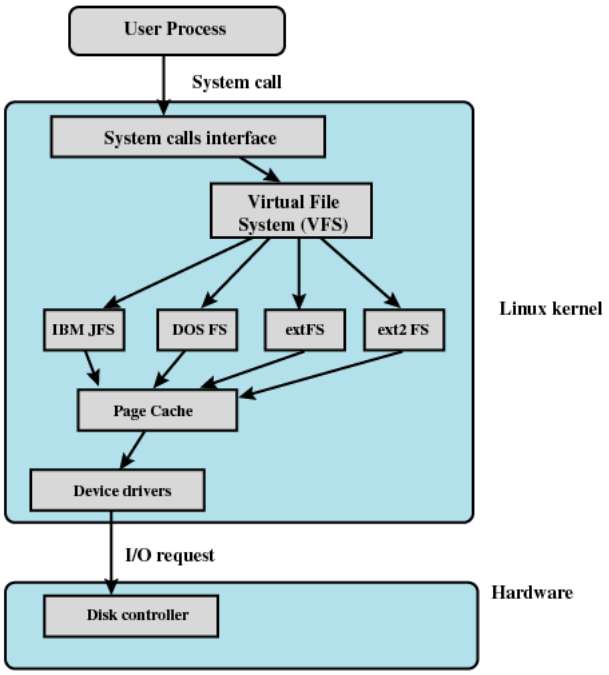


Figure 12.15 Linux Virtual File System Context

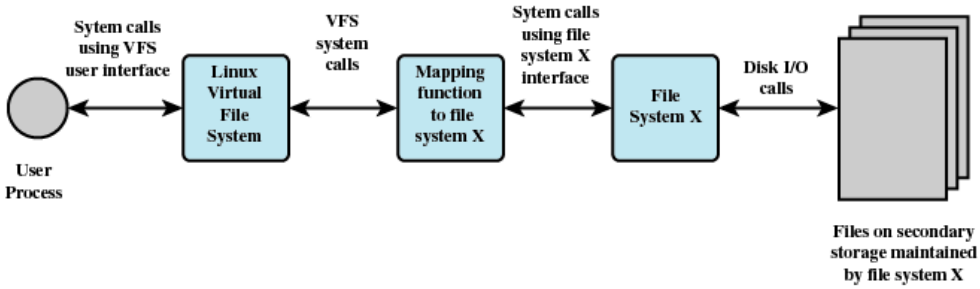


Figure 12.16 Linux Virtual File System Concept

Primary Objects in VFS

- Superblock object
  - Represents a specific mounted file system
- Inode object
  - Represents a specific file
- Dentry object
  - Represents a specific directory entry
- File object
  - Represents an open file associated with a process

Linux File Systems

- ext2
  - 2nd extended file systems
  - introduced in 1993
  - has journaling feature
  - file size up to 2TB
  - file system size up to 32TB

Windows File System

- Key features of NTFS
  - Recoverability
  - Security
  - Large disks and large files
  - Multiple data streams
  - General indexing facility

NTFS Volume and File Structure

- Sector
  - The smallest physical storage unit on the disk
- Cluster
  - One or more contiguous sectors
- Volume
  - Logical partition on a disk

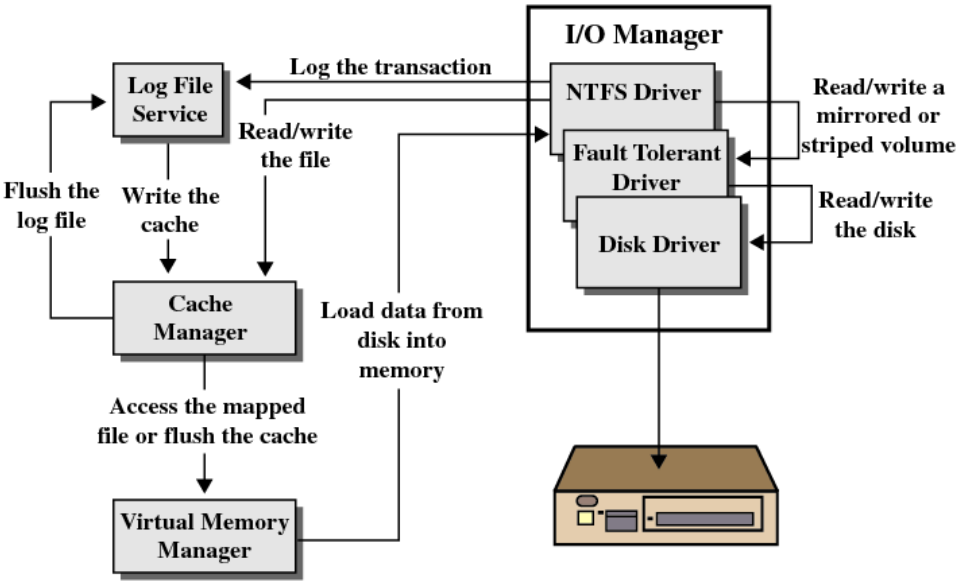


Figure 12.18 Windows NTFS Components