Griffith UNIVERSITY

2803ICT

File System Management

based on
William Stallings
Chapter 12

TOPICS

1) Introduction and overview
2) Advanced C programming
3) Memory management
4) File system and device IO
5) Socket programming
6) Processes
7) Interprocess communication (IPC)
8) Multithreading
9) Synchronisation (*2 weeks*)
10) Distributed software

Files

- Data collections created by users
- 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 organised into hierarchical or more complex structure to reflect the relationships among files

File System

- Provide a means to store data organised 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 Management System Objectives

Griffith UNIVERSITY

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

Typical Software Organisation

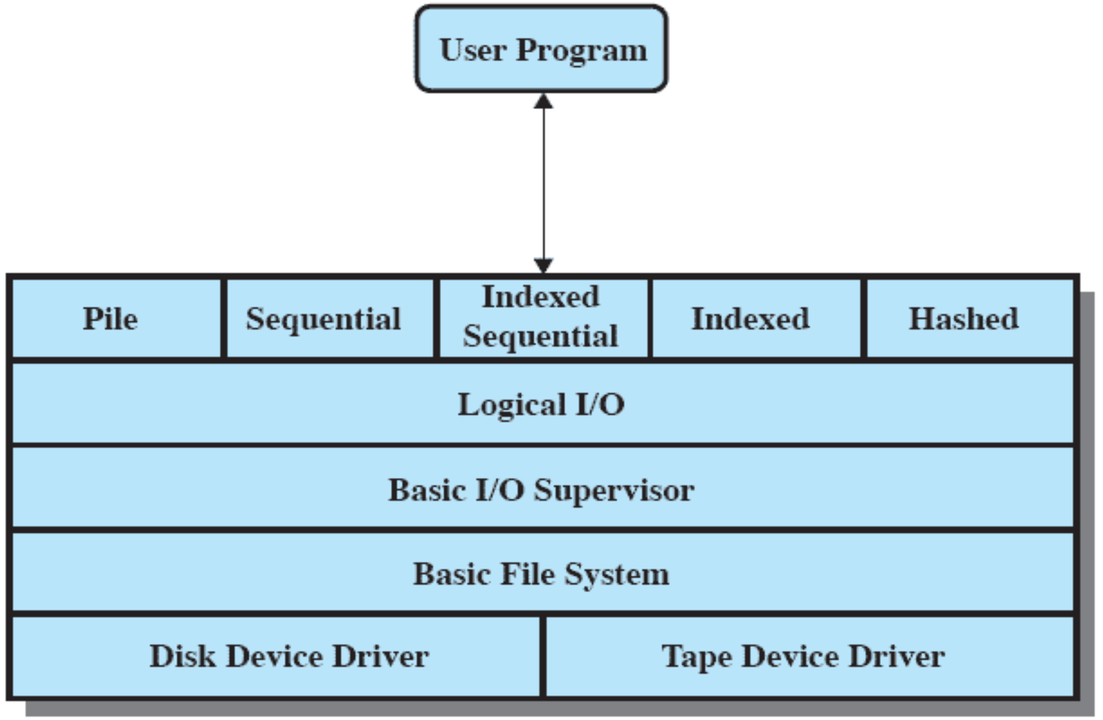
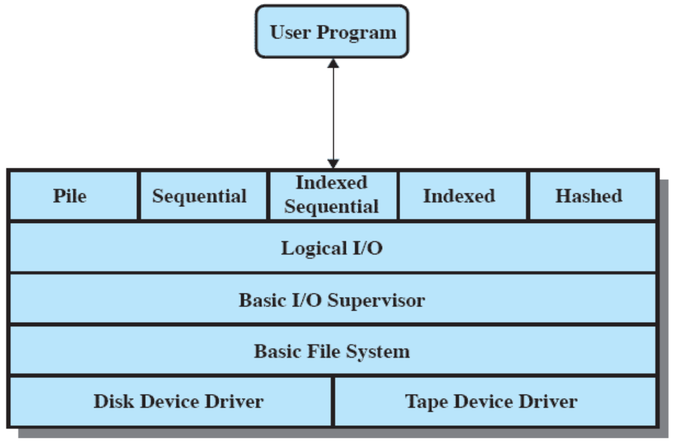


Figure 12.1 File System Software Architecture

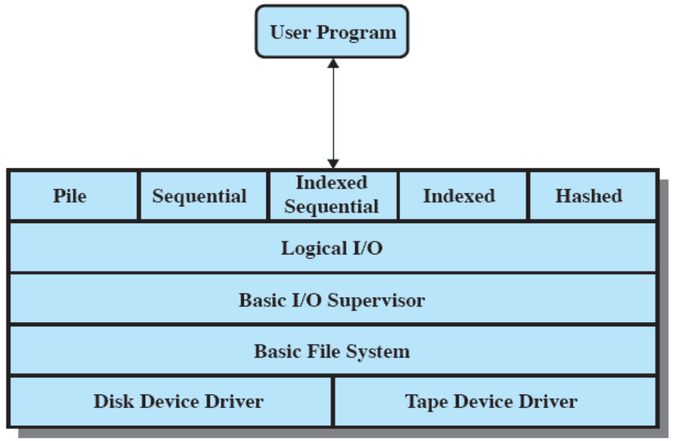
Device Drivers

- 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



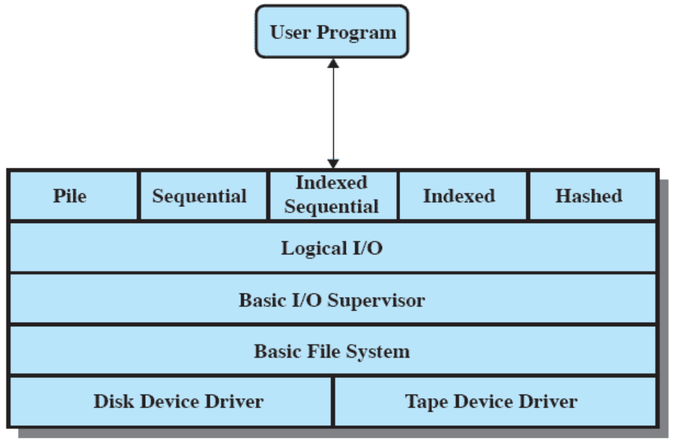
Basic File System

- 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
- Considered part of the operating system



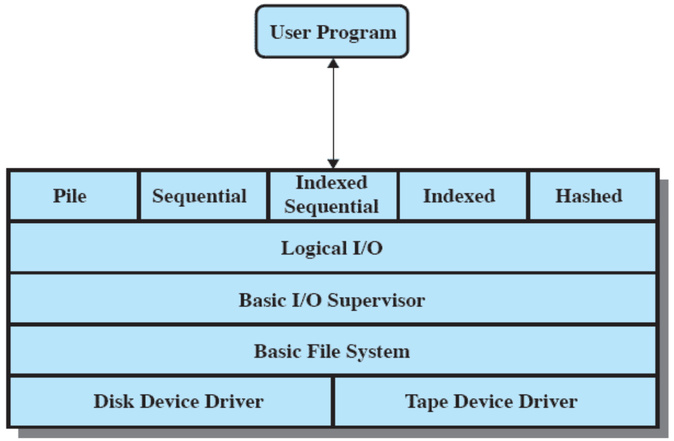
Basic I/O Supervisor

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

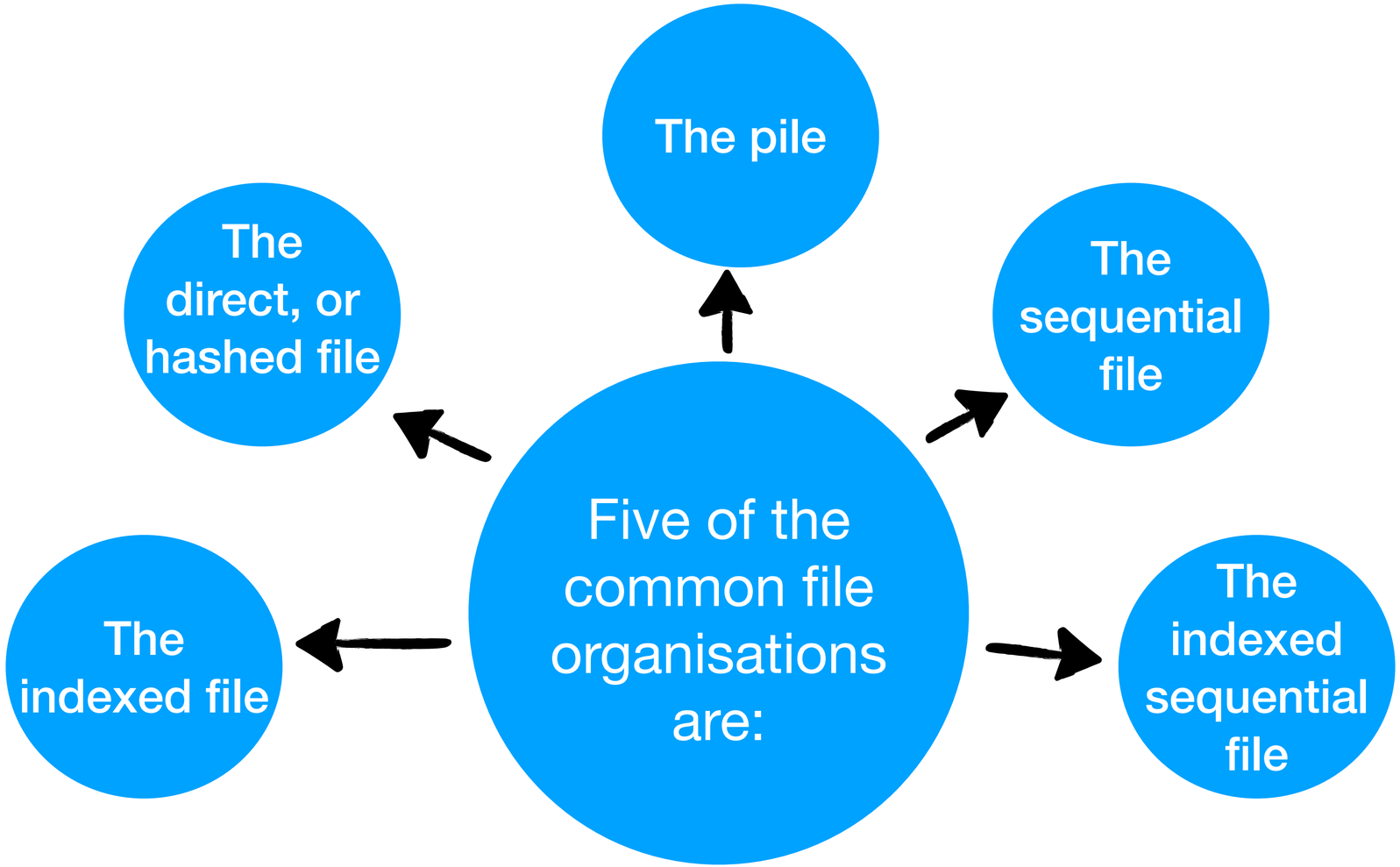


File Organisation and Access

- File organisation is the **logical structuring** of the records as determined by the way in which they are accessed
- In choosing a file organisation, 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

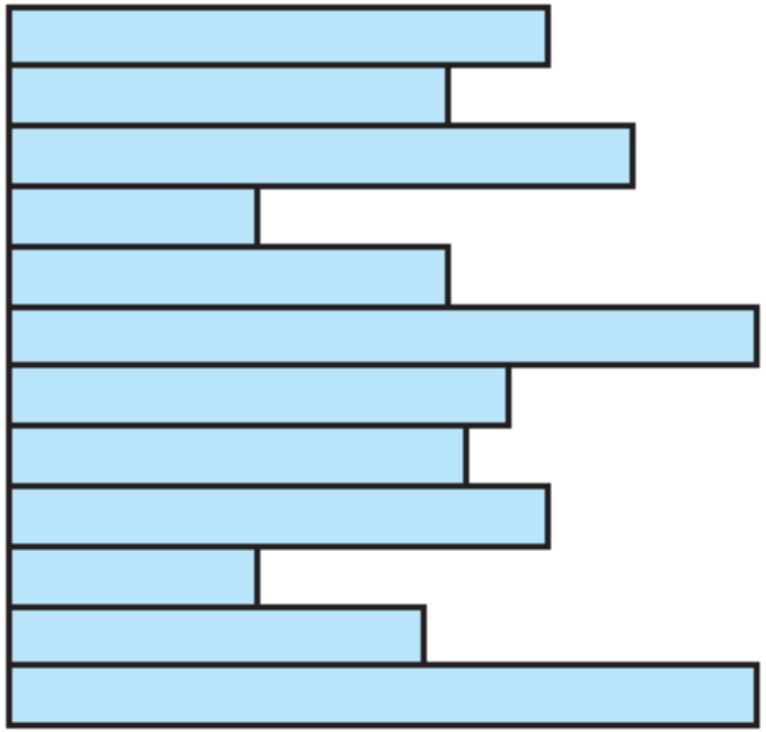


File Organisation Types



The Pile

- **Least complicated** form of file organisation
- Data are collected in the order they arrive
- Each record consists of **one burst of data**
- Purpose is simply to accumulate the mass of data and save it
- Record access is by **exhaustive search**

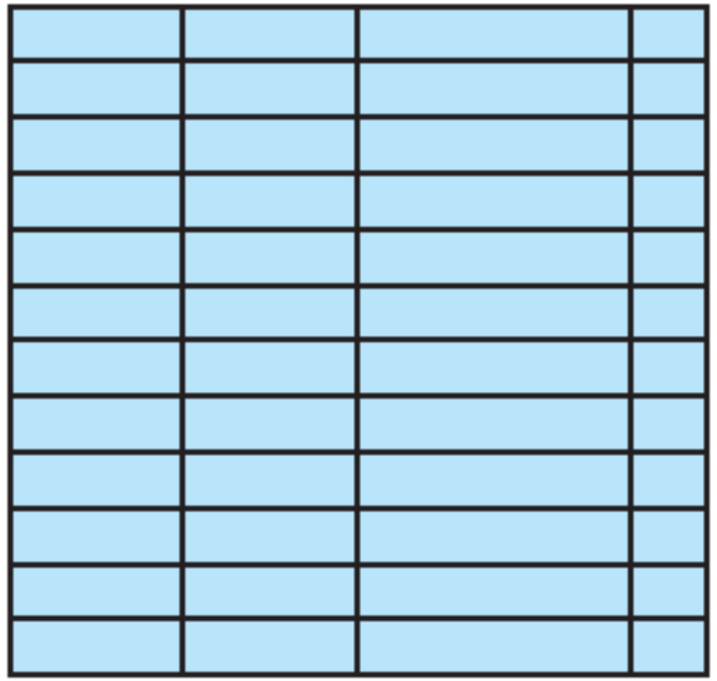


Variable-length records
Variable set of fields
Chronological order

(a) Pile File

The Sequential File

- **Most common** form of file structure
- A **fixed format** is used for records
- **Key field** uniquely identifies the record
- Typically used in batch applications
- Only organisation that is easily stored on tape as well as disk

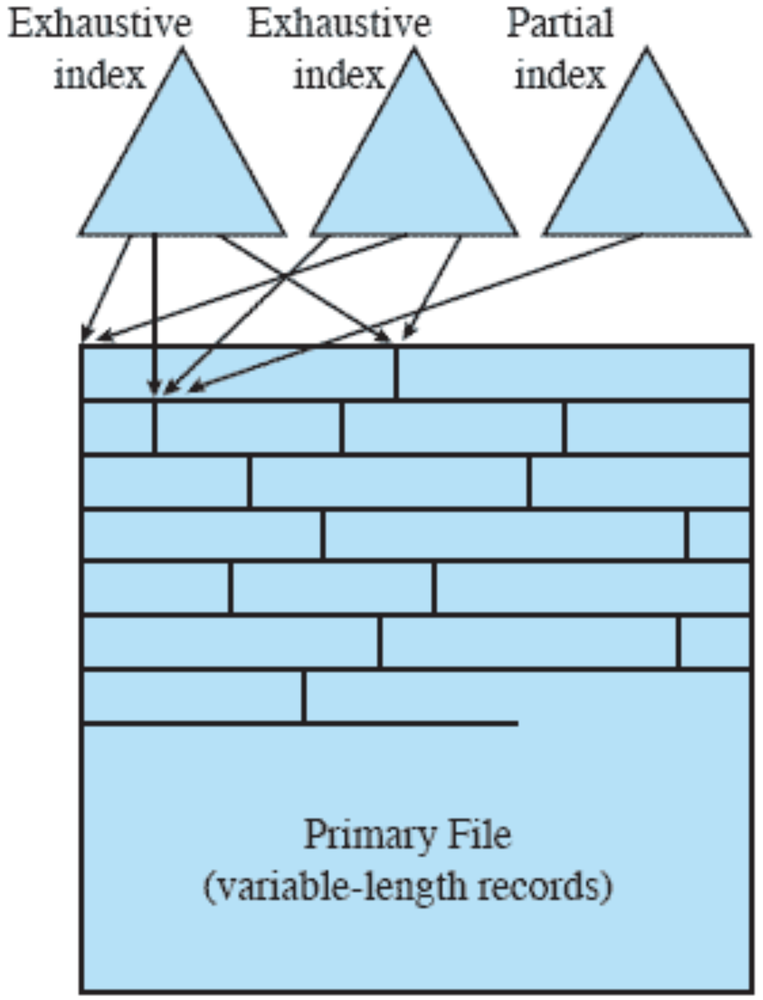


Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

Indexed File

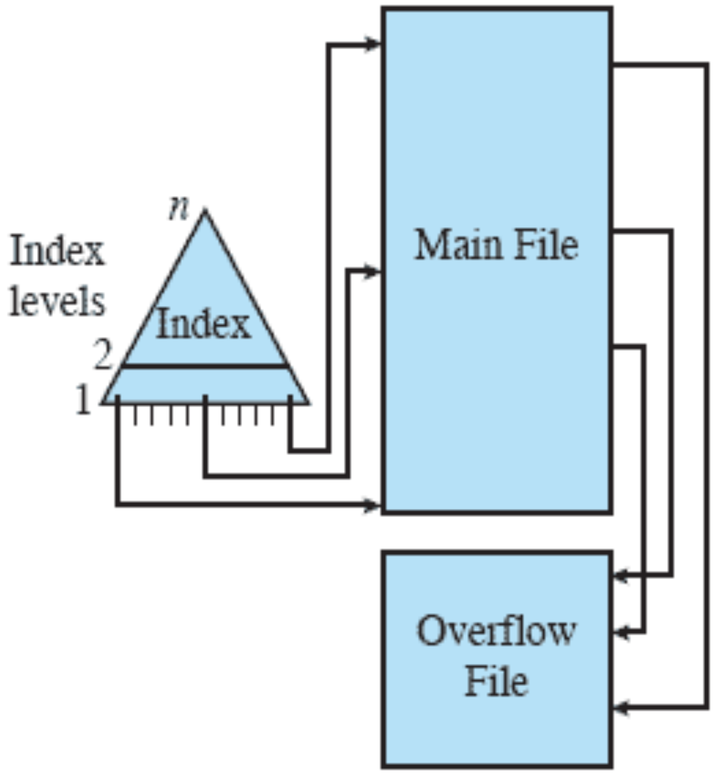
- **Records are accessed only through their indexes**
- **Variable-length records** can be employed
- Exhaustive index contains one entry for every record in the main file
- Partial index contains entries to records where the field of interest exists
- Used mostly in applications where timeliness of information is critical
- Examples would be airline reservation systems and inventory control systems



(d) Indexed File

Indexed Sequential File

- Adds an index to the file to **support random access**
- Adds an overflow file
- Greatly reduces the time required to access a single record
- Multiple levels of indexing can be used to provide greater efficiency in access



(c) Indexed Sequential File

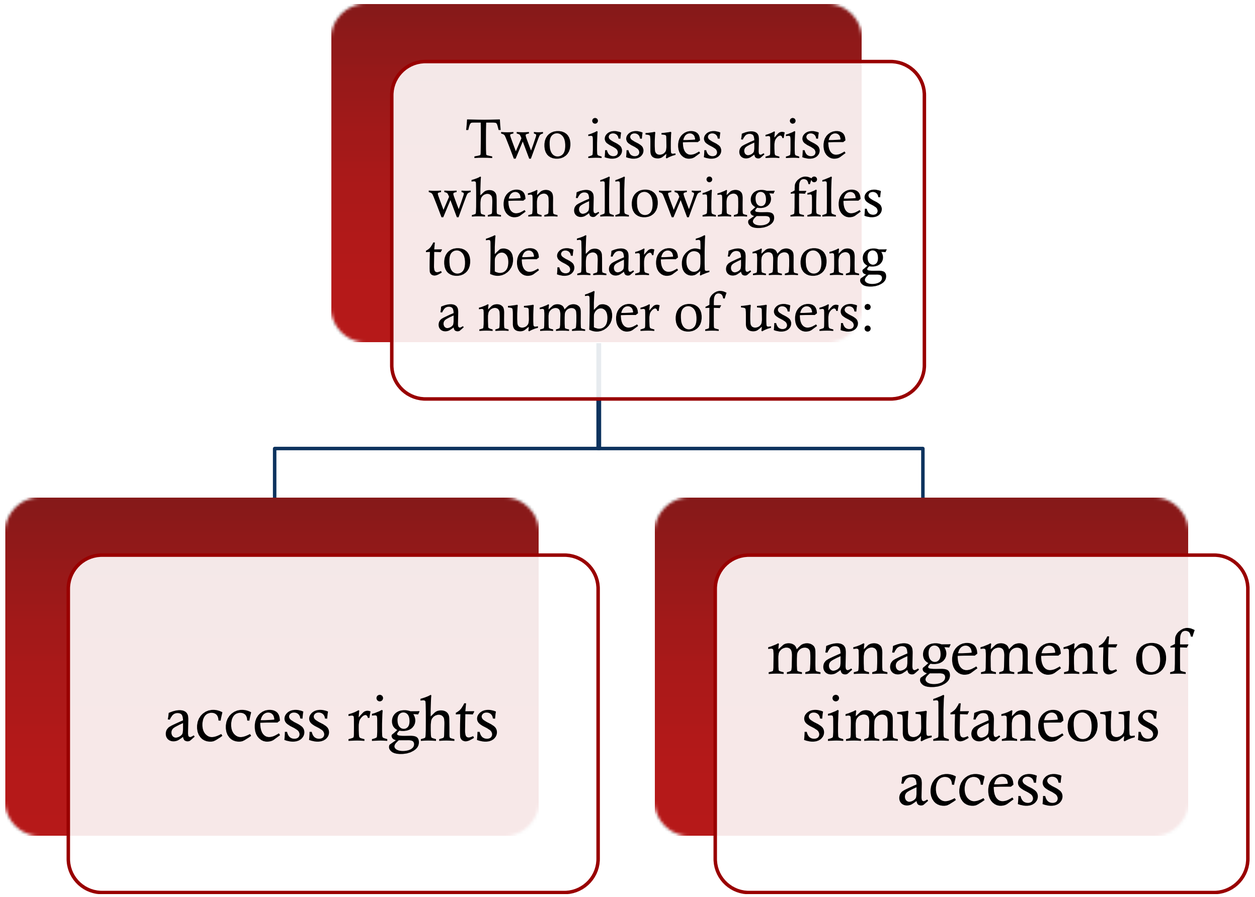
Direct or Hashed File

- Access directly any block of a known address
- **Makes use of hashing on the key value**
- Often used where:
    - very rapid access is required
    - fixed-length records are used
    - records are always accessed one at a time
- Examples are: directories, pricing tables, schedules, name lists

Operations Performed on a Directory

- To understand the requirements for a file structure, it is helpful to consider the types of operations that may be performed on the directory:

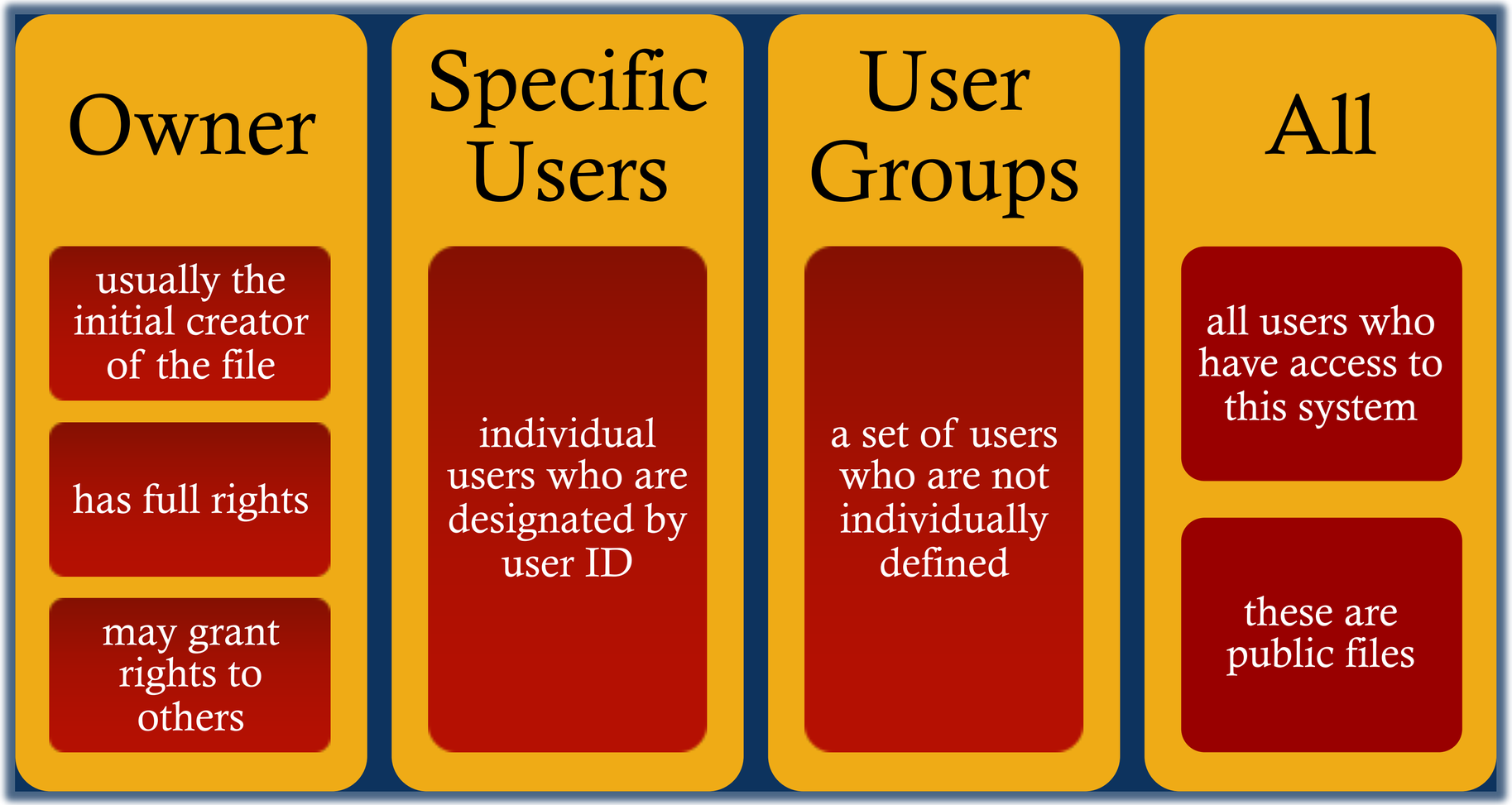
File Sharing



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 and can then petition the owner for additional 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
- *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

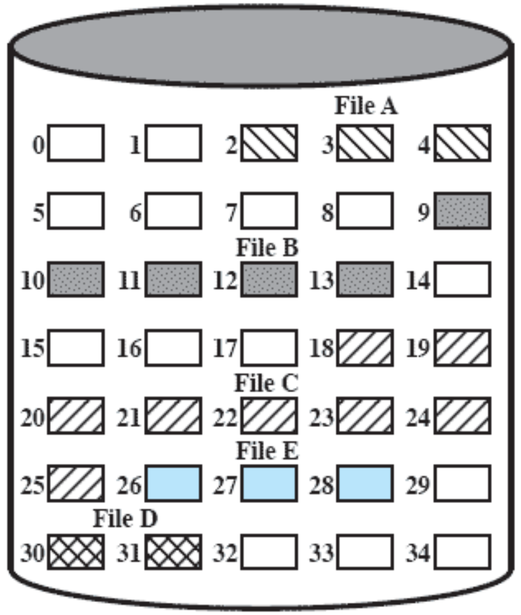


File Allocation

- 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

Contiguous File Allocation

- **A single contiguous set of blocks** is allocated to a file at the time of file creation
- Preallocation strategy using variable-size portions
- Is the best from the point of view of the individual sequential file

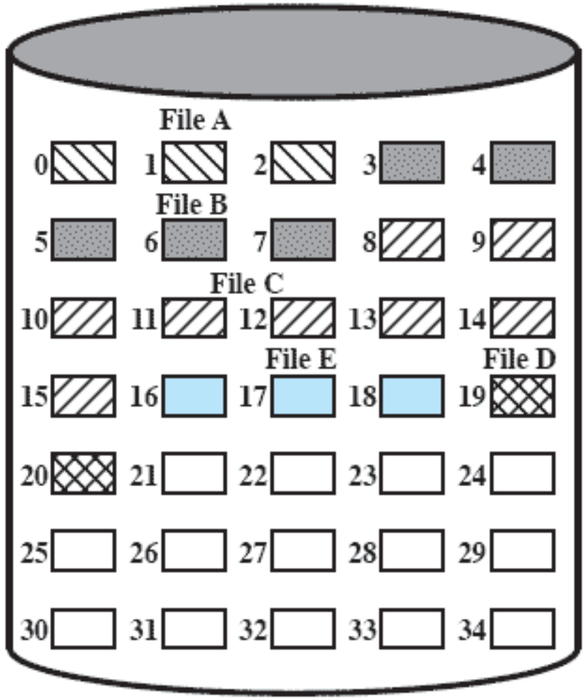


File Allocation Table

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

After Compaction



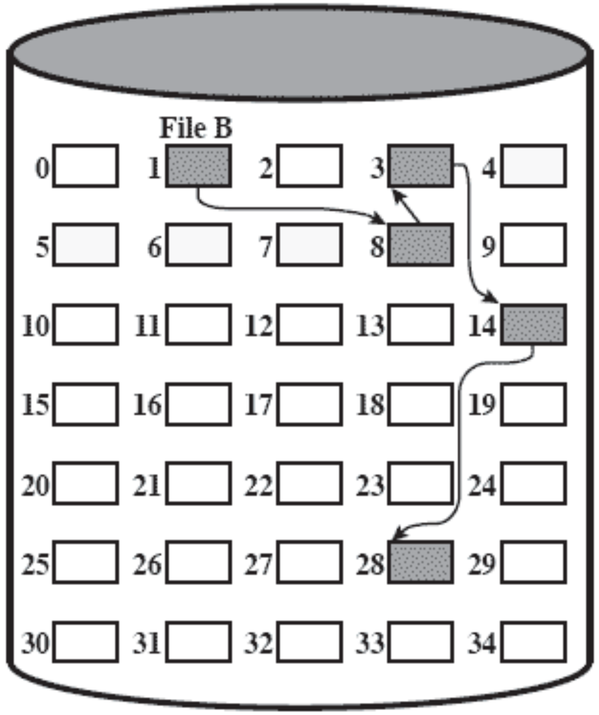
File Allocation Table

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

Figure 12.8 Contiguous File Allocation (After Compaction)

Chained Allocation

- **Allocation is on an individual block basis**
- Each block contains a pointer to the next block in the chain
- The file allocation table needs just a single entry for each file
- No external fragmentation to worry about
- Best for sequential files



File Allocation Table

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

Figure 12.9 Chained Allocation

Chained Allocation After Consolidation

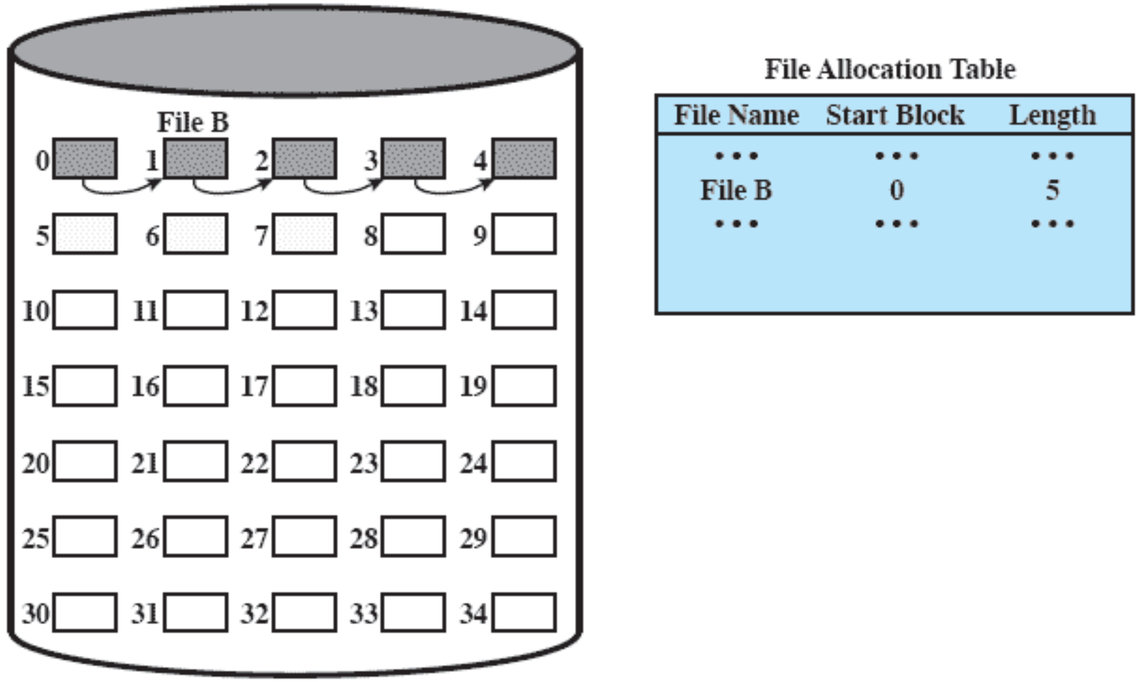


Figure 12.10 Chained Allocation (After Consolidation)

Indexed Allocation with Block Portions

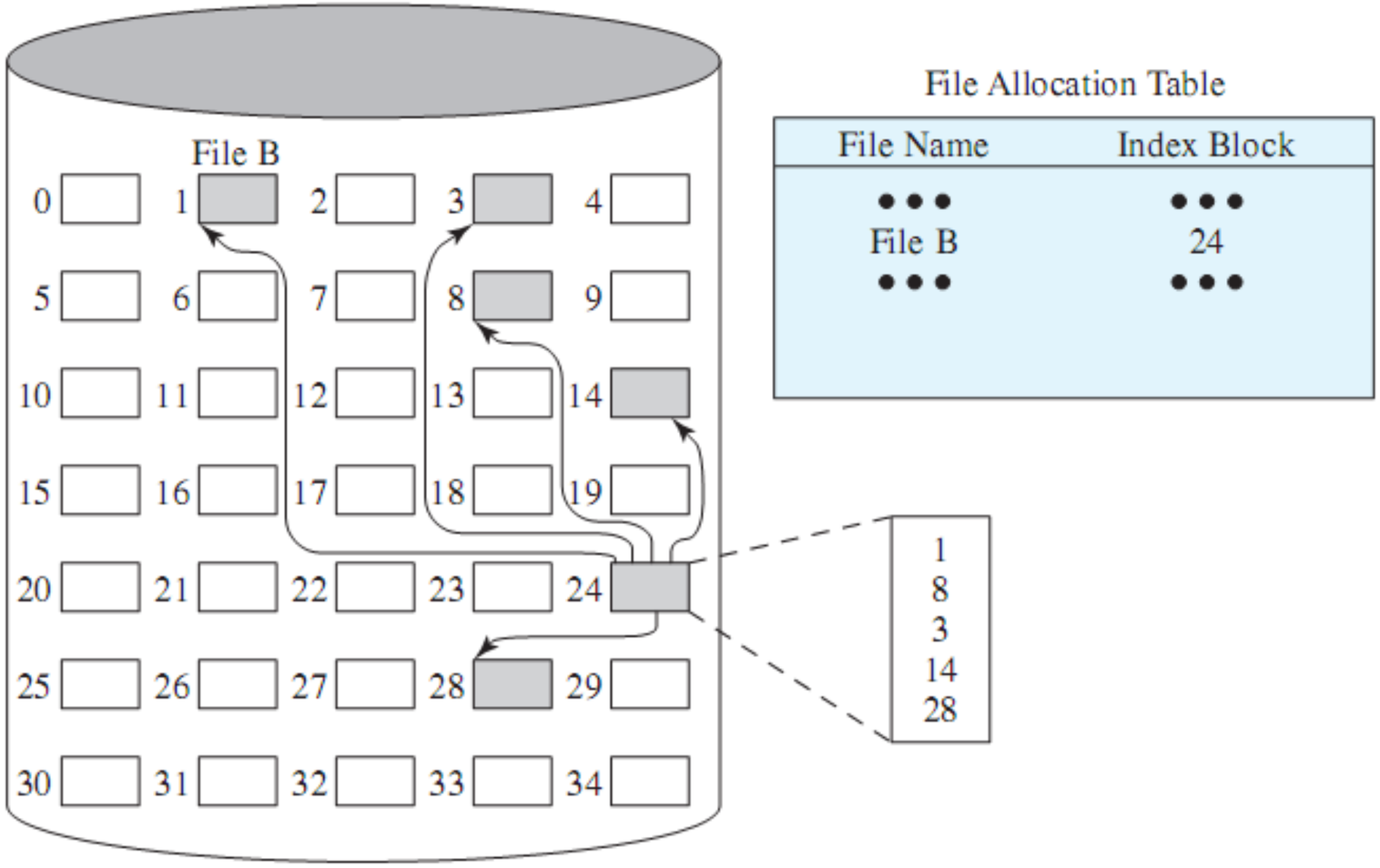


Figure 12.11 Indexed Allocation with Block Portions

Indexed Allocation with Variable Length Portions

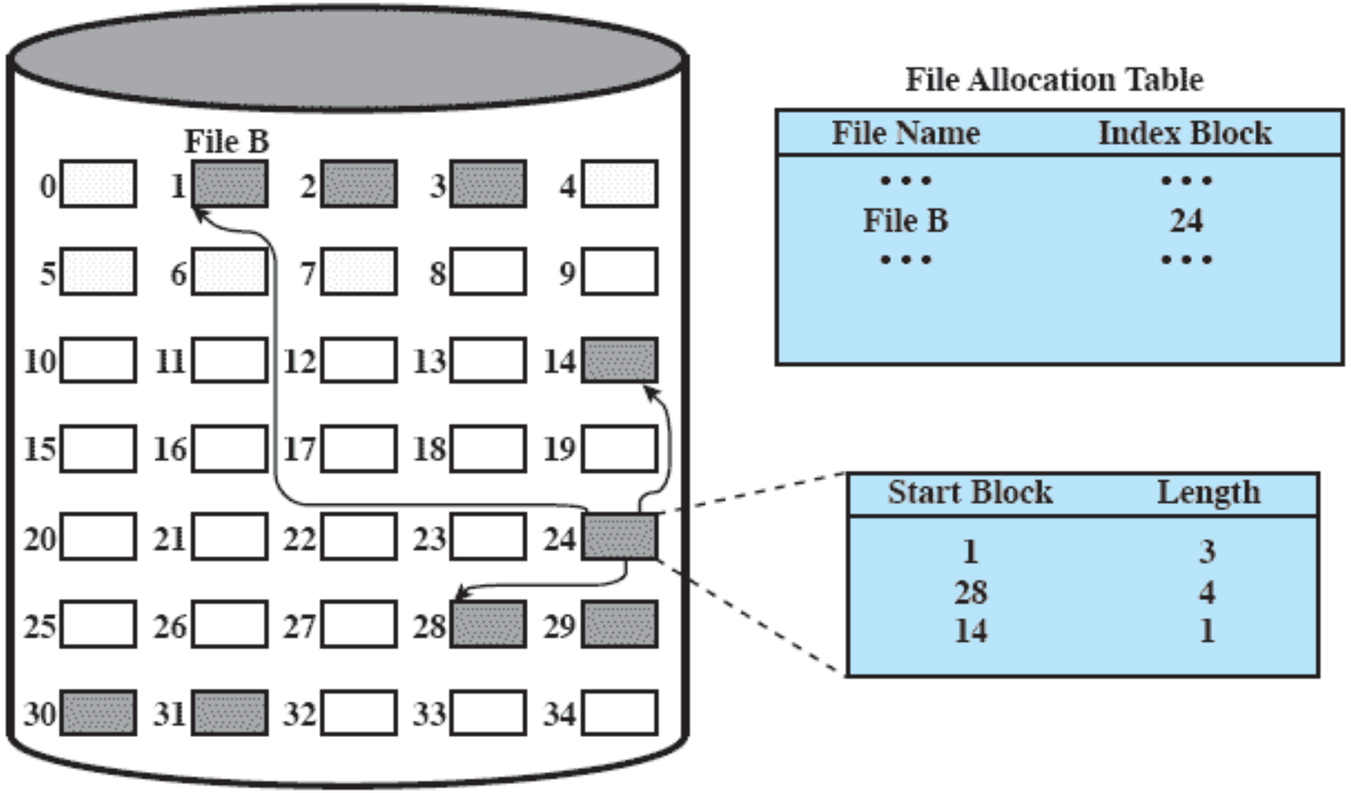


Figure 12.12 Indexed Allocation with Variable-Length Portions

Free Space Management

- Just as allocated space must be managed, so must the unallocated space
- 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 and must be stored on disk

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

- the list can be treated as a push-down stack with the first few thousand elements of the stack kept in main memory
- the list can be treated as a FIFO queue, with a few thousand entries from both the head and the tail of the queue 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

Access Matrix

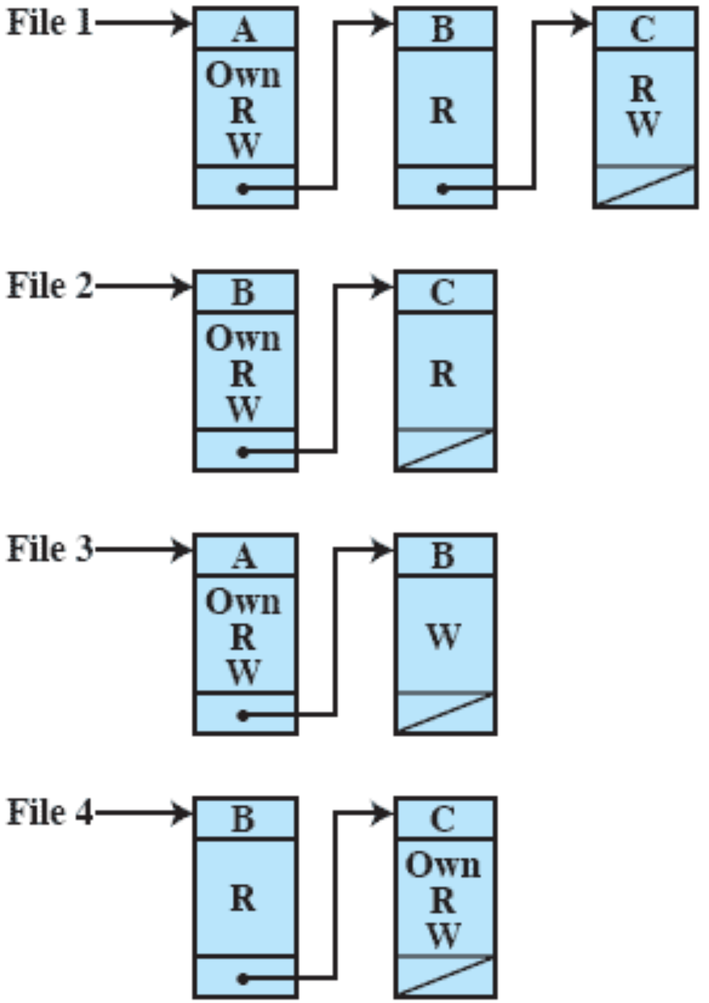
- The basic elements are:
- **subject** – an entity capable of accessing objects
- **object** – anything to which access is controlled
- **access right** – the way in which an object is accessed by a subject

	File 1	File 2	File 3	File 4	Account 1	Account 2
User A	Own R W		Own R W		Inquiry Credit	
User B	R	Own R W	W	R	Inquiry Debit	Inquiry Credit
User C	R W	R		Own R W		Inquiry Debit

(a) Access matrix

Access Control Lists

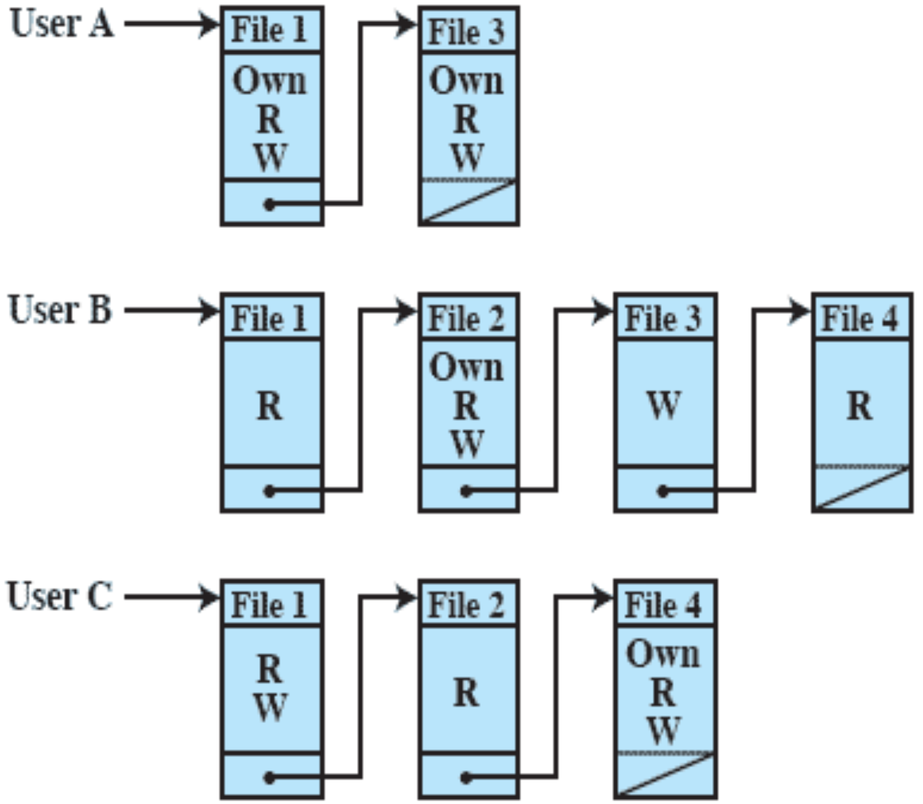
- A matrix may be decomposed by columns, yielding access control lists
- The access control list lists users and their permitted access rights



(b) Access control lists for files of part (a)

Capability Lists

- Decomposition by rows yields capability tickets
- A capability ticket specifies authorised objects and operations for a user



(c) Capability lists for files of part (a)

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

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

Named pipes
- an interprocess communications facility

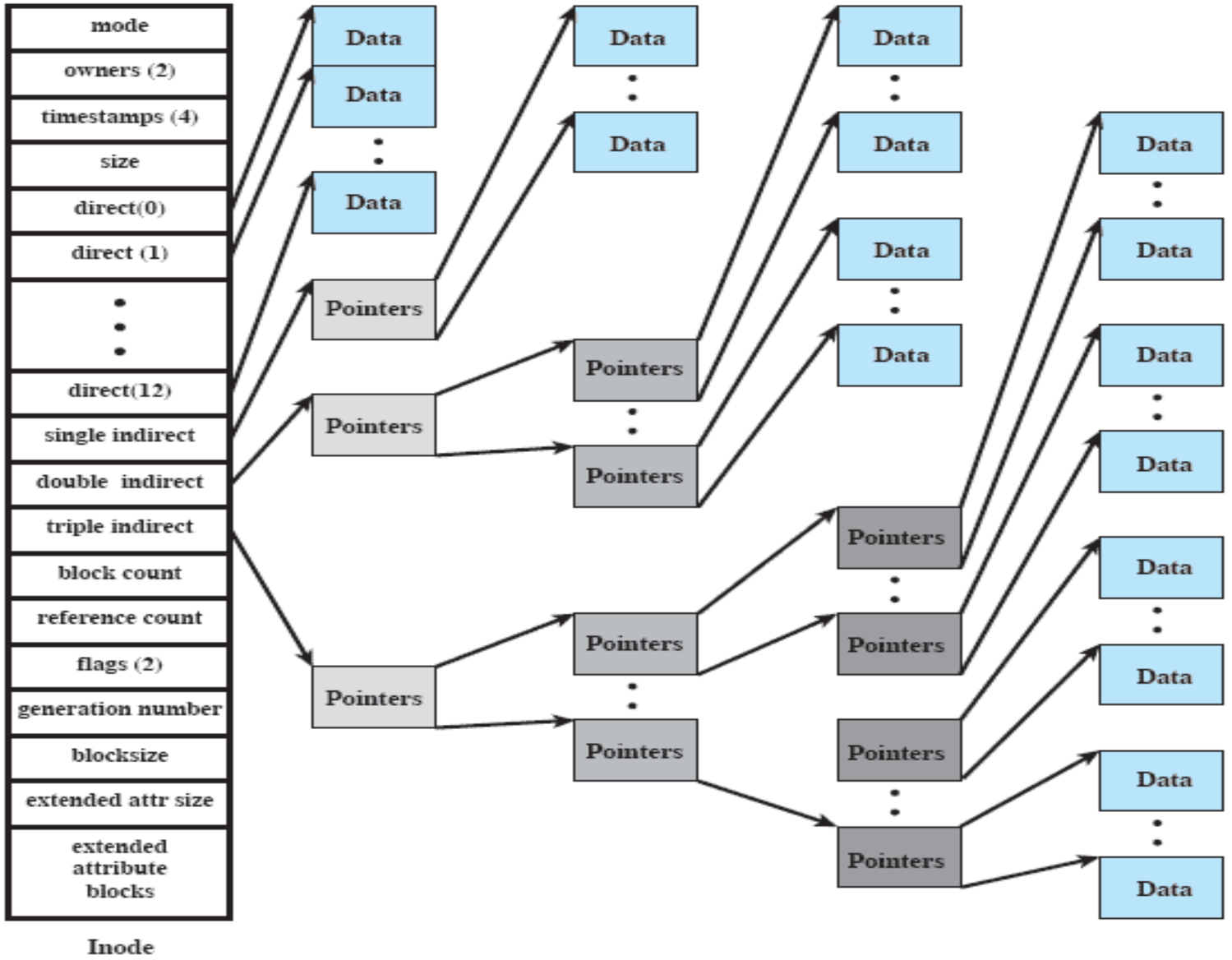
Links
- an alternative file name for an existing file

Symbolic links
- a data file that contains the name of the file it is linked to

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 File Structure



File Allocation

- 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 4 Kbyte Block Size

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

UNIX Directories and Inodes

- Directories are structured in a hierarchical tree
- Each directory can contain files and/or other directories
- A directory that is inside another directory is referred to as a subdirectory

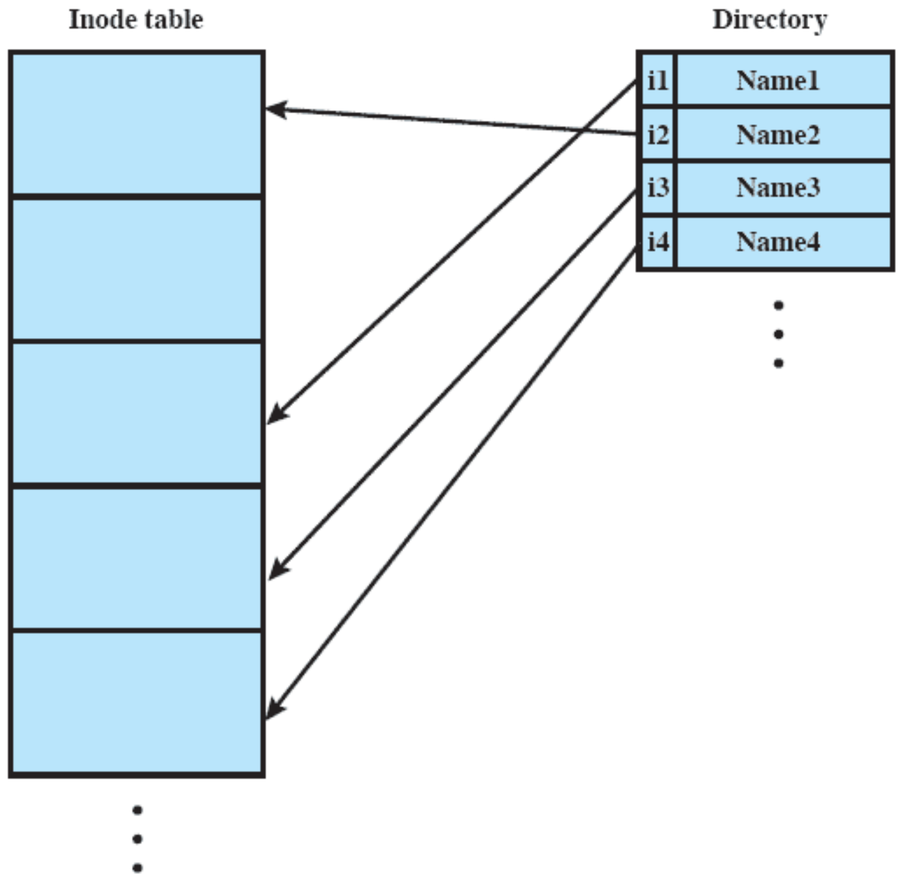
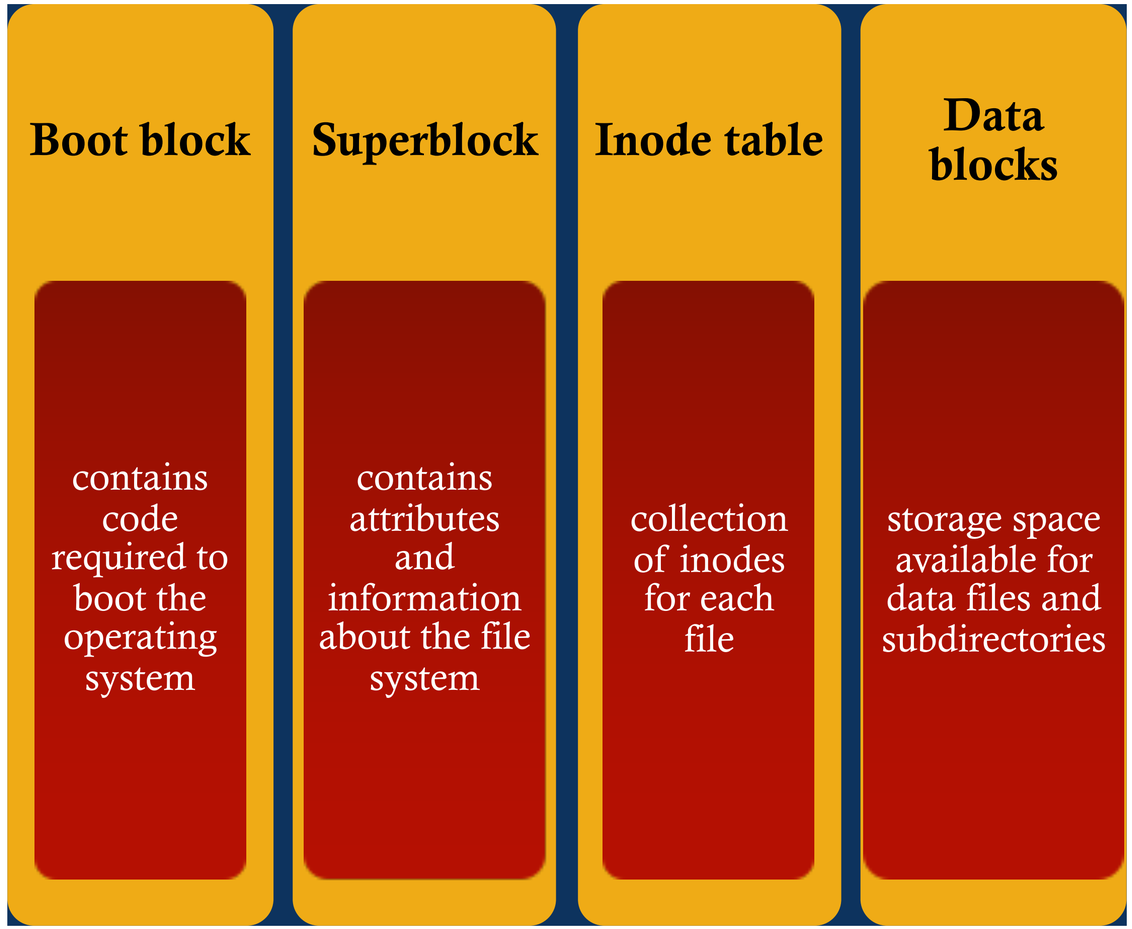


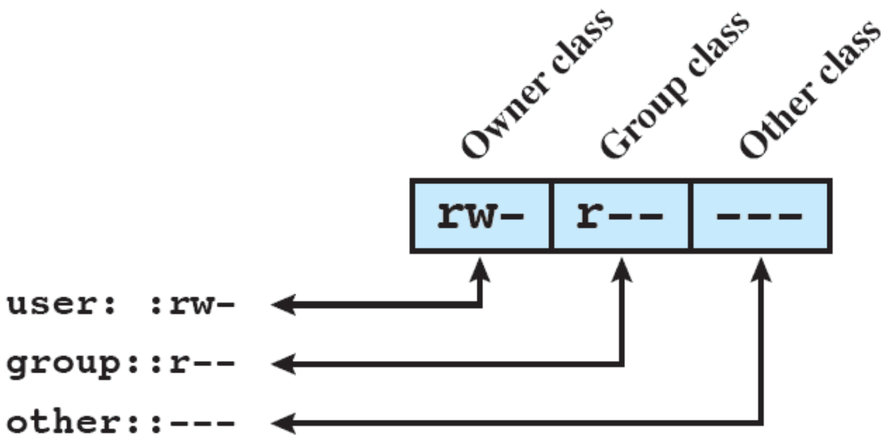
Figure 12.15 UNIX Directories and Inodes

Volume Structure

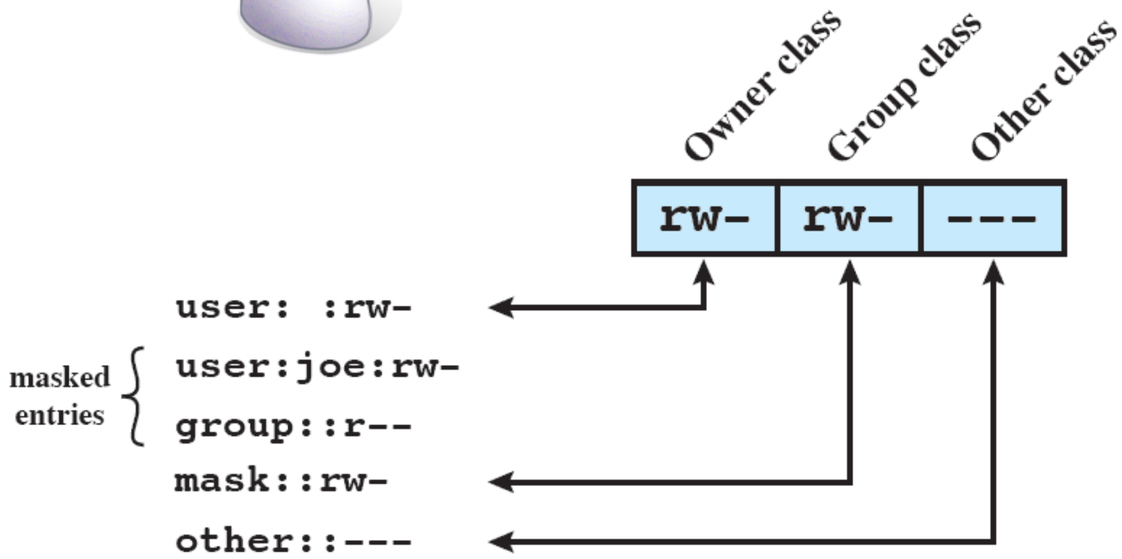
- A UNIX file system resides on a single logical disk or disk partition and is laid out with the following elements:



UNIX File Access Control



(a) Traditional UNIX approach (minimal access control list)



(b) Extended access control list

Access Control Lists in UNIX

Griffith UNIVERSITY

- FreeBSD allows the administrator to assign a list of UNIX user IDs and groups to a file
- Any number of users and groups can be associated with a file, each with three protection bits (read, write, execute)
- A file may be protected solely by the traditional UNIX file access mechanism
- FreeBSD files include an additional protection bit that indicates whether the file has an extended ACL

Linux Virtual File System (VFS)

- Presents a single, uniform file system interface to user processes
- Defines a common file model that is capable of representing any conceivable file system's general feature and behaviour
- Assumes files are objects that share basic properties regardless of the target file system or the underlying processor hardware

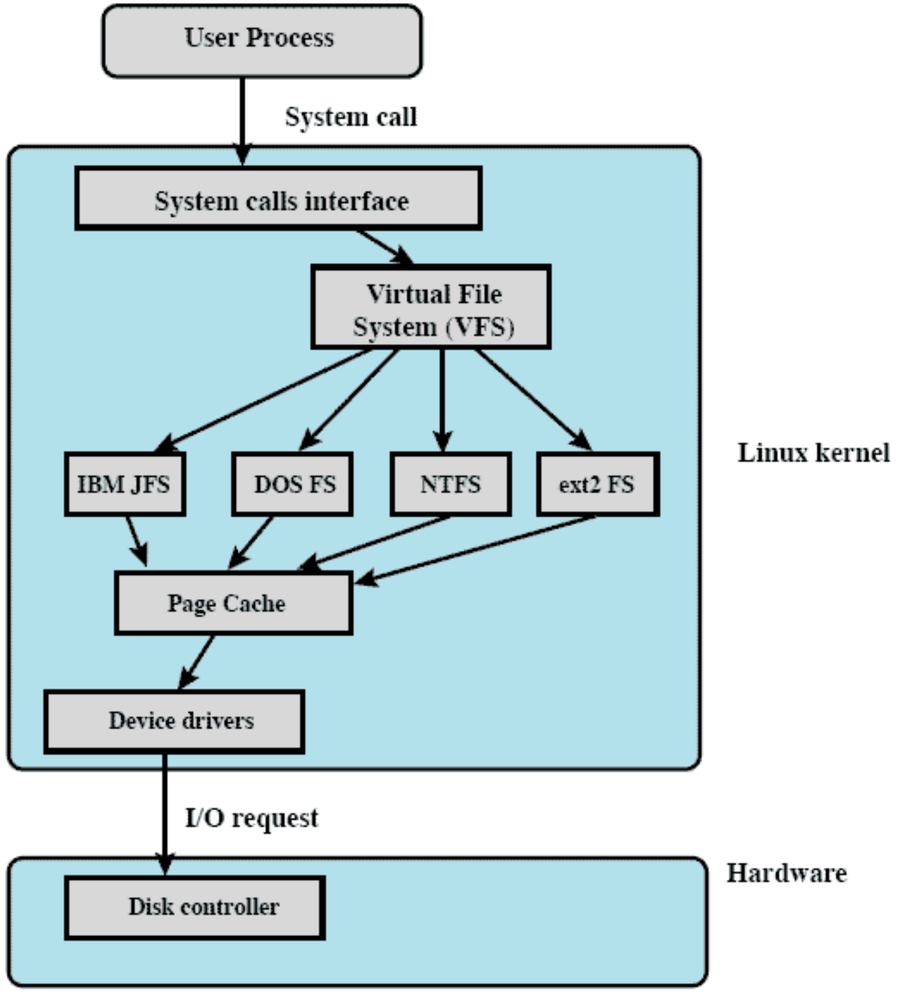


Figure 12.17 Linux Virtual File System Context

The Role of VFS Within the Kernel

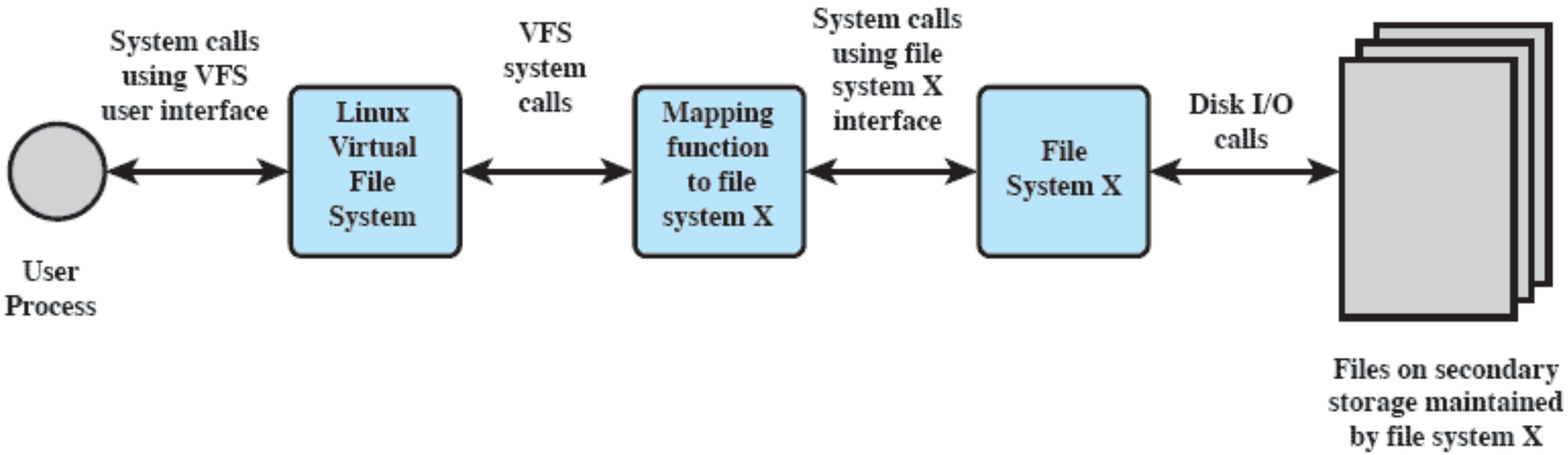


Figure 12.18 Linux Virtual File System Concept

Primary Object Types in VFS

Griffith UNIVERSITY

Superblock Object

- represents a specific mounted file system

Dentry Object

- represents a specific directory entry

Inode Object

- represents a specific file

File Object

- represents an open file associated with a process

Windows File System

- The developers of Windows NT designed a new file system, the New Technology File System (NTFS) which is intended 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

- NTFS makes use of the following disk storage concepts:

Sector

- the smallest physical storage unit on the disk
- the data size in bytes is a power of 2 and is almost always 512 bytes

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 bytes

Table 12.5
Windows NTFS Partition and Cluster Sizes

Volume Size	Sectors per Cluster	Cluster Size
≤ 512 Mbyte	1	512 bytes
512 Mbyte - 1 Gbyte	2	1K
1 Gbyte - 2 Gbyte	4	2K
2 Gbyte - 4 Gbyte	8	4K
4 Gbyte - 8 Gbyte	16	8K
8 Gbyte - 16 Gbyte	32	16K
16 Gbyte - 32 Gbyte	64	32K
> 32 Gbyte	128	64K

NTFS Volume Layout

- Every element on a volume is a file, and every file consists of a collection of attributes
    - even the data contents of a file is treated as an attribute

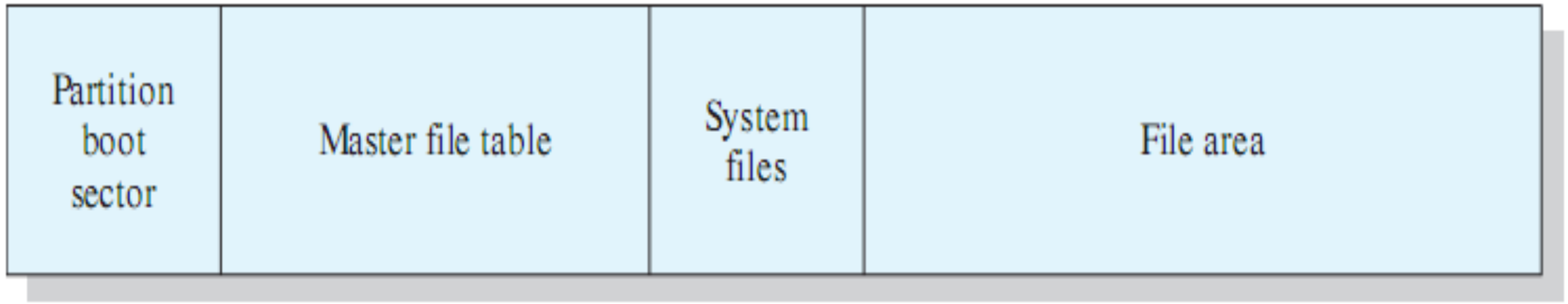


Figure 12.19 NTFS Volume Layout

Master File Table (MFT)

- The heart of the Windows file system is the MFT
- The MFT is organised 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

Table 12.6

Attribute Type	Description
Standard information	Includes access attributes (read-only, read/write, etc.); time stamps, including when the file was created or last modified; and how many directories point to the file (link count).
Attribute list	A list of attributes that make up the file and the file reference of the MFT file record in which each attribute is located. Used when all attributes do not fit into a single MFT file record.
File name	A file or directory must have one or more names.
Security descriptor	Specifies who owns the file and who can access it.
Data	The contents of the file. A file has one default unnamed data attribute and may have one or more named data attributes.
Index root	Used to implement folders.
Index allocation	Used to implement folders.
Volume information	Includes volume-related information, such as the version and name of the volume.
Bitmap	Provides a map representing records in use on the MFT or folder.

Windows NTFS Components

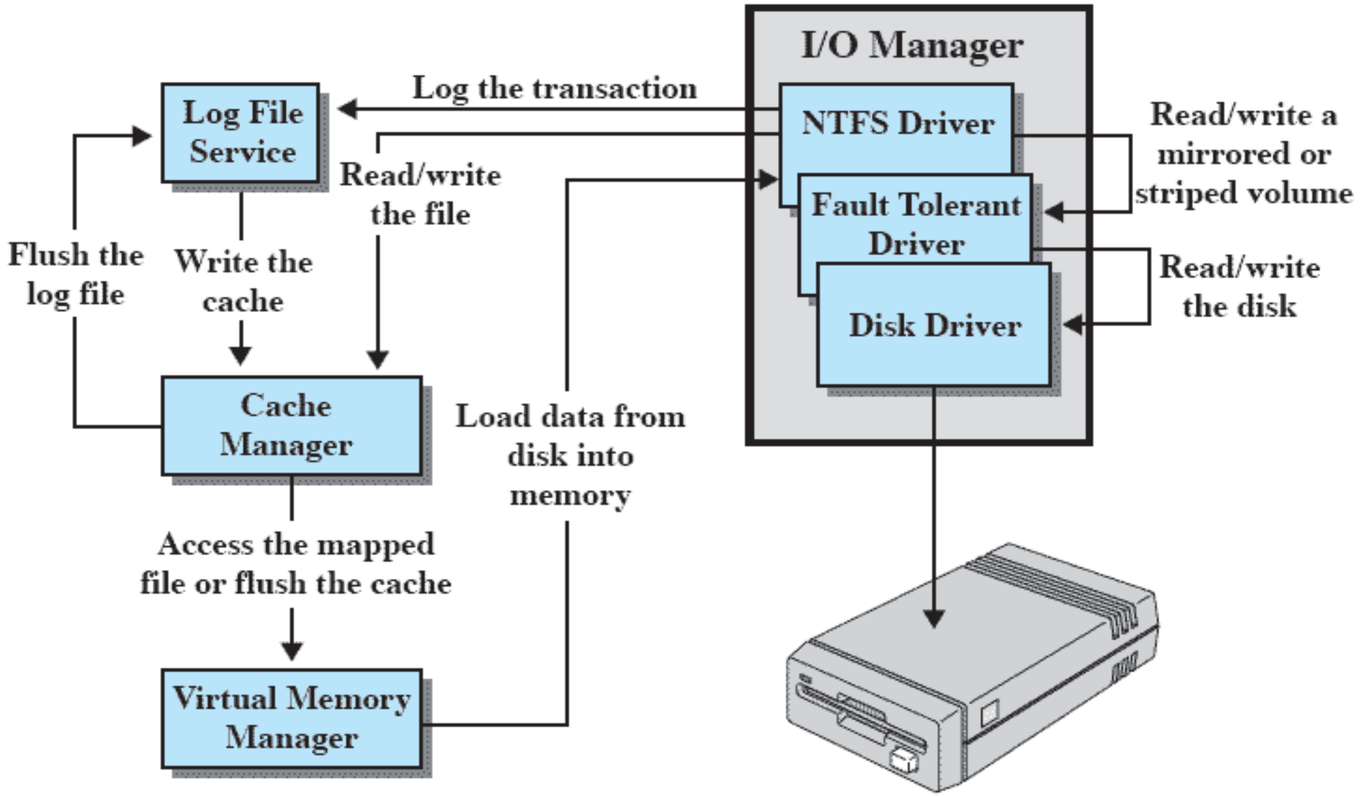


Figure 12.20 Windows NTFS Components

Summary

- **A file management system:**
    - is a set of system software that provides services to users and applications in the use of files
    - is typically viewed as a system service that is served by the operating system
- **Files:**
    - consist of a collection of records
    - if a file is primarily to be processed as a whole, a sequential file organisation is the simplest and most appropriate
    - if sequential access is needed but random access to individual file is also desired, an indexed sequential file may give the best performance
    - if access to the file is principally at random, then an indexed file or hashed file may be the most appropriate
    - directory service allows files to be organised in a hierarchical fashion
- **Some sort of blocking strategy is needed**
- **Key function of file management scheme is the management of disk space**
    - strategy for allocating disk blocks to a file
    - maintaining a disk allocation table indicating which blocks are free