GANESAR COLLEGE OF ARTS AND SCIENCE
MELAISIVAPURI
OPERATING SYSTEM

UNIT – V

FILE MANAGEMENT

THE FILE MANAGER

- The File Manager (also called the file management system) is the software responsible for creating, deleting, modifying, and controlling access to files—as well as for managing the resources used by the files.
- The File Manager provides support for libraries of programs and data to online users, for spooling operations, and for interactive computing.

Responsibilities of the File Manager:

- The File Manager has a complex job. It’s in charge of the system’s physical components, its information resources, and the policies used to store and distribute the files.

  To carry out its responsibilities, it must perform these four tasks:

  1. Keep track of where each file is stored.

  2. Use a policy that will determine where and how the files will be stored, making sure to efficiently use the available storage space and provide efficient access to the files.

  3. Allocate each file when a user has been cleared for access to it, then record its use.

  4. Deallocate the file when the file is to be returned to storage, and communicate its availability to others who may be waiting for it.

- For example, the file system is like a library, with the File Manager playing the part of the librarian who performs the same four tasks:

1. A librarian uses the catalog to keep track of each item in the collection; each entry lists the call number and the details that help patrons find the books they want.

2. The library relies on a policy to store everything in the collection including oversized books, magazines, books-on-tape, DVDs, maps, and videos.

3. When it’s requested, the item is retrieved from its shelf and the borrower’s name is noted in the circulation records.

4. When the item is returned, the librarian makes the appropriate notation in the circulation records and reserves it.

Definitions

- A **field** is a group of related bytes that can be identified by the user with a name, type, and size. A record is a group of related fields.
- A **file** is a group of related records that contains information to be used by specific application programs to generate reports..
- A **database** appears to the File Manager to be a type of file, but databases are more complex because they're actually groups of related files that are interconnected at various levels to give users flexibility of access to the data stored.
- **Program files** contain instructions and data files contain data; but as far as storage is concerned, the File Manager treats them exactly the same way.
- **Directories** are special files with listings of filenames and their attributes. Data collected to monitor system performance and provide for system accounting is collected into files.

Interacting with the File Manager

The user communicates with the File Manager, which responds to specific commands.

- OPEN,
- DELETE,
- RENAME,
- COPY.

Actually, files can be created with other system-specific terms: for example, the first time a user gives the command to save a file, it's actually the CREATE command.

OPEN NEW command within a program indicates to the File Manager that a file must be created.

These commands and many more were designed to be very simple to use.

Typical Volume Configuration

- ✓ The active files for a computer system reside on secondary storage units. Some devices accommodate removable storage units—such as CDs, DVDs, floppy disks, USB devices, and other removable media.
- ✓ Other devices feature integrated storage units, such as hard disks and nonremovable disk packs.
- ✓ Each volume in the system is given a name. An easy-to-access place on each unit: the innermost part of the CD or DVD, the beginning of the tape, or the first sector of the outermost track of the disk pack. Once identified, the operating system can interact with the storage unit.

- ✓ **The master file directory (MFD**) is stored immediately after the volume descriptor and lists the names and characteristics of every file contained in that volume. The filenames in the MFD can refer to program files, data files, and/or system files. And if the File Manager supports subdirectories, they're listed in the MFD as well. The remainder of the volume is used for file storage.
- ✓ The first operating systems supported only a single directory per volume. This directory was created by the File Manager and contained the names of files, usually organized in alphabetical, spatial, or chronological order.

Introducing Subdirectories

- ✓ File Managers create an MFD for each volume that can contain entries for both files and subdirectories.
- ✓ A subdirectory is created when a user opens an account to access the computer system. Although this user directory is treated as a file, its entry in the MFD is flagged to indicate to the File Manager that this file is really a subdirectory and has unique properties—in fact, its records are filenames pointing to files.

Each file entry in every directory contains information describing the file; it's called the file descriptor. Information typically included in a file descriptor includes the following:

• Filename—within a single directory, filenames must be unique; in some operating systems, the filenames are case sensitive

• File type—the organization and usage that are dependent on the system (for example, files and directories)

• File size—although it could be computed from other information, the size is kept here for convenience

• File location—identification of the first physical block (or all blocks) where the file is stored

File-Naming Conventions

- ✓ A file's name can be much longer than it appears.
- ✓ The two components common to many filenames are a **relative filename** and an **extension.**
- ✓ To avoid confusion, in the following discussion we'll use the term "complete filename" to identify the file's **absolute filename** (that's the long name that includes all path information), and **"relative filename"** to indicate the name without path information that appears in directory listings and folders.

Some extensions (such as EXE, BAT, COB, and FOR) are restricted by certain operating systems because they serve as a signal to the system to use a specific compiler or program to run these files.

There may be other components required for a file's complete name. Here's how a file named INVENTORY_COST.DOC is identified by different operating systems:

1. Using a Windows operating system and a personal computer with three disk drives, the file's complete name is composed of its relative name and extension, preceded by the drive label and directory name:

   C:\IMFST\FLYNN\INVENTORY_COST.DOC

2. A UNIX or Linux system might identify the file as:

   /usr/imfst/flynn/inventory_cost.doc

   The first entry is represented by the forward slash (/). This represents a special master directory called the root.
   Next is the name of the first subdirectory, usr/imfst, followed by a sub-subdirectory, /flynn, in this multiple directory system. The final entry is the file's relative name, inventory_cost.doc.

- ✓ Why don't users see the complete file name when accessing a file?
  First, the File Manager selects a directory for the user when the interactive session begins, so all file operations requested by that user start from this "home" or "base" directory. Second, from this home directory, the user selects a subdirectory, which is called a **current directory** or **working directory**.

FILE ORGANIZATION

- ✓ File organization, we are talking about the arrangement of records within a file because all files are composed of records.
- ✓ When a user gives a command to modify the contents of a file, it's actually a command to access records within the file.

Record Format

All files are composed of records. When a user gives a command to modify the contents of a file, it's actually a command to access records within the file.

Within each file, the records are all presumed to have the same format: they can be of fixed length or of variable length,.

(a)

Dan	Whitesto	1243 Ele	Harrisbu	PA	412 683

(b)

Dan	Whitestone	1243 Elementary Ave.	Harrisburg	PA

- **Fixed-length records** are the most common because they're the easiest to access directly.
  The critical aspect of fixed-length records is the size of the record. If it's too small—smaller than the number of characters to be stored in the record—the leftover characters are truncated.
  But if the record size is too large— larger than the number of characters to be stored—storage space is wasted.

- **Variable-length records** don't leave empty storage space and don't truncate any characters, thus eliminating the two disadvantages of fixed-length records.
  The record format, how it's blocked, and other related information is kept in the file descriptor.

Physical File Organization

The physical organization of a file has to do with the way records are arranged and the characteristics of the medium used to store it.

On magnetic disks (hard drives), files can be organized in one of several ways: sequential, direct, or indexed sequential.

To select the best of these file organizations, the programmer or analyst usually considers these practical characteristics:

• Volatility of the data—the frequency with which additions and deletions are made
• Activity of the file—the percentage of records processed during a given run
• Size of the file
• Response time—the amount of time the user is willing to wait before the requested operation is completed.

Sequential record organization is by far the easiest to implement because records are stored and retrieved serially, one after the other. To find a specific record, the file is searched from its beginning until the requested record is found.

- ✓ To speed the process, some optimization features may be built into the system. One is to select a key field from the record and then sort the records by that field before storing them.
- ✓ Although this technique aids the search process, it complicates file maintenance because the original order must be preserved every time records are added or deleted.

A direct record organization uses direct access files, which, of course, can be implemented only on direct access storage devices.

- ✓ These files give users the flexibility of accessing any record in any order without having to begin a search from the beginning of the file to do so. It's also known as "random organization," and its files are called "random access files."
- ✓ Records are identified by their relative addresses—their addresses relative to the beginning of the file. These **logical addresses** are computed when the records are stored and then again when the records are retrieved.
- ✓ The user identifies a field (or combination of fields) in the record format and designates it as the key field because it uniquely identifies each record. The program used to store the data follows a set of instructions, called a **hashing algorithm**.

Indexed sequential record organization combines the best of sequential and direct access.

- ✓ It's created and maintained through an Indexed Sequential Access Method (ISAM) application, which removes the burden of handling overflows and preserves record order from the shoulders of the programmer.
- ✓ It uses this information to generate an index file through which the records are retrieved. This organization divides an ordered sequential file into blocks of equal size.
- ✓ Their size is determined by the File Manager to take advantage of physical storage devices and to optimize retrieval strategies.
- ✓ An indexed sequential file also has overflow areas, but they're spread throughout the file, perhaps every few records, so expansion of existing records can take place, and new records can be located in close physical sequence as well as in logical sequence.
- ✓ For most dynamic files, indexed sequential is the organization of choice because it allows both direct access to a few requested records and sequential access to many.

PHYSICAL STORAGE ALLOCATION

- ✓ The File Manager must work with files not just as whole units but also as logical units or records.
- ✓ Records within a file must have the same format, but they can vary in length,
- ✓ In turn, records are subdivided into fields. In most cases, their structure is managed by application programs and not the operating system.

Contiguous Storage

- When records use contiguous storage, they're stored one after the other. This was the scheme used in early operating systems.

- It’s very simple to implement and manage.
- Any record can be found and read, once its starting address and size are known, so the directory is very streamlined.
- The primary disadvantage is that a file can’t be expanded unless there’s empty space available immediately.
- The second disadvantage is fragmentation (slivers of unused storage space), which can be overcome by compacting and rearranging files.

The File Manager keeps track of the empty storage areas by treating them as files—they’re entered in the directory but are flagged to differentiate them from real files. Usually the directory is kept in order by sector number, so adjacent empty areas can be combined into one large free space.

Noncontiguous Storage

- **Noncontiguous storage allocation** allows files to use any storage space available on the disk.
- A file’s records are stored in a contiguous manner, only if there’s enough empty space. Any remaining records and all other additions to the file are stored in other sections of the disk.
- In some systems these are called the extents of the file and are linked together with pointers.
- File extents are usually linked in one of two ways.
- The directory entry consists of the filename, the storage location of the first extent, the location of the last extent, and the total number of extents not counting the first.

Indexed Storage

- Indexed storage allocation allows direct record access by bringing together the pointers linking every extent of that file into an index block.
- Every file has its own index block, which consists of the addresses of each disk sector that make up the file.
- When a file is created, the pointers in the index block are all set to null.
- Then, as each sector is filled, the pointer is set to the appropriate sector address—to be precise, the address is removed from the empty space list and copied into its position in the index block.

ACCESS METHODS

- ✓ Access methods are dictated by a file’s organization; the most flexibility is allowed with indexed sequential files and the least with sequential.
- ✓ A file that has been organized in sequential fashion can support only sequential access to its records, and these records can be of either fixed or variable length.

- ✓ The File Manager uses the address of the last byte read to access the next sequential record. Therefore, the current byte address (CBA) must be updated every time a record is accessed, such as when the READ command is executed.

Record 1	Record 2	Record 3	Record 4	Record 5

RL = x RL = x RL = x RL = x RL = x

(a)

m	Record 1	n	Record 2	p	Rec.3	q	Record 4

RL = m RL = n RL = p RL = q

(b)

Fixed- length versus variable length records.

(a) Fixed length records have the same number of bytes, so record length (RL) is the constant x. (b) With variable-length records, RL isn't a constant.

Sequential Access

- For sequential access of fixed-length records, the CBA is updated simply by incrementing it by the record length (RL), which is a constant:

$$\text{CBA} = \text{CBA} + \text{RL}$$

- For sequential access of variable-length records, the File Manager adds the length of the record (RL) plus the number of bytes used to hold the record length N, (which holds the constant shown as m, n, p, or q,) to the CBA.

$$\text{CBA} = \text{CBA} + \text{N} + \text{RL}$$

Direct Access

- If a file is organized in direct fashion, it can be accessed easily in either direct or sequential order if the records are of fixed length.
- In the case of direct access with fixed-length records, the CBA can be computed directly from the record length and the desired record number RN (information provided through the READ command) minus 1:

$$CBA = (RN - 1) * RL$$

For example, if we're looking for the beginning of the eleventh record and the fixed record length is 25 bytes, the CBA would be:

$$(11 - 1) * 25 = 250$$

Direct access was first called "random access" because it allowed access to data stored in a random order. It did not require sequential storage.

Levels in a File Management System

The highest level module is called the "basic file system," and it passes information through the access control verification module to the logical file system, which, in turn, notifies the physical file system, which works with the Device Manager. Each level of the file management system is implemented using structured and modular programming techniques that also set up a hierarchy—that is, the higher positioned modules pass information to the lower modules,

- ✓ so that they, in turn, can perform the required service and continue the communication down the chain to the lowest module, which communicates with the physical device and interacts with the Device Manager.
- ✓ Only then is the record made available to the user's program.

Each of the modules can be further subdivided into more specific tasks, as we can see when we follow this I/O instruction through the file management system:

READ RECORD NUMBER 7 FROM FILE CLASSES INTO STUDENT

CLASSES is the name of a direct access file previously opened for input, and STUDENT is a data record previously defined within the program and occupying specific memory locations.

- ✓ This information is used by the basic file system, which activates the access control verification module to verify that this user is permitted to perform this operation with this file.
- ✓ If access is allowed, information and control are passed along to the logical file system. If not, a message saying "access denied" is sent to the user.

Using the information passed down by the basic file system, the logical file system transforms the record number to its byte address using the familiar formula:

$$CBA = (RN - 1) * RL$$

This result, together with the address of the first physical record and, in the case where records are blocked, the physical block size, is passed down to the physical file system, which computes the location where the desired record physically resides.

If there's more than one record in that block, it computes the record's offset within that block using these formulas:

- block number = integer $[\frac{byte\ address}{physical\ block\ size}]$ + address of the first physical record

- offset = remainder $[\frac{byte\ address}{physical\ block\ size}]$

This information is passed on to the device interface module, which, in turn, transforms the block number to the actual cylinder/surface/record combination needed to retrieve the information from the secondary storage device.

ACCESS CONTROL VERIFICATION MODULE

- The first operating systems couldn't support file sharing among users. For instance, early systems needed 10 copies of a compiler to serve 10 users.
- . In fact, any file can be shared—from data files and user-owned program files to system files. The advantages of file sharing are numerous.
- In addition to saving space, it allows for synchronization of data updates, as when two applications are updating the same data file.
- It also improves the efficiency of the system's resources because if files are shared in main memory, then there's a reduction of I/O operations.
- There are five possible actions that can be performed on a file—the ability to READ only, WRITE only, EXECUTE only, DELETE only, or some combination of the four. Each file management system has its own method to control file access.

Access Control Matrix

- ✓ The access control matrix is intuitively appealing and easy to implement, but because of its size it only works well for systems with a few files and a few users.
- ✓ In the matrix, each column identifies a user and each row identifies a file. The intersection of the row and column contains the access rights for that user to that file, as Table 8.2 illustrates.

	User 1	User 2	User 3	User 4	User 5
File 1	RWED	R-E-	----	RWE-	--E
File 2	----	R-E-	R-E-	--E-	---
File 3	----	RWED	----	--E-	---
File 4	R-E-	----	----	----	RWED
File 5	----	----	----	----	RWED

- ✓ The access control matrix showing access rights for each user for each file. User 1 is allowed unlimited access to File 1 but is allowed only to read and execute File 4 and is denied access to the three other files. R = Read Access, W = Write Access, E = Execute Access, D = Delete Access, and a dash (-) = Access Not Allowed.

Access Control Lists

- ✓ The access control list is a modification of the access control matrix.
- ✓ Each file is entered in the list and contains the names of the users who are allowed to access it and the type of access each is permitted.
- ✓ To shorten the list, only those who may use the file are named; those denied any access are grouped under a global heading such as WORLD, as shown in Table 8.4.

File	Access
File 1	USER1 (RWED), USER2 (R-E-), USER4 (RWE-), USER5 (--E-), WORLD (----)
File 2	USER2 (R-E-), USER3 (R-E-), USER4 (--E-), WORLD (----)
File 3	USER2 (RWED), USER4 (--E-), WORLD (----)
File 4	USER1 (R-E-), USER5 (RWED), WORLD (----)
File 5	USER5 (RWED), WORLD (----)

Some systems shorten the access control list even more by putting every user into a category: system, owner, group, and world.

- ✓ SYSTEM or ADMIN is designated for system personnel who have unlimited access to all files in the system. The OWNER has absolute control over all files created in the owner's account.
- ✓ An owner may create a GROUP file so that all users belonging to the appropriate group have access to it. WORLD is composed of all other users in the system.

Capability Lists

- ✓ A capability list shows the access control information from a different perspective. It lists every user and the files to which each has access, as shown in Table 8.5.

User	Access
User 1	File 1 (RWED), File 4 (R-E-)
User 2	File 1 (R-E-), File 2 (R-E-), File 3 (RWED)
User 3	File 2 (R-E-)
User 4	File 1 (RWE-), File 2 (--E-), File 3 (--E-)
User 5	File 1 (--E-), File 4 (RWED), File 5 (RWED)

- ✓ An access control list showing which users are allowed to access each file. This method requires less storage space than an access control matrix.
- ✓ A capability list shows files for each user and requires less storage space than an access control matrix; and when users are added or deleted from the system, is easier to maintain than an access control list.

DATA COMPRESSION

- ✓ Data compression algorithms consist of two types: lossless algorithms typically used for text or arithmetic files, which retain all the data in the file throughout the compression-decompression process; and lossy algorithms, which are typically used for image and sound files and remove data permanently.
- ✓ At first glance, one wouldn’t think that a loss of data would be tolerable; but when the deleted data is unwanted noise, tones beyond a human’s ability to hear, or light spectrum that we can’t see, deleting this data can be undetectable and therefore acceptable.

Text Compression

- ✓ To compress text in a database, three methods are described briefly here: records with repeated characters, repeated terms, and front-end compression.
- ✓ Records with repeated characters: Data in a fixed-length field might include a short name followed by many blank characters. This can be replaced with a variable-length field and a special code to indicate how many blanks were truncated.

For example, let’s say the original string, ADAMS, looks like this when it’s stored uncompressed in a field that’s 15 characters wide (b stands for a blank character):

ADAMSbbbbbbbbbb

When it’s encoded it looks like this:

ADAMSb10

Likewise, numbers with many zeros can be shortened, too, with a code (in this case, the pound sign #) to indicate how many zeros must be added to recreate the original number. For instance, if the original entry is this number:

300000000

the encoded entry is this:

3#8

<u>Other Compression Schemes</u>

- ✓ Lossy compression allows a loss of data from the original file to allow significant compression.
- ✓ This means the compression process is irreversible as the original file cannot be reconstructed.
- ✓ The specifics of the compression algorithm are highly dependent on the type of file being compressed, with JPEG a popular option for still images and MPEG for video images.
- ✓ For video and music files, the International Organization for Standardization (ISO) has issued MPEG standards that “are international standards dealing with the compression, decompression, processing, and coded representation of moving pictures, audio, and their combination.”

****