11.1 File Systems

In Chapter 5 we established the differences between main and secondary memory. Recall that main memory is where active programs and data are held while in use. Main memory is volatile, meaning that the data stored on it is lost if electric power is turned off. Secondary memory is nonvolatile—the data stored on it is maintained even when power is not on. Thus we use secondary memory for permanent storage of our data.

The most widely used secondary storage device is the magnetic disk drive. It includes both the hard drives that are found in the computer's main box and disks that are portable and can be moved easily between computers. The basic concepts underlying both types of disks are the same. Other secondary memory devices, such as tape drives, are used primarily for archival purposes. While many of the concepts that we explore in this chapter apply to all secondary storage devices, it's perhaps easiest to think about a standard disk drive.

We store data on a disk in files, a mechanism for organizing data on an electronic medium. A file is a named collection of related data. From the user's point of view, a file is the smallest amount of data that can be written to secondary memory. Organizing everything into files presents a uniform view for data storage. A file system is the logical view that an operating system provides so that users can manage data as a collection of files. A file system is often organized by grouping files into directories.

- File A named collection of data, used for organizing secondary memory
- File system The operating system's logical view of the files it manages
- Directory A named group of files
- Text file A file that contains characters
- Binary file A file that contains data in a specific format, requiring a special interpretation of its bits

A file is a generic concept. Different types of files are managed in different ways. A file, in general, contains a program (in some form) or data (of one type or another). Some files have a very rigid format; others are more flexible.

A file may be considered a sequence of bits, bytes, lines, or records, depending on how you look at it. As with any data in memory, you have to apply an interpretation to the bits stored in a file before they have meaning. The creator of a file decides how the data in a file is organized, and any users of the file must understand that organization.

Text and Binary Files

All files can be broadly classified as either text files or binary files. In a text file, the bytes of data are organized as characters from the ASCII or Unicode character sets. (Character sets are described in Chapter 3.) A binary file requires a specific interpretation of the bits based on the data in the file.

The terms "text file" and "binary file" are somewhat misleading. They seem to imply that the information in a text file is not stored as binary data. Ultimately, however, all data on a computer is stored as binary digits. These terms refer to how those bits are formatted: as chunks of 8 or 16 bits, interpreted as characters, or in some other special format.

Some information lends itself to a character representation, which often makes it easier for a human to read and modify the information. Though text files contain only characters, those characters can represent a variety of information. For example, an operating system may store much of its data as text files, such as information about user accounts. A program written in a high-level language is stored as a text file, which is sometimes referred to as a *source file*. Any text editor can be used to create, view, and change the contents of a text file, no matter what specific type of information it contains.

For other information types, it is more logical and efficient to represent data by defining a specific binary format and interpretation. Only programs set up to interpret that type of data can then be used to view or modify it. For example, many types of files store image information: bitmap, GIF, JPEG, and TIFF, to name a few. As we discussed in Chapter 3, even though each stores information about an image, all of these files store that information in different ways. Their internal formats are very specific. A program must be set up to view or modify a specific type of binary file. That's why a program that can handle a GIF image may not be able to handle a TIFF image, or vice versa.

Some files you might assume to be text files actually are not. Consider, for instance, a report that you type in a word processing program and save to disk. The document is actually stored as a binary file because, in addition to the characters in the document, it contains information about formatting, styles, borders, fonts, colors, and "extras" such as graphics or clip art. Some of the data (the characters themselves) are stored as text, but the additional information requires that each word processing program have its own format for storing the data in its document files.

File Types

Most files, whether they are in text or binary format, contain a specific type of information. For example, a file may contain a Java™ program, or a JPEG image, or an MP3 audio clip. Other files contain files created by specific applications, such as a Microsoft® Word document or a Visio drawing. The kind of information contained in a document is called the **file type**. Most operating systems recognize a list of specific file types.

A common mechanism for specifying a file type is to indicate the type as part of the file's name. File names are often separated, usually by a period, into two parts: the main name and the **file extension**. The extension indicates the type of the file. For example, the `.java` extension in the file name `MyProg.java` indicates that it is a Java source code program file. The `.jpg` extension in the file name `family.jpg` indicates that it is a JPEG image file. Figure 11.1 lists some common file extensions.

File type The specific kind of information contained in a file, such as a Java program or a Microsoft Word document

File extension Part of a file name that indicates the file type

FIGURE 11.1 Some common file types and their extensions

Extensions	File type
`txt`	text data file
`mp3, au, wav`	audio file
`gif, tiff, jpg`	image file
`doc, wp3`	word processing document
`java, c, cpp`	program source files

File types allow the operating system to operate on the file in ways that make sense for that file. They also usually make life easier for the user. The operating system keeps a list of recognized file types and associates each type with a particular kind of application program. In an operating system with a graphical user interface (GUI), a particular icon is often associated with a file type as well. When you see a file in a folder, it is shown with the appropriate icon in the GUI. That makes it easier for the user to identify a file at a glance because now both the name of the file and its icon indicate which type of file it is. When you double-click on the icon to open the program, the operating system starts the program associated with that file type and loads the file.

For example, you might like a particular editor that you use when developing a Java program. You can register the `.java` file extension with the operating system and associate it with that editor. Then whenever you open a file with a `.java` extension, the operating system runs the appropriate editor. The details of how you associate an extension with an application program depend on the operating system you are using.

Some file extensions are associated with particular programs by default, which you may change if appropriate. In some cases, a file type could be associated with various types of applications, so you have some choice. For example, your system may currently associate the `.gif` extension with a particular Web browser, so that when you open a GIF image file, it is displayed in that browser window. You may choose to change the association so that when you open a GIF file, it is brought into your favorite image editor instead.

A file extension is merely an indication of what the file contains. You can name a file anything you want (as long as you use the characters that the operating system allows for file names). You could give any file a `.gif` extension, for instance, but that doesn't make it a GIF image file. Changing the extension does not change the data in the file or its internal format. If you attempt to open a misnamed file in a program that expects a particular format, you will get an error message.

File Operations

With the help of the operating system, you might perform any of several operations to and with a file:

- Create a file
- Delete a file
- Open a file
- Close a file
- Read data from a file
- Write data to a file
- Reposition the current file pointer in a file
- Append data to the end of a file
- Truncate a file (delete its contents)
- Rename a file
- Copy a file

Let's briefly examine how each of these operations is accomplished.

The operating system keeps track of secondary memory in two ways. It maintains a table indicating which blocks of memory are free (that is, available for use) and, for each directory, it maintains a table that records information about the files in that directory. To create a file, the operating system finds enough free space in the file system for the file content, puts an entry for the file in the appropriate directory table, and records the name and location of the file. To delete a file, the operating system indicates that the memory space previously used by the file is now free and removes the appropriate entry in the directory table.

Most operating systems require that a file be opened before read and write operations are performed on it. The operating system maintains a small table of all currently open files to avoid having to search for the file in the large file system every time a subsequent operation is performed. To close the file when it is no longer in active use, the operating system removes the entry in the open file table.

At any point in time, an open file has a current file pointer (an address) indicating the place where the next read or write operation should occur. Some systems keep a separate read pointer and a write pointer for a file. Reading a file means that the operating system delivers a copy of the data in the file, starting at the current file pointer. After the read occurs, the file pointer is updated. When data is written to a file, the data is stored at the location indicated by the current file pointer, and then the file pointer is updated. Often an operating system allows a file to be open for reading or writing, but not for both operations at the same time.

The current file pointer for an open file might be repositioned to another location in the file to prepare for the next read or write operation. Appending

data to the end of a file requires that the file pointer be positioned to the end of a file; the appropriate data is then written at that location.

It is sometimes useful to "erase" the data in a file. Truncating a file means deleting the contents of the file without removing the administrative entries in the file tables. This operation avoids the need to delete a file and then recreate it. Sometimes the truncating operation is sophisticated enough to erase part of a file, from the current file pointer to the end of the file.

An operating system also provides an operation to change the name of a file, which is called renaming the file. Likewise, it provides the ability to create a complete copy of the contents of a file, giving the copy a new name.

File Access

The data in a file can be accessed in several different ways. Some operating systems provide only one type of file access, whereas others provide a choice of access methods. The type of access available for a given file is established when the file is created.

Let's examine the two primary access techniques: sequential access and direct access. The differences between these two techniques are analogous to the differences between the sequential nature of magnetic tape and the direct access offered by a magnetic disk, as discussed in Chapter 5. However, both types of files can be stored on either type of medium. File access techniques define the ways that the current file pointer can be repositioned. They are independent of the physical restrictions of the devices on which the file is stored.

Sequential file access The technique in which data in a file is accessed in a linear fashion

Direct file access The technique in which data in a file is accessed directly, by specifying logical record numbers

The most common access technique, and the simplest to implement, is sequential file access, which views the file as a linear structure. It requires that the data in the file be processed in order. Read and write operations move the current file pointer according to the amount of data that is read or written. Some systems allow the file pointer to be reset to the beginning of the file and/or to skip forward or backward by a certain number of records. See Figure 11.2.

Files with direct file access are conceptually divided into numbered logical records. Direct access allows the user to set the file pointer to any

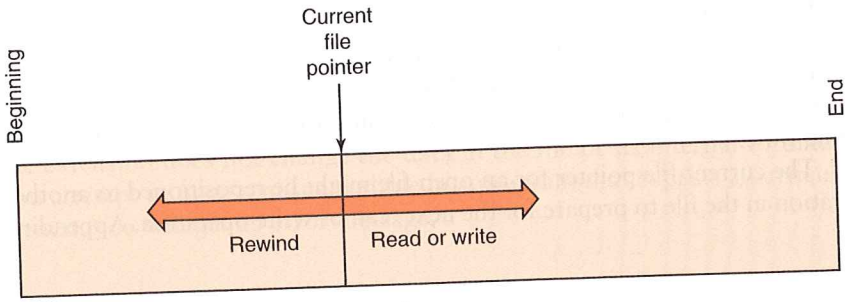

FIGURE 11.2 Sequential file access

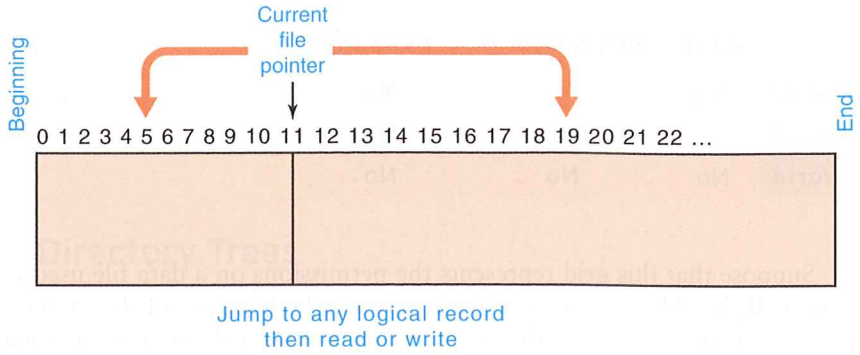

FIGURE 11.3 Direct file access

particular record by specifying the record number. Therefore, the user can read and write records in any particular order desired, as shown in Figure 11.3. Direct access files are more complicated to implement, but are helpful in situations where specific portions of large data stores must be available quickly, such as in a database.

File Protection

In multiuser systems, file protection is of primary importance. That is, we don't want one user to be able to access another user's files unless such access is specifically allowed. It is the operating system's responsibility to ensure valid file access. Different operating systems administer their file protection in different ways. In any case, a file protection mechanism determines who can use a file and for what general purpose.

For example, a file's protection settings in the UNIX operating system are divided into three categories: Owner, Group, and World. Under each category you can determine whether the file can be read, written, and/or executed. Under this mechanism, if you can write to a file, you can also delete the file.

Each file is "owned" by a particular user, who is often the creator of the file. The Owner usually has the strongest permissions regarding the file. A file may also have a group name associated with it—a group is simply a list of users. The Group permissions apply to all users in the associated group. You might create Group permissions, for instance, for all users who are working on a particular project. Finally, World permissions apply to anyone who has access to the system. Because these permissions give access to the largest number of users, they are usually the most restricted.

Using this technique, the permissions on a file can be shown in a 3×3 grid:

	Read	Write/Delete	Execute
Owner	Yes	Yes	No
Group	Yes	No	No
World	No	No	No

Suppose that this grid represents the permissions on a data file used in project Alpha. The owner of the file (perhaps the manager of the project) may read from or write to the file. Suppose also that the owner sets up a group (using the operating system) called TeamAlpha, which contains all members of the project team, and associates that group with this data file. The members of the TeamAlpha group may read the data in the file, but may not change it. No one else is given any permission to access the file. Note that no user is given execute privileges for the file because it is a data file, not an executable program.

Other operating systems set up their protection schemes in different ways, but the goal is the same: to control access so as to protect against deliberate attempts to gain inappropriate access as well as minimize inadvertent problems caused by well-intentioned but hazardous users.

11.2 Directories

As mentioned earlier, a directory is a named collection of files. It is a way to group files so that you can organize them in a logical manner. For example, you might place all of your papers and notes for a particular class in a directory created for that class. The operating system must carefully keep track of directories and the files they contain.

A directory, in most operating systems, is represented as a file. The directory file contains data about the other files in the directory. For any given file, the directory contains the file name, the file type, the address on disk where the file is stored, and the current size of the file. The directory also contains information about the protections set up for the file. In addition, it may hold information describing when the file was created and when it was last modified.

The internal structure of a directory file could be set up in a variety of ways, and we won't explore those details here. However, once it is set up, this structure must be able to support the common operations that are performed on directory files. For instance, the user must be able to list all of the files in the directory. Other common operations are creating,

?

RFID tags

As you leave a store after buying a pack of batteries, the batteries "tell" the store's inventory system to order batteries because stock is low. Radio-frequency identification (RFID) makes this type of communication possible. If the battery packaging has an RFID tag, it can tell a central radio-frequency transceiver where it is. In addition to items in retail stores, RFID technology is used to track shipping pallets, library books, vehicles, and animals. If you've ever used EZPass to go through a toll booth, or SpeedPass® to pay for your gas, you've used RFID technology. Researchers have even experimented with implanting RFID tags in people! In 2004, a club owner in Barcelona, Spain, and Rotterdam, the Netherlands, offered to implant his VIP customers with RFID tags. These chips identified the customers as VIPs, and the chips were used by the customers to pay for their drinks.