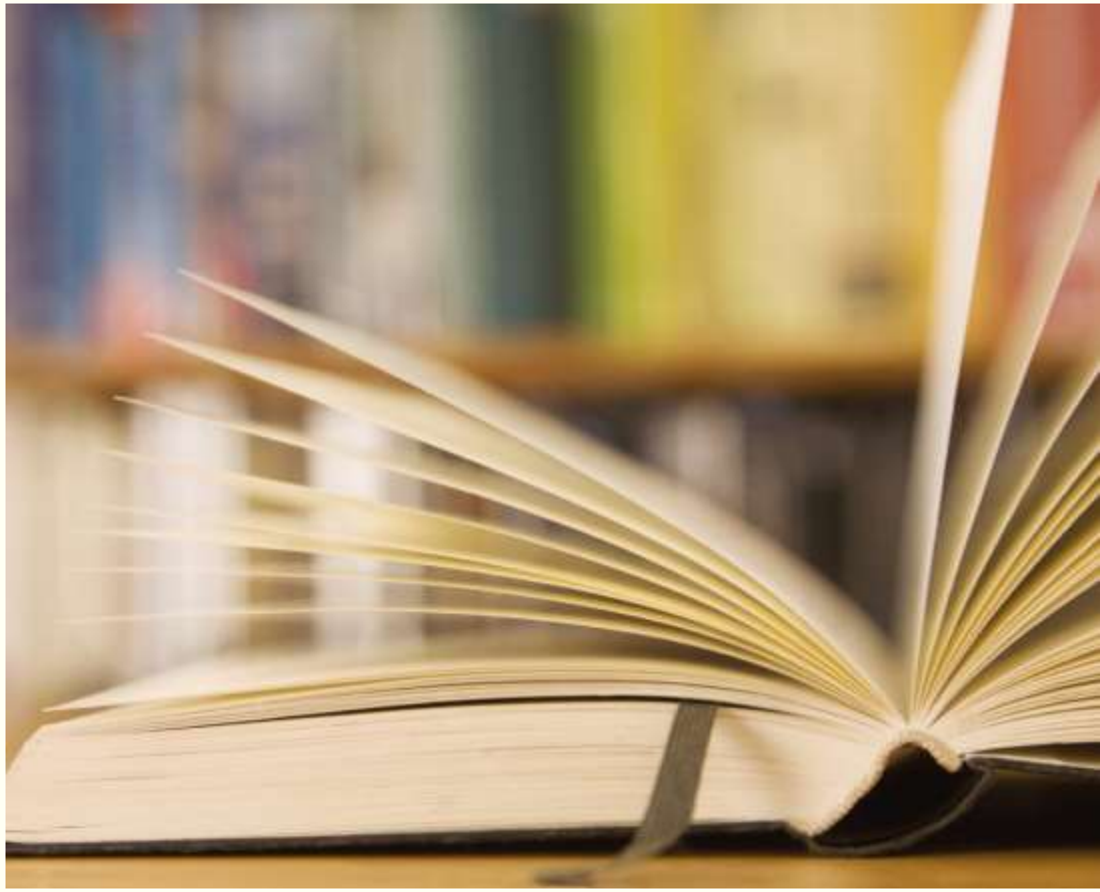
10. FILE STRUCTURE



FPT UNIVERSITY

Content

Objectives

After studying this chapter, the student should be able to:

- Define two categories of access methods: sequential access and random access.
- Understand the structure of sequential files and how they are updated.
- Understand the structure of indexed files and the relation between the index and the
- data file.
- Understand the idea behind hashed files and describe some hashing methods.
- Describe address collisions and how they can be resolved.
- Define directories and how they can be used to organize files.
- Distinguish between text and binary files.

1 - ACCESS METHODS

1. Introduction

- Files are stored on auxiliary or secondary storage devices. The two most common forms of secondary storage are disk and tape. Files in secondary storage can be both read from and written to. Files can also exist in forms that the computer can write to but not read. For example, the display of information on the system monitor is a form of file, as is data sent to a printer. In a general sense, the keyboard is also a file, although it cannot store data.

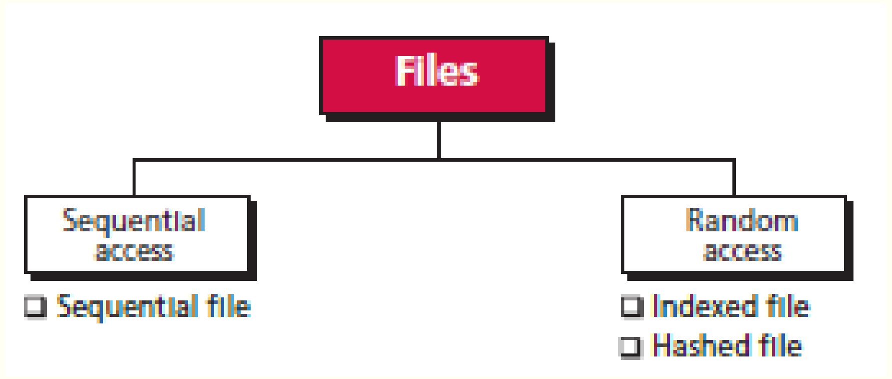


Figure 10.1 *A taxonomy of file structures*

2. SEQUENTIAL FILES

- A sequential file is one in which records can only be accessed one after another from beginning to end. Figure 13.2 shows the layout of a sequential file. Records are stored one after another in auxiliary storage, such as tape or disk, and there is an EOF (end-of-file) marker after the last record.

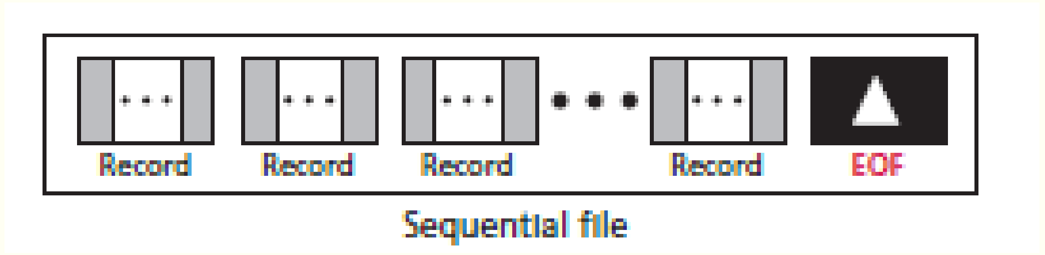
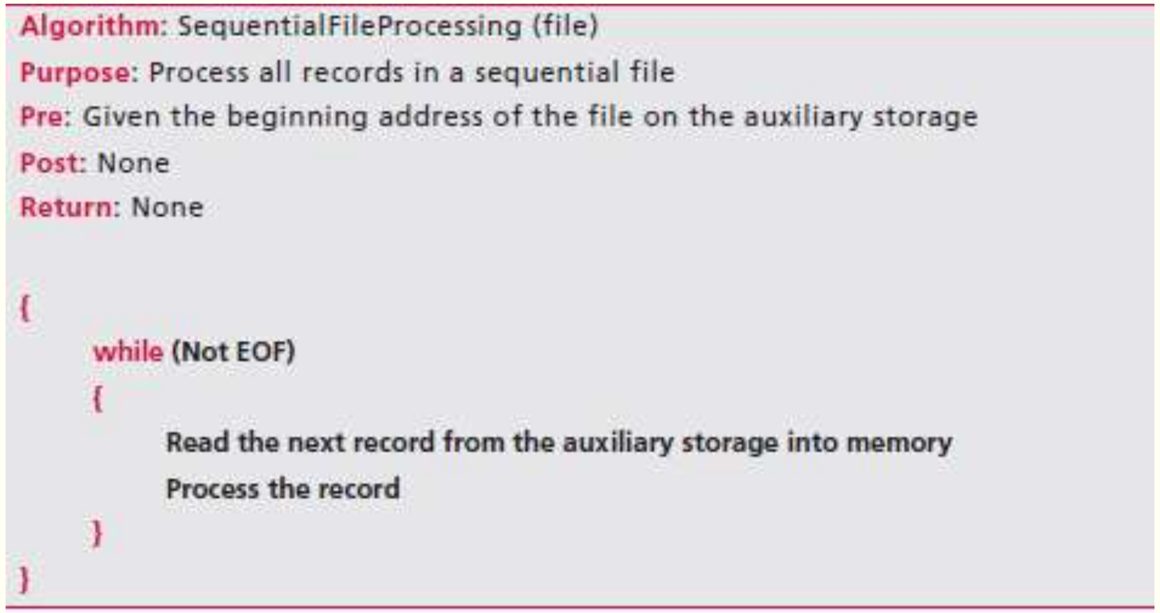


Figure 10.2 *A sequential file*

Example

- Algorithm 10.1 shows how records in a sequential file are processed. We process the records one by one. After the operating system processes the last record, the EOF is detected and the loop is exited.



```
Algorithm: SequentialFileProcessing (file)
Purpose: Process all records in a sequential file
Pre: Given the beginning address of the file on the auxiliary storage
Post: None
Return: None

{
    while (Not EOF)
    {
        Read the next record from the auxiliary storage into memory
        Process the record
    }
}
```

Algorithm 10.1 Pseudocode for processing records in a sequential file

3. INDEXED FILES

- To access a record in a file randomly, we need to know the address of the record. For example, suppose a customer wants to check their bank account. Neither the customer nor the teller knows the address of the customer's record. The customer can only give the teller their account number (key). Here, an indexed file can relate the account number (key) to the record address (Figure 13.5).

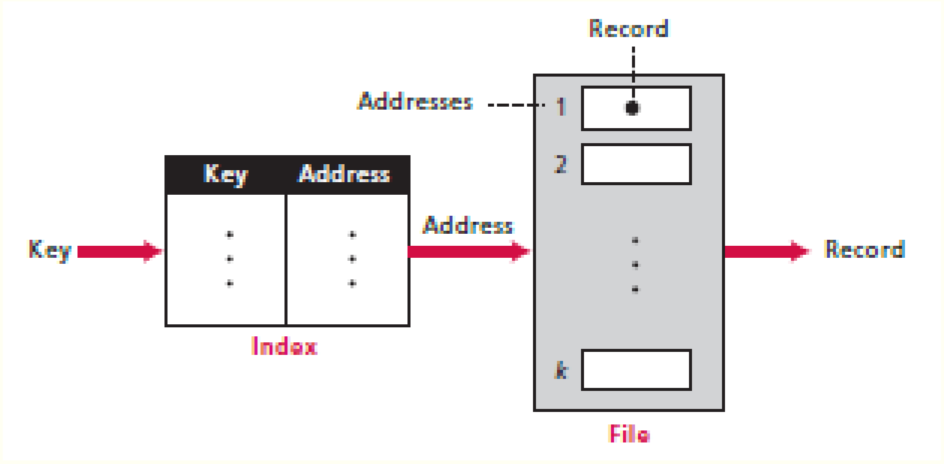
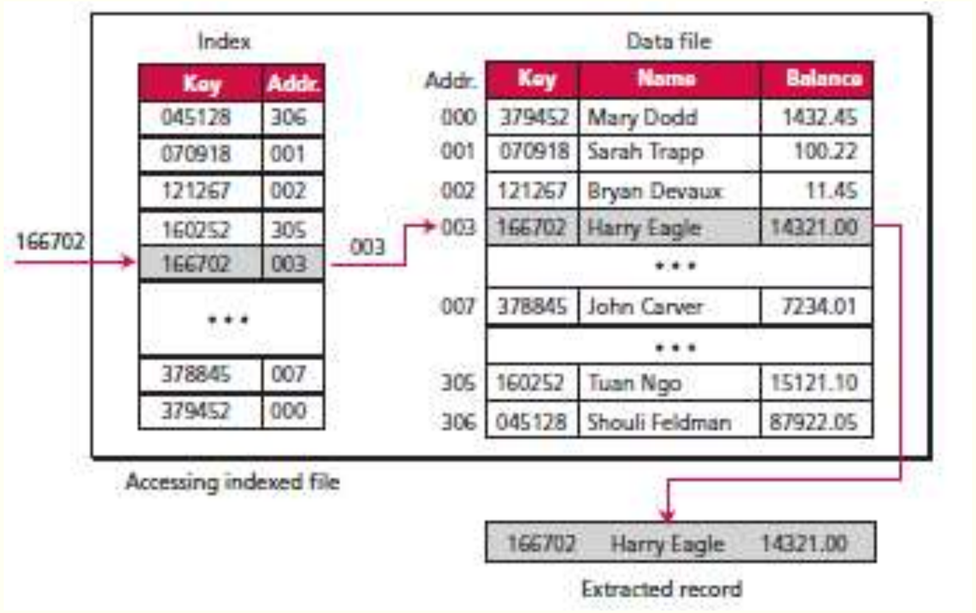


Figure 10.3 *Mapping in an indexed file*



Key	Addr.
045128	306
070918	001
121267	002
160252	305
166702	003
• • •	
378845	007
379452	000

Addr.	Key	Name	Balance
000	379452	Mary Dodd	1432.45
001	070918	Sarah Trapp	100.22
002	121267	Bryan Devaux	11.45
003	166702	Harry Eagle	14321.00
	• • •		
007	378845	John Carver	7234.01
	• • •		
305	160252	Tuan Ngo	15121.10
306	045128	Shouli Feldman	87922.05

166702	Harry Eagle	14321.00

Figure 10.4 *Logical view of an indexed file*

4. HASHED FILES

- In an indexed file, the index maps the key to the address. A hashed file uses a mathematical function to accomplish this mapping. The user gives the key, the function maps the key to the address and passes it to the operating system, and the record is retrieved (Figure 10.5).

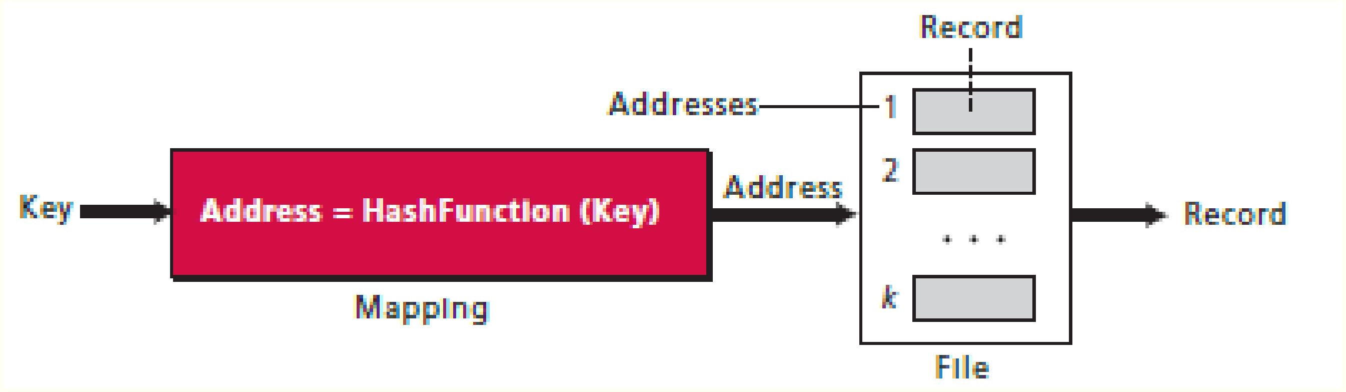
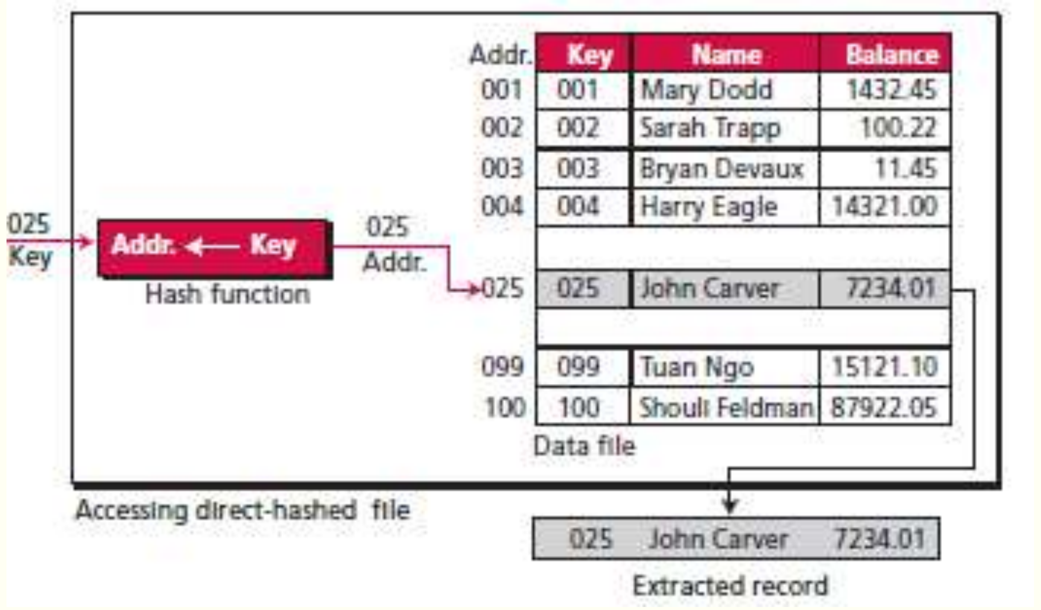


Figure 10.5 *Mapping in a hashed file*

Hashing methods

- ***Direct hashing***

- In **direct hashing**, the key is the data file address without any algorithmic manipulation. The file must therefore contain a record for every possible key. Although situations suitable for direct hashing are limited, it can be very powerful because it guarantees that there are no *synonyms* or *collisions* (discussed later in this chapter), as with other methods



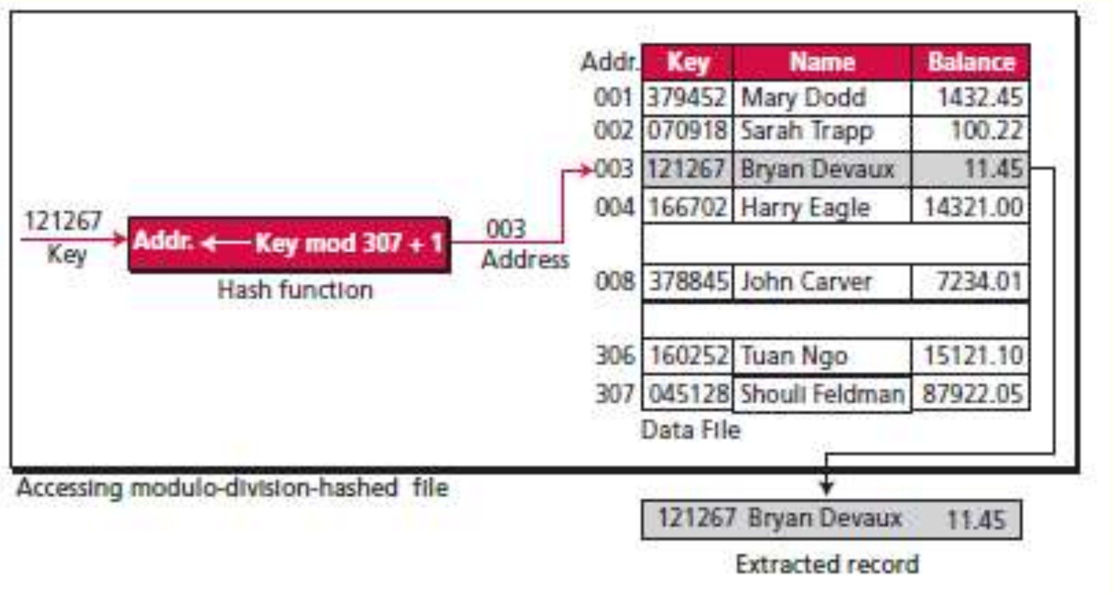
Addr.	Key	Name	Balance
001	001	Mary Dodd	1432.45
002	002	Sarah Trapp	100.22
003	003	Bryan Devaux	11.45
004	004	Harry Eagle	14321.00
025	025	John Carver	7234.01
099	099	Tuan Ngo	15121.10
100	100	Shouli Feldman	87922.05

Figure 10.6 *Direct hashing*

Hashing methods (cont)

- ***Modulo division hashing***

- Also known as **division remainder hashing**, the **modulo division** method divides the key by the file size and uses the remainder plus 1 for the address. This gives the simple hashing algorithm that follows, where *list_size* is the number of elements in the file. The reason for adding a 1 to the mod operation result is that our list starts with 1 instead of 0:



Addr.	Key	Name	Balance
001	379452	Mary Dodd	1432.45
002	070918	Sarah Trapp	100.22
003	121267	Bryan Devaux	11.45
004	166702	Harry Eagle	14321.00
008	378845	John Carver	7234.01
306	160252	Tuan Ngo	15121.10
307	045128	Shouli Feldman	87922.05

Figure 10.7 *Modulo division*

5. Collision resolution

- When we hash a new key to an address, we may create a collision.
- There are several methods for handling collisions, each of them independent of the hashing algorithm. That is, any hashing method can be used with any collision resolution method

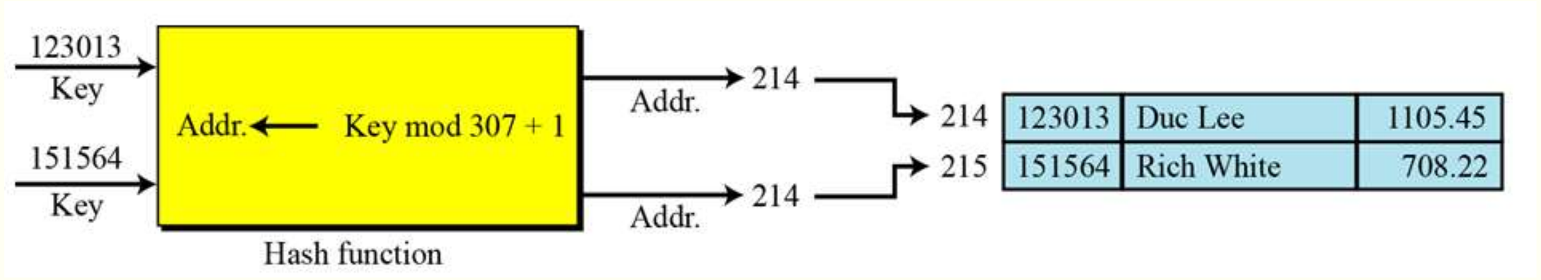


Figure 13.11 *Open addressing resolution*

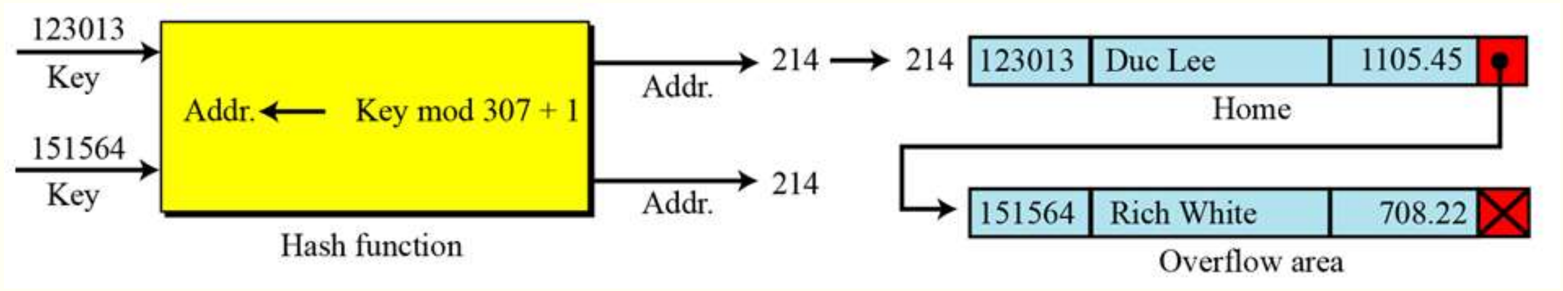


Figure 10.8 *Linked list resolution*

2- TEXT VERSUS BINARY

1. TEXT VERSUS BINARY

- Two terms used to categorize files: text files and binary files. A file stored on a storage device is a sequence of bits that can be interpreted by an application program as a text file or a binary file, as shown in Figure 13.15.

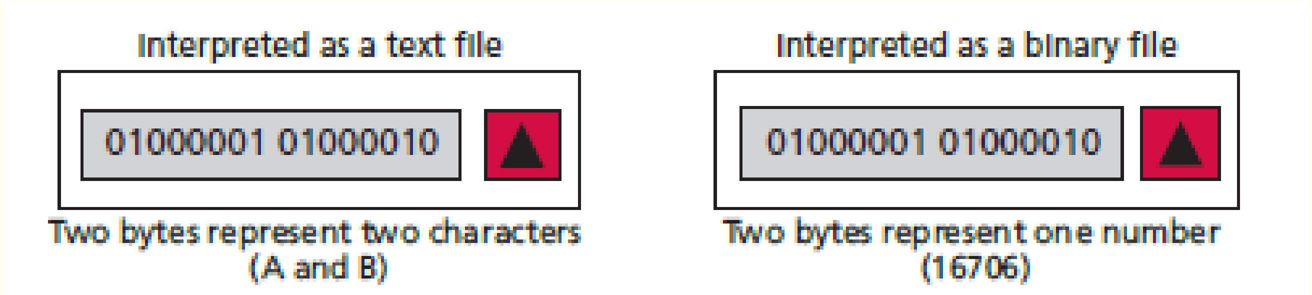
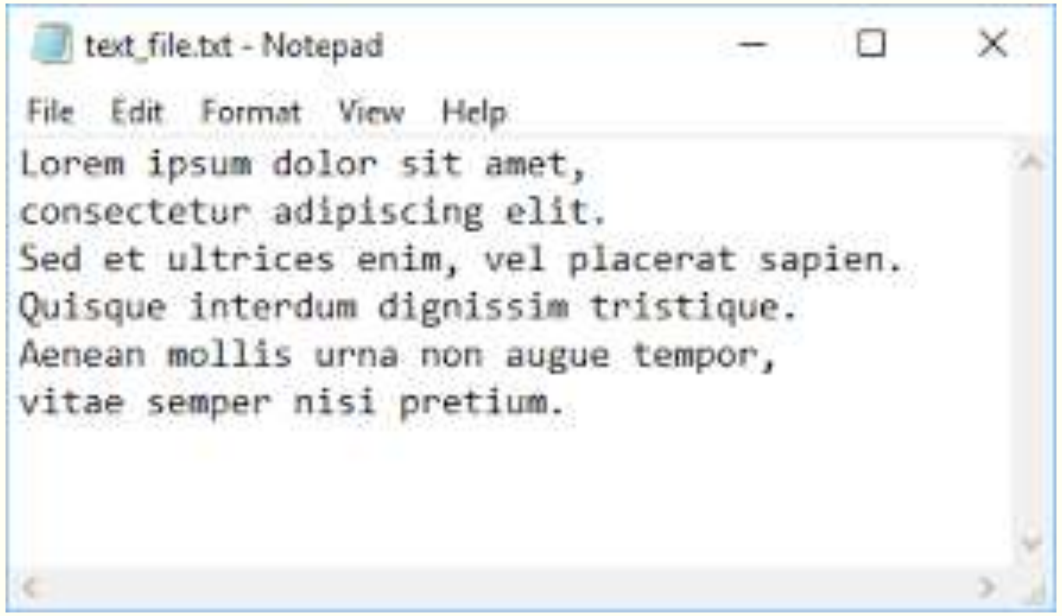

Interpreted as a text file
01000001 01000010
Two bytes represent two characters (A and B)
Interpreted as a binary file
01000001 01000010
Two bytes represent one number (16706)

Figure 10.9 *Text and binary interpretations of a file*

2. Text File

- A **text file** is a file of characters. It cannot contain integers, floating-point numbers, or any other data structures in their internal memory format. To store these data types, they must be converted to their character equivalent formats.

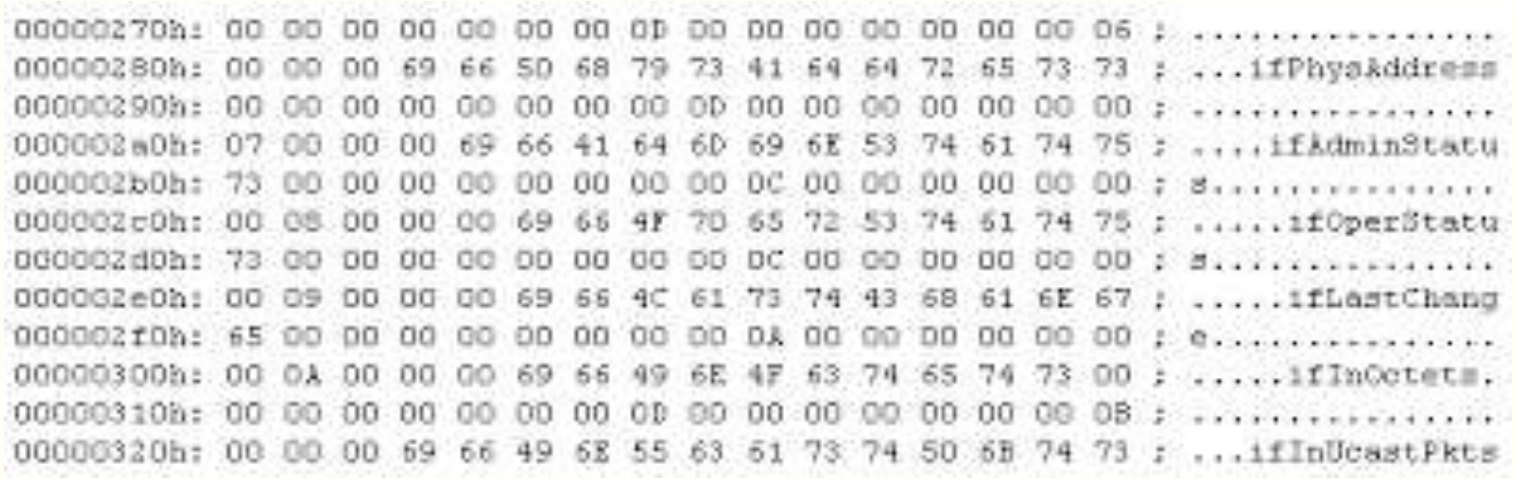


```
Lorem ipsum dolor sit amet,
consectetur adipiscing elit.
Sed et ultrices enim, vel placerat sapien.
Quisque interdum dignissim tristique.
Aenean mollis urna non augue tempor,
vitae semper nisi pretium.
```

Figure 10.10 *Text file*

3. Binary files

- A **binary file** is a collection of data stored in the internal format of the computer. In this definition, data can be an integer (including other data types represented as unsigned integers, such as image, audio, or video), a floating-point number, or any other structured data (except a file).



```
00000270h: 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 06 ; ................
00000280h: 00 00 00 69 66 50 68 79 73 41 64 64 72 65 73 73 ; ...ifPhysAddress
00000290h: 00 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 ; ................
000002a0h: 07 00 00 00 69 66 41 64 6D 69 6E 53 74 61 74 75 ; ....ifAdminStatu
000002b0h: 73 00 00 00 00 00 00 00 00 0C 00 00 00 00 00 00 ; s...............
000002c0h: 00 08 00 00 00 69 66 4F 70 65 72 53 74 61 74 75 ; .....ifOperStatu
000002d0h: 73 00 00 00 00 00 00 00 00 DC 00 00 00 00 00 00 ; s...............
000002e0h: 00 09 00 00 00 69 66 4C 61 73 74 43 68 61 6E 67 ; .....ifLastChang
000002f0h: 65 00 00 00 00 00 00 00 00 DA 00 00 00 00 00 00 ; e...............
00000300h: 00 0A 00 00 00 69 66 49 6E 4F 63 74 65 74 73 00 ; .....ifInOctets.
00000310h: 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 0B ; ................
00000320h: 00 00 00 69 66 49 6E 55 63 61 73 74 50 6B 74 73 ; ...ifInUcastPkts
```

Figure 10.11 *Binary file*

4. DIRECTORIES

- Directories are provided by most operating systems for organizing files. A directory performs the same function as a folder in a filing cabinet. However, a directory in most operating systems is represented as a special type of file that holds information about other files.

- A directory not only serves as a kind of index that tells the operating system where files are located on an auxiliary storage device, but can also contain other information about the files it contains, such as who has access to each file, or the date when each file was created, accessed, or modified.

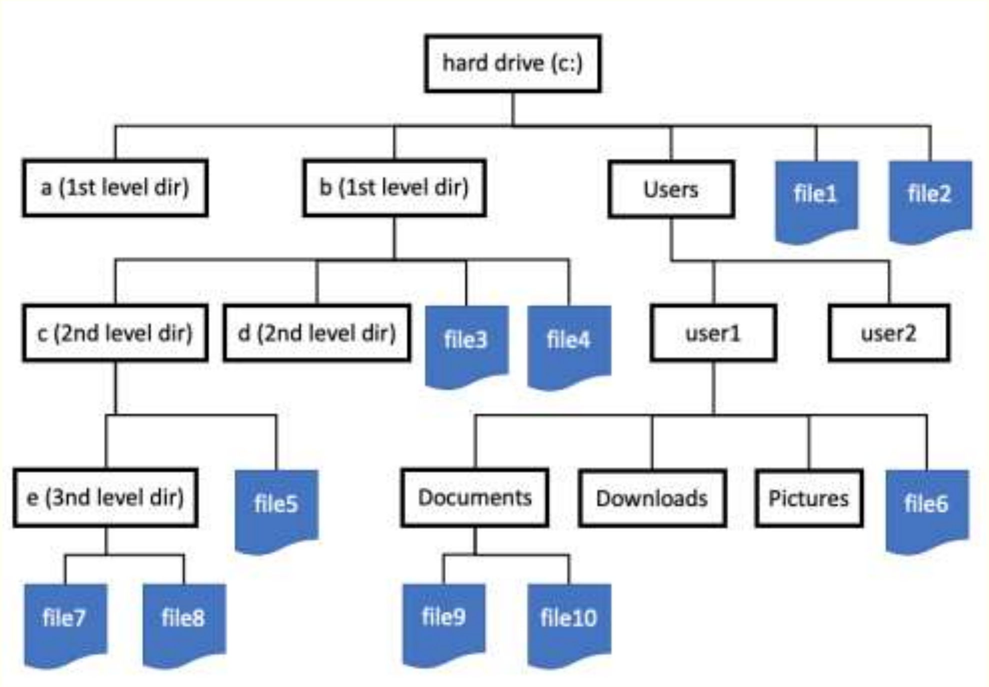


Figure 10.12 Directories in Window