OPERATING SYSTEMS

Topic – File management

File Concept

Computers can store information on various non-volatile storage media, such as

- magnetic disks
- magnetic tapes
- optical disks
- Solid State Disk (SSD)

The operating system provides a uniform logical view of stored information and abstracts from the physical properties of its storage devices to define a logical storage unit, the **file.**

Files are mapped by the operating system onto physical devices. These storage devices are usually *nonvolatile*, *so the contents are persistent between system reboots*.

File Concept (cont.)

- A File is a named collection of related information that is recorded on secondary storage.
- A File is the smallest allotment of logical secondary storage
- Data can be written to secondary storage only within a file.

The information in a file is defined by its creator. Many different types of information may be stored in a file:

- ✓ source or executable programs,
- ✓ numeric or text data,
- ✓ photos, music,
- ✓ video, and so on.

A file has a certain defined structure, which depends on its type.

File Attributes

A file's attributes vary from one operating system to another but typically consist of these:

Name: The symbolic file name is the only information kept in human readable form.

Identifier: This unique tag, usually a number, identifies the file within the file system; it is the non-human-readable name for the file.

Type: This information is needed for systems that support different types of files.

Location: This information is a pointer to a device and to the location of the file on that device.

Size: The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed size are included in this attribute.

Protection: Access-control information determines who can do reading, writing, executing, and so on.

Time, date, and user identification: This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

Operations on Files

A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files. The operating system can provide system calls to create, write, read, reposition, delete, and truncate files.

Creating a file: Two steps are necessary to create a file. First, space in the file system must be found for the file. Second, an entry for the new file must be made in the directory.

Writing a file: To write a file, a system call is made by specifying both the name of the file and the information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.

Operations on Files (cont.)

- **Reading a file:** To read from a file, we use a system call is used by specifying the name of the file and where (in memory) the next block of the file should be put. Again, the directory is searched for the associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated. Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current file-position pointer. Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning within a file: The directory is searched for the appropriate** entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file **seek.**

Operations on Files(cont.)

Deleting a file: To delete a file, the named file searched in the directory for. After finding the associated directory entry, all file space is released (so that it can be reused by other files) and finally the directory entry is released.

Truncating a file. The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged—except for file length—but lets the file be reset to length zero and its file space released

File Types

‘ *Whether the operating system should recognize and support* ***file types***’ should be considered while designing a file system—indeed, an entire operating system.

If an operating system recognizes the type of a file, it can then operate on the file in reasonable ways.

For example, a common mistake occurs when a user tries to output the binary-object form of a program. This attempt normally produces garbage; however, the attempt can succeed if the operating system has been told that the file is a binary-object program.

A common technique for implementing file types is to include the type as part of the file name. The name is split into two parts—a name and an extension, usually separated by a period. The system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

Common File Types

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, perl, asm	source code in various languages
batch	bat, sh	commands to the command interpreter
markup	xml, html, tex	textual data, documents
word processor	xml, rtf, docx	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	gif, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	rar, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, mp3, mp4, avi	binary file containing audio or A/V information

File Types (cont.)

The operating system uses the extension to indicate the type of the file and the type of operations that can be done on that file.

Application programs also use extensions to indicate file types in which they are interested. For example, Java compilers expect source files to have a .java extension, and the *Microsoft Word* processor expects its files to end with a .doc or .docx extension.

UNIX does allow file-name-extension hints, but these extensions are neither enforced nor depended on by the operating system; they are meant mostly to aid users in determining what type of contents the file contains. Extensions can be used or ignored by a given application, but that is up to the application's programmer

Sequential Access Method

Files store information and this information must be accessed and read into computer memory. The information in the file can be accessed in several ways.

Sequential Access (based on magnetic tape model)

In sequential access method, the information in the file is processed in order, one record after the other; for example, editors and compilers usually access files in this fashion.

Read Operation: read next() reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location

Write Operation: write next() appends to the end of the file and advances to the end of the newly written material

A file pointer can be reset to the beginning, and may be able to skip forward or backward *n records for some integer n*

Direct Access Method

Direct Access (based on disk model of a file)

Here, a file is made up of fixed-length logical records that allow programs to read and write records rapidly in no particular order. For direct the file is viewed as a numbered sequence of blocks or records and there are no restrictions on the order of reading or writing for a direct-access file.

Direct-access files are of great use for immediate access to large amounts of information. In Databases Applications, when a query arrives, we compute which block contains the answer and then read that block directly to provide the desired information.

In direct-access method, the file operations must be modified to include the block number as a parameter. Here the Read and Write Operations are read(n) and write(n), where n is the block number, and it is a relative block number (relative to the beginning of the file)

The sequential access can be easily simulated on a direct-access file, however, is extremely inefficient and clumsy.

Other Access Methods

Other Access Methods

Other access methods can be built on top of a direct-access method. These methods generally involve the construction of an index for the file. The i**ndex** contains pointers to the various blocks. To find a record in the file, we first search the index and then use the pointer to access the file directly and to find the desired record.