22516 Operating System

Unit-VI File Management

6.1 File - Concepts, Attributes, Operations, types and File System Structure.

File in OS

In an operating system, a file is a named collection of related data that appears to the user as a single, contiguous block of information and that is retained in storage. Files are the means by which data is stored and organized on a computer.

Here are some key concepts related to files in an operating system:

1. **File Structure**: A file is generally organized in a particular structure to make it easier to access and manage the data. The structure can be as simple as a stream of bytes (a text file), or it can be complex, like a structured database.
2. **File Type**: The type of a file is typically indicated by its extension (for example, .txt for a text file, .jpg for a JPEG image file). The file type often determines how the file is interpreted by the operating system or applications.
3. **File Operations**: The operating system provides a variety of operations that can be performed on files, including creating, reading, writing, deleting, and renaming files.
4. **File Attributes**: Files typically have associated metadata, known as attributes, that include information like the file size, creation date, permissions, and the location of the file's data on disk.
5. **File System**: The file system is the part of the operating system that is responsible for creating, deleting, reading, and writing files. It organizes the files into directories (also known as folders), which can contain files and other directories. File systems also manage free space on the disk and provide access control to files and directories.
6. **File Paths**: Files are located within the file system by their path, which shows the directories one must traverse to reach the file.
7. **Access Control**: Operating systems implement access control mechanisms to control who can access a file and what they can do with it. This can include read, write, and execute permissions.

In summary, files are a crucial part of any operating system, enabling users and applications to store and retrieve data in a persistent and organized manner.

File Attributes in OS

File attributes are metadata that provide important information about a file. The operating system uses these attributes to manage and organize files effectively. While the specific attributes may vary between different operating systems, common file attributes include:

1. **Name**: The name of the file, typically including a file extension that indicates the file type (e.g., .txt for text files, .jpg for JPEG images).
2. **Size**: The size of the file, usually measured in bytes.
3. **Type**: This attribute specifies the format of the file's content. This can be a text, executable, image, audio, etc.

4. **Location**: The location or address of the file in the system's storage. This can refer to the directory path where the file is located or the specific blocks on the disk where the file's data is stored.
5. **Creation Time**: The date and time when the file was created.
6. **Modification Time**: The date and time of the most recent changes made to the content of the file.
7. **Access Time**: The date and time when the file was last accessed or read.
8. **Permissions**: The access rights that determine who can read, write, or execute the file. These are often broken down into permissions for the owner of the file, a group, and all users.
9. **Owner**: The user or system account that owns the file.
10. **Hidden Flag**: A flag that, if set, indicates that the file should not be displayed in a standard file listing.
11. **Read-Only Flag**: A flag that, if set, indicates that the file can be read but not modified.

These attributes allow the operating system and users to manage and manipulate files effectively. Tools within the operating system allow users to view and sometimes modify these attributes. For instance, in Unix-based systems, the 'ls -l' command displays many of these attributes, and the 'chmod' command can change the permissions attribute.

File Operations in OS

File operations in an operating system refer to the various actions that can be performed on files. These operations allow users and applications to create, read, write, update, delete, and manipulate files. Here are the common file operations in an operating system:

1. **Creating a File**: This operation involves the creation of a new file with a given name and file type. The operating system allocates space in the file system to store the file's data and assigns initial attributes, such as size and permissions.
2. **Opening a File**: To access the contents of a file, it must be opened. When a file is opened, the operating system sets up a file control block (FCB) or file descriptor, which keeps track of the file's attributes and current position within the file.
3. **Reading from a File**: The read operation allows data to be retrieved from a file. The operating system reads a specified number of bytes from the file and moves the file pointer to the next position.
4. **Writing to a File**: The write operation enables data to be written into a file. The operating system writes a specified number of bytes into the file and moves the file pointer accordingly.
5. **Updating a File**: Updating a file typically involves modifying the content of an existing file. This is achieved through a combination of reading and writing operations.
6. **Closing a File**: After finishing the operations on a file, it must be closed to release any resources associated with it. Closing a file ensures that any changes made are saved to disk and that the file is no longer accessible until it's opened again.
7. **Renaming a File**: This operation allows the file to be given a new name.
8. **Deleting a File**: The delete operation permanently removes the file from the file system, freeing up the allocated space.
9. **Seeking in a File**: The seek operation sets the file pointer to a specific position within the file, allowing random access to different parts of the file.

File operations are crucial for managing data in an operating system, and they are essential for the functioning of applications and user interactions with files. These operations are facilitated by the file system, which is responsible for organizing and managing files on storage devices.

File types in OS

File types in an operating system generally refer to the format of a file, which typically dictates how the file is stored and used. The type of a file is often indicated by its extension, a set of characters appended to the end of the file name.

Here are some common file types:

1. **Text Files (.txt, .doc, .docx, .rtf, .csv, .xml)**: These files primarily contain human-readable text. They can be opened by text editors or word processors.
2. **Image Files (.jpg, .png, .gif, .bmp, .tiff)**: These files store images in various formats, each of which may use different compression techniques and support different features such as transparency or animation.
3. **Audio Files (.mp3, .wav, .ogg, .flac)**: These files contain sound data. They can be played using media player software.
4. **Video Files (.mp4, .avi, .mkv, .mov, .wmv)**: These files contain video data, often along with audio data. They can be played using media player software.
5. **Executable Files (.exe, .bin, .bat, .sh)**: These files contain instructions that can be executed directly by the computer. On Windows, these typically have a .exe extension, while on Unix-like systems they might not have an extension at all.
6. **Archive and Compressed Files (.zip, .rar, .tar, .gz)**: These files contain one or more other files and directories, often compressed to save space.
7. **System Files (.dll, .sys, .drv)**: These files are used by the operating system to perform various functions and are usually not directly used by the user.
8. **Database Files (.db, .sql, .mdb)**: These files store data in a structured format, which can be accessed and manipulated using database management software.
9. **Web Files (.html, .css, .js)**: These files are used in the development of web pages and websites. HTML files contain the structure of a webpage, CSS files control the appearance of the webpage, and JS files contain scripts to make the webpage interactive.

Different operating systems and software will handle different file types in their own unique ways. It's always important to ensure that the software necessary to handle a particular file type is available in your system.

File System Structure in OS

The file system structure in an operating system is the way files are organized and managed on storage devices, such as hard drives, SSDs, or memory cards. This structure allows for efficient storage, retrieval, and manipulation of files.

Here's an overview of a typical hierarchical file system structure, which is used in many modern operating systems like Windows, macOS, and Linux:

1. **Root Directory**: At the top of the file system is the root directory. On Unix-like operating systems, this is denoted by a forward slash ("/"), while in Windows, it's usually represented by a drive letter (e.g., C:). The root directory contains files and subdirectories.
2. **Subdirectories**: Each directory or subdirectory under the root directory can contain files and additional subdirectories. This structure can continue for several levels, creating a tree-like hierarchy of directories and subdirectories. This hierarchical organization helps keep files organized and makes it easier for users and applications to locate specific files.
3. **Files**: At the lowest level of the hierarchy are the individual files, each with a unique path that includes all the directories from the root to the file itself.

4. **Path**: The location of a file or directory is specified by its path, which represents the sequence of directories that must be traversed to reach the file or directory. Absolute paths start from the root directory (e.g., /home/user/documents/file.txt on Unix-like systems or C:\Users\Username\Documents\file.txt on Windows), while relative paths start from the current working directory.
5. **Permissions**: Each file and directory has associated permissions, which specify who can read, write, or execute the file or directory. These permissions can usually be set for three types of users: the owner of the file, the group associated with the file, and all other users.

File system structures can vary widely in their complexity and features. Some file systems are flat, with no subdirectories, while others are hierarchical, as described above. Some file systems support additional features, such as file versioning, encryption, or redundancy to protect against data loss. The specific file system used by an operating system can significantly impact performance and data management.

6.2 Access Methods - Sequential, Direct, Swapping, File Allocation Methods- Contiguous, Linked, Indexed.

File Access Methods in OS

File access methods in an operating system are the ways in which files can be accessed and read from or written to. These methods define how data is retrieved and stored, and the approach can vary based on the type of data and the specific requirements of the system.
Here are the main types of file access methods:

1. **Sequential Access**: In sequential access, files are read from or written to in a continuous sequence from the beginning to the end, much like a tape. To reach a particular point in a file, all preceding data must be read first. This method is simple and suitable for certain types of data, like audio or video files, which are typically accessed in a sequential manner.
2. **Direct Access (or Random Access)**: In direct access, files can be read from or written to at any point without traversing through other data first. It allows quick access to specific locations within a file and is often used with databases, where records need to be retrieved or updated quickly without reading through every preceding record.
3. **Indexed Access**: In indexed access, an index is created that references the locations of various blocks of data in the file. To read or write data, the operating system first consults the index to find the specific locations of data in the file. Indexed access can be a sort of combination of sequential and direct access, and it's particularly useful for large databases where direct access to specific data is needed, but the data also has some natural order.

These access methods have different advantages and trade-offs. Sequential access is straightforward and efficient for certain tasks, but it can be slow for large files where specific data needs to be accessed quickly. Direct and indexed access allow for quicker access to specific data but may require more complex management and increased overhead. The choice of file access method depends on the specific use case and system requirements.

File Swapping in OS

File swapping, also known as process swapping, or just swapping, is a method used by an operating system to manage memory and efficiently use system resources. It involves moving data or processes between the main memory and the disk when the main memory is full or when inactive data or processes need to be stored temporarily.
Swapping is used as part of memory management, particularly in systems that use virtual memory. When a process is to be executed, it is loaded into the main memory (RAM). However, RAM is limited. When it's full and the operating system needs to load another process into memory, the operating system may choose to move some inactive or low-priority processes from the RAM to a reserved space on the disk, called the swap space or page file. This frees up space in the main memory for the new process.
The swapped-out process remains in the swap space until it's needed again. At that point, the operating system may swap another process out of the main memory to make room to load the swapped-out process back into memory.
While swapping is a necessary function for many systems, especially those with limited memory resources, it can lead to decreased system performance if used excessively, as accessing data from disk is significantly slower than accessing it from the main memory. This situation is often referred to as "thrashing". Hence, efficient memory management techniques are important to minimize the need for swapping and maintain good system performance.

File Allocation Methods in OS

File allocation refers to the way files are stored on the disk by an operating system. The file allocation method determines how the disk blocks are allocated to files which affects disk space usage, file access speed, and disk performance.

There are three common file allocation methods:

1. **Contiguous Allocation**: In this method, each file occupies a contiguous set of blocks on the disk. This allows for efficient sequential and direct access as all blocks of a file are located together. However, it suffers from issues like external fragmentation and difficulty in file size estimation.
2. **Linked Allocation**: In linked allocation, each file is a linked list of disk blocks, which are not necessarily contiguous. Each block contains a pointer to the next block in the file. This method eliminates external fragmentation but it is inefficient for direct access as you have to traverse through the blocks sequentially following the pointers. This method is suitable for sequential access files.
3. **Indexed Allocation**: In indexed allocation, each file has its own index block which stores the addresses of the disk blocks that belong to the file. This method supports efficient direct access, and doesn't suffer from external fragmentation. However, it can be inefficient for small files, as an entire index block may be allocated for just a few blocks of data.

Some file systems use a combination of these methods. For example, Unix/Linux uses a combination of direct, singly indirect, doubly indirect, and trebly indirect pointers, which can be seen as a mix of linked and indexed allocation.

Each method has its own advantages and disadvantages and the choice of method often depends on the specific use case, file size, access patterns, and performance requirements.

6.3 Directory structure-- Single level, two levels, tree-structured directory, Disk Organization and disk Structure- Physical structure, Logical structure, Raid structure of disk, raid level Oto 6.

File Directory structure in OS

The file directory structure in an operating system represents the way files are organized and stored in the file system. It allows the user and the system to locate files and directories efficiently. The file directory structure is hierarchical, meaning it begins from a root and branches out into subdirectories.

Here are the key components of a typical file directory structure:

1. **Root Directory**: This is the top-level directory from which all other directories branch out. On Unix and Linux systems, the root directory is denoted by a slash (/). On Windows systems, the root directory is typically denoted by a drive letter (e.g., C:).
2. **Subdirectories**: Also known as child directories, these are directories that exist within other directories. A directory can contain multiple subdirectories, and each of those subdirectories can contain further subdirectories, and so on, forming a tree-like structure.
3. **Files**: These are the actual data elements stored within directories. Every file is associated with a directory. In most file systems, files are organized within directories and can be identified by a path that represents the sequence of directories leading to the file.
4. **Path**: The path to a file or directory provides an exact location in the file system. Paths can be absolute or relative. An absolute path starts from the root directory (e.g., /home/user/documents/file.txt on Unix/Linux or C:\Users\Username\Documents\file.txt on Windows). A relative path starts from the current directory.
5. **Parent Directory**: The parent directory is the directory above a subdirectory. For instance, if we have a directory path /home/user/documents, "user" is the parent directory of "documents".

The file directory structure makes it possible to organize files in a way that reflects their relationships and importance. This not only facilitates efficient file storage and retrieval but also simplifies file management tasks such as searching for files and backing up data.

Single level, two levels, tree-structured directory, Disk Organization and disk Structure Physical structure, Logical structure, Raid structure of disk, raid level Oto 6 in OS

Disk Organization and Disk Structure:

1. **Physical Structure**: The physical structure of a disk refers to how data is physically stored on the disk's surface. The disk is divided into concentric circles called tracks, and each track is further divided into sectors. The combination of a track and a sector is called a disk block, which is the smallest unit of data that can be read from or written to the disk.
2. **Logical Structure**: The logical structure of a disk refers to how the operating system organizes and manages data on the disk. The logical structure includes the file system, which is responsible for managing files and directories. The file system maintains a mapping between the logical addresses of files and directories and the physical addresses on the disk.

RAID (Redundant Array of Independent Disks):

RAID is a storage technology that uses multiple disks to improve performance, reliability, and fault tolerance. There are several RAID levels, each offering different benefits:

1. **RAID 0 (Striping)**: Data is striped across multiple disks, which improves performance by allowing multiple disks to work in parallel. However, there is no redundancy, so if one disk fails, data loss occurs.

2. **RAID 1 (Mirroring)**: Data is mirrored across two disks, providing redundancy. If one disk fails, data can be recovered from the other disk.

3. **RAID 2**: This level is rarely used, as it involves bit-level striping and error correction using Hamming code.

4. **RAID 3**: Data is striped at the byte level across multiple disks, with one dedicated disk for parity. Provides good performance for large sequential transfers but limited parallelism for small transfers.

5. **RAID 4**: Data is striped at the block level across multiple disks, with one dedicated disk for parity. Offers better random read performance compared to RAID 3.

6. **RAID 5**: Data and parity are distributed across all disks, providing better fault tolerance than RAID 3 and RAID 4. RAID 5 is widely used due to its good balance of performance and redundancy.

The choice of RAID level depends on the specific requirements of the system, such as performance, capacity, and fault tolerance needs. Each RAID level offers different trade-offs between performance and data redundancy.

DIPLOMA SOLUTION
"DIPLOMA SOLUTION"
Connect with Us...
+91 8108332570
MSBTE Diploma Solution
www.diplomasolution.com