**Georgia Southern University**

**Allen E. Paulson College of Engineering and Computing**

**Information Technology Department**

**Dr. Atef Shalan**

**Course: IT2530-Operating Systems**

**Module 6-** File System Management– reading material

Contents and Reading Material

[**6** File System Management 1](#_Toc90319432)

[6.1 File System Overview 1](#_Toc90319433)

[6.2 File System Architecture 2](#_Toc90319434)

[6.3 File system types 3](#_Toc90319435)

[6.4 Directory structure 3](#_Toc90319436)

[6.5 File sharing 4](#_Toc90319437)

[6.6 File allocation 4](#_Toc90319438)

# File System Management

## File System Overview

The function of computer systems is tightly dependent on data processing. In computers, files provide identification and isolation of data chunks. [A file system](https://www.guru99.com/file-systems-operating-system.html) is a data structure and management method that allows an operating system to control data storage and retrieval. Files are structured containers of data that can have long-term existence on secondary storage devices. Files can be also accessed and shared among multiple processes.

Each file has a number of attributes uniquely identifying and describing the file identification and contents status. Some of these attributes are the following.

* **Filename:** an identifier describing the file and that is readable to system users.
* **File type:** the type of the file, e.g., executable, text data, image, video, etc.
* **File path**: location of the file in the file system structure.
* **Access position**: a pointer to the current read/write content part.
* **Storage location**: a hard disk position where the file is stored.
* **File size**: the number of bytes in the file.
* **File access protection**: Access control specification of the read, write, execute, and delete operations.
* **File usage information**: time of creation, last modified, etc.

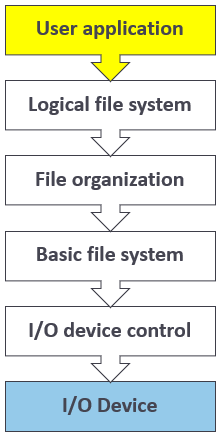
File systems provide interfaces to operating systems to handle files. The following are some operations on files include:

* **Create a new file**: allocate disk space on disk, and store the file attributes and content.
* **Open a file**: enable handlers to the file contents for reading or writing.
* **Close a file**: remove file handlers and terminate current access operations.
* **Write to file**: add new data to the file
* **Read from file**: read file contents to memory.
* **Delete file:** remove file identification attributes, and regain disk space.

## File System Architecture

Operating Systems use a layered structure for file management where every layer is responsible for some activities. Every operating system varies with respect to the layers of its file system architecture. Most operating systems manage the file management functions into the six layers of [file system architecture](https://www.javatpoint.com/os-file-system-structure) shown in Figure 1.

Figure - File system architecture



* **User application** 
  + File access services are provided at the application interface level to users
* **Logical file system**
  + A standard interface between applications and the file systems
  + Provides the application program interface (API) for high-level file access operations, e.g., OPEN, CLOSE, READ, etc.
  + Controls file access permissions and operations by users
  + Maintains file attributes and metadata
* **File organization**
  + Implements access methods for different file structures and data processing.
  + Maps logical file blocks to physical file blocks based on the file organization structure.
  + Provides storage space management.
  + Passes requests of physical blocks to the basic file system.
* **Basic file system**
  + A part of the operating system and AKA physical I/O level.
  + Provides file I/O initiation and termination
  + Maintains the I/O buffering blocks in main memory
  + Maintains control structures disk scheduling
  + Provide instructions to the I/O control layer to fetch the required blocks.
* **I/O device control**
  + Contains the low-level codes to access the hard disk.
  + Handling device I/O interrupts.
  + Processes physical blocks for read/write operations.
* **I/O Device**
  + Communicates directly with peripheral devices
  + Processes the completion of I/O requests

## File system types

There are several categories of file systems based on the device types and the purpose of the file system. Examples of file system types are as follows.

* **Disk drive file systems**
  + Supports random access and short access time.
  + Allow shared access to multiple processes
  + Examples: File Allocation Table (FAT) and New Technology File System (NTFS)
* **Tape drive file systems**
  + Manages data storage on long reels of magnetic media
  + No supports for random access and only sequential access is provided
* **Database file systems**
  + Supports database structures: field, record, data file, database.
  + File structures provide support for fast data retrieval and processing
* **Network file system**
  + Supports remote file access to clients
  + Utilizes network protocol for remote access, e.g., Network File System (NFS) and File Transfer Protocol (FTP) protocols.
* **Other file system categories**
  + **Flash memory file systems**, e.g., flash file systems
  + **Optical disk file systems**, e.g., ISO-13346 Universal Disk Format file system for Digital Video Disc (DVD) and UDF2.5 file system for Blu-ray disks.
  + **Special device file systems**, e.g., device file system (devfs), and universal device (udev).

## Directory structure

In computers, files are organized in structured containers called directories (AKA folders). File systems use these folders to group files together in separate containers. Depending on the storage device and file system design, [folder structures](https://www.geeksforgeeks.org/file-systems-in-operating-system/) can be flat (i.e., linear or array-like structure) or multilevel (or hierarchies) where folders may contain nested folders (AKA subfolders). Most disk file systems support hierarchical directory structures, e.g., [FAT, HFS, NTFS](https://docs.microsoft.com/en-us/troubleshoot/windows-client/backup-and-storage/fat-hpfs-and-ntfs-file-systems), etc. In hierarchical directory structures, the top-most directory is the only directory that does not have a parent and it is usually called the root directory. File location in the folder hierarchy is specified by a file attribute called file path. A file path includes a list of all subfolders where the files exist starting from the root. [Main operations on directories](https://www.geeksforgeeks.org/file-systems-in-operating-system/) include the following.

* **Search** for a file/a directory
* **Create** a file/a directory
* **Delete** a file/a directory
* **Rename** a file/a directory

## File sharing

Operating systems’ [file sharing](https://en.wikipedia.org/wiki/File_sharing) allows multi-user and multi-process access to computer programs, data files, digital media, documents, etc. File systems attributes include information about the access control of files and directories. File access control information includes a list of access rights ([access permissions](https://en.wikipedia.org/wiki/File-system_permissions#Classes)) to be granted for/or revoked from different users. Operating systems vary with respect to the list of access rights associated with files. When the access right of a file is set to user, the user will be able to make the type of access associated with this access right. When permission is not set, the corresponding rights are denied. Most operating systems provide file access control through the following list of access rights.

* **Read**: a file read permission grants the ability to read a file.
* **Execute:** a file execute permission grants the ability to run the file.
* **Write**: a file write permission grants the ability to modify a file.

When set to a directory the above access permissions will be interpreted to a different meaning. A directory read permission grants access to read the names of files in the directory. A directory execute permission allows users to actually enter that directory (e.g., using cd command) but not be able to read its contents, see what files are located there. The write permission of a directory grants the ability to create new files or modify files in that directory if specific file access permission is granted to users. Therefore,  without the execute permission, the write permission is meaningless for directories. In most operating systems access rights are granted at the following levels.

* **Owner**: the owner (usually the creator or high administrator) of the file.
* **Specific user**: a system user, application, or process.
* **Specific user group**: A group name with specific privileges that applies to all associated users.
* **Everyone**: some access rights are granted to everyone in the system.

In Linux, read, write, and execute permission are specified at the level to the owner user, ownership group, and all other users.

In operating systems, simultaneous access allows multiple users (i.e., processes) to access a file at the same time. In most operating systems, simultaneous access is provided using a [file locking mechanism](https://en.wikipedia.org/wiki/File_locking) that restricts access to a file or directory so that only one user or process is allowed to modify or delete the file at a specific time. The file locking mechanism also and prevents the reading of the file while it's being modified.

## File allocation

Operating systems organize file storage in secondary storage devices in physical blocks. The distribution of physical blocks of a single file in the secondary storage device is called [fille allocation](https://www.geeksforgeeks.org/file-systems-in-operating-system/) (or file blocking). There are three methods of blocking that can be used:

* **Contiguous Allocation** 
  + A file is allocated a number of contiguous blocks.
  + The File Allocation Table (FAT) system records the starting block and the length of the file in blocks.
  + **Advantages**
    - Reading the whole file does not require head movement.
  + **Disadvantage** 
    - External fragmentation problem, i.e., enough storage may exist for new file allocation but the available storage is not contiguous. A compaction algorithm is necessary to free combine the available free blocks (or gaps).
    - Pre-allocation is necessary to declare the size of the file at the time of creation.
* **Linked (Non-contiguous) allocation**
  + In linked allocation, blocks do not need to be contiguous.
  + Chained allocation is used by linking each block to the next block in the chain using pointer variables.
  + Like contiguous allocation, the FAT system records the starting block and the length of the file in blocks.
  + **Advantages**
    - This scheme allows allocating blocks and increasing file size as needed.
    - No external fragmentation problem
  + **Disadvantage**
    - Overhead of chaining
    - Internal fragmentation may exist in the last block of every file.
* **Indexed allocation**
  + the most popular form of file allocation
  + The allocated file blocks are listed in an index block
  + Indexed allocation allows using indexes of single blocks or variable-sized blocks. The index of variable-sized blocks refers to the index of a first block in a sequence of chained blocks.
  + No external fragmentation
  + Less overhead to maintain the block index information located in a single block.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_