Answer on Question #43690, Engineering, SolidWorks | CosmoWorks | Ansys

Task:

what is Elaborate on the filemode. Fileaccess ,and Fileshare enumeration detail explain.

Answer:

FileMode Enumeration: Specifies how the operating system should open a file.

	Member name	Description
	Append	Opens the file if it exists and seeks to the end of the file, or creates a new file. This requires FileIOPermissionAccess.Append permission.**FileMode.Append** can be used only in conjunction with **FileAccess.Write**. Trying to seek to a position before the end of the file throws anIOException exception, and any attempt to read fails and throws a NotSupportedException exception.
	Create	Specifies that the operating system should create a new file. If the file already exists, it will be overwritten. This requiresFileIOPermissionAccess.Write permission. **FileMode.Create** is equivalent to requesting that if the file does not exist, use CreateNew; otherwise, use Truncate. If the file already exists but is a hidden file, an UnauthorizedAccessException exception is thrown.
	CreateNew	Specifies that the operating system should create a new file. This requires FileIOPermissionAccess.Write permission. If the file already exists, an IOException exception is thrown.
	Open	Specifies that the operating system should open an existing file. The ability to open the file is dependent on the value specified by theFileAccess enumeration. A System.IO.FileNotFoundException exception is thrown if the file does not exist.
	OpenOrCreate	Specifies that the operating system should open a file if it exists; otherwise, a new file should be created. If the file is opened with**FileAccess.Read**, FileIOPermissionAccess.Read permission is required. If the file access is **FileAccess.Write**, FileIOPermissionAccess.Writepermission is required. If the file is opened with **FileAccess.ReadWrite**, both FileIOPermissionAccess.Read and FileIOPermissionAccess.Writepermissions are required.
	Truncate	Specifies that the operating system should open an existing file. When the file is opened, it should be truncated so that its size is zero bytes. This

		requires FileIOPermissionAccess.Write permission. Attempts to read from a file opened with **FileMode.Truncate** cause anArgumentException exception.

For an example of creating a file and writing text to a file, see How to: Write Text to a File. For an example of reading text from a file, see How to: Read Text from a File. For an example of reading from and writing to a binary file, see How to: Read and Write to a Newly Created Data File.
A **FileMode** parameter is specified in many of the constructors for FileStream, IsolatedStorageFileStream, and in the **Open** methods of File and FileInfo to control how a file is opened. **FileMode** parameters control whether a file is overwritten, created, opened, or some combination thereof. Use **Open** to open an existing file. To append to a file, use **Append**. To truncate a file or create a file if it doesn't exist, use **Create**.
The following **FileStream** constructor opens an existing file (**FileMode.Open**).
C# : FileStream s2 = new FileStream(name, FileMode.Open, FileAccess.Read, FileShare.Read);

FileAccess Enumeration : Defines constants for read, write, or read/write access to a file. This enumeration has a FlagsAttribute attribute that allows a bitwise combination of its member values.

	Member name	Description
	Read	Read access to the file. Data can be read from the file. Combine with **Write** for read/write access.
	ReadWrite	Read and write access to the file. Data can be written to and read from the file.
	Write	Write access to the file. Data can be written to the file. Combine with **Read** for read/write access.

For an example of creating a file and writing text to a file, see How to: Write Text to a File. For an example of reading text from a file, see How to: Read Text from a File. For an example of reading from and writing to a binary file, see How to: Read and Write to a Newly Created Data File.
A **FileAccess** parameter is specified in many of the constructors for File, FileInfo, FileStream, and other constructors where it is important to control the kind of access users have to a file.
The following **FileStream** constructor grants read-only access to an existing file (**FileAccess.Read**).
C# : FileStream s2 = new FileStream(name, FileMode.Open, FileAccess.Read, FileShare.Read);

FileShare Enumeration : Contains constants for controlling the kind of access other FileStream objects can have to the same file. This enumeration has a FlagsAttribute attribute that allows a bitwise combination of its member values.

	Member name	Description

	Delete	Allows subsequent deleting of a file.
	Inheritable	Makes the file handle inheritable by child processes. This is not directly supported by Win32.
	None	Declines sharing of the current file. Any request to open the file (by this process or another process) will fail until the file is closed.
	Read	Allows subsequent opening of the file for reading. If this flag is not specified, any request to open the file for reading (by this process or another process) will fail until the file is closed. However, even if this flag is specified, additional permissions might still be needed to access the file.
	ReadWrite	Allows subsequent opening of the file for reading or writing. If this flag is not specified, any request to open the file for reading or writing (by this process or another process) will fail until the file is closed. However, even if this flag is specified, additional permissions might still be needed to access the file.
	Write	Allows subsequent opening of the file for writing. If this flag is not specified, any request to open the file for writing (by this process or another process) will fail until the file is closed. However, even if this flag is specified, additional permissions might still be needed to access the file.

For an example of creating a file and writing text to a file, see How to: Write Text to a File. For an example of reading text from a file, see How to: Read Text from a File. For an example of reading from and writing to a binary file, see How to: Read and Write to a Newly Created Data File. A typical use of this enumeration is to define whether two processes can simultaneously read from the same file. For example, if a file is opened and Read is specified, other users can open the file for reading but not for writing. A FileShare parameter is specified in some of the constructors for FileStream, IsolatedStorageFileStream, and in some of the **Open** methods of File and FileInfo to control how a file is opened.

The following FileStream constructor opens an existing file and grants read-only access to other users (Read).

C# : FileStream s2 = new FileStream(name, FileMode.Open, FileAccess.Read, FileShare.Read);