# C# - FILE I/O



A **file** is a collection of data stored in a disk with a specific name and a directory path. When a file is opened for reading or writing, it becomes a **stream**.

The stream is basically the sequence of bytes passing through the communication path. There are two main streams: the **input stream** and the **output stream**. The **input stream** is used for reading data from file *readoperation* and the **output stream** is used for writing into the file *writeoperation*.

## C# I/O Classes

The System.IO namespace has various classes that are used for performing numerous operations with files, such as creating and deleting files, reading from or writing to a file, closing a file etc.

The following table shows some commonly used non-abstract classes in the System.IO namespace –

| Sr.No. | I/O Class & Description |
|---|---|
| 1 | **BinaryReader**<br><br>Reads primitive data from a binary stream. |
| 2 | **BinaryWriter**<br><br>Writes primitive data in binary format. |
| 3 | **BufferedStream**<br><br>A temporary storage for a stream of bytes. |
| 4 | **Directory**<br><br>Helps in manipulating a directory structure. |
| 5 | **DirectoryInfo**<br><br>Used for performing operations on directories. |

| | |
|---|---|
| 6 | **DriveInfo**<br>Provides information for the drives. |
| 7 | **File**<br>Helps in manipulating files. |
| 8 | **FileInfo**<br>Used for performing operations on files. |
| 9 | **FileStream**<br>Used to read from and write to any location in a file. |
| 10 | **MemoryStream**<br>Used for random access to streamed data stored in memory. |
| 11 | **Path**<br>Performs operations on path information. |
| 12 | **StreamReader**<br>Used for reading characters from a byte stream. |
| 13 | **StreamWriter**<br>Is used for writing characters to a stream. |
| 14 | **StringReader**<br>Is used for reading from a string buffer. |

| | |
|---|---|
| 15 | **StringWriter**<br><br>Is used for writing into a string buffer. |

## The FileStream Class

The **FileStream** class in the System.IO namespace helps in reading from, writing to and closing files. This class derives from the abstract class Stream.

You need to create a **FileStream** object to create a new file or open an existing file. The syntax for creating a **FileStream** object is as follows –

```
FileStream <object_name> = new FileStream( <file_name>, <FileMode Enumerator>,
   <FileAccess Enumerator>, <FileShare Enumerator>);
```

For example, we create a FileStream object **F** for reading a file named **sample.txt as shown** –

```
FileStream F = new FileStream("sample.txt", FileMode.Open, FileAccess.Read,
   FileShare.Read);
```

| Sr.No. | Parameter & Description |
|---|---|
| 1 | **FileMode**<br><br>The **FileMode** enumerator defines various methods for opening files. The members of the FileMode enumerator are –<br><br>• **Append** – It opens an existing file and puts cursor at the end of file, or creates the file, if the file does not exist.<br>• **Create** – It creates a new file.<br>• **CreateNew** – It specifies to the operating system, that it should create a new file.<br>• **Open** – It opens an existing file.<br>• **OpenOrCreate** – It specifies to the operating system that it should open a file if it exists, otherwise it should create a new file.<br>• **Truncate** – It opens an existing file and truncates its size to zero bytes. |
| 2 | **FileAccess** |

| | |
|---|---|
| | **FileAccess** enumerators have members: **Read**, **ReadWrite** and **Write**. |
| 3 | **FileShare**<br><br>**FileShare** enumerators have the following members –<br><br>• **Inheritable** – It allows a file handle to pass inheritance to the child processes<br>• **None** – It declines sharing of the current file<br>• **Read** – It allows opening the file for readin.<br>• **ReadWrite** – It allows opening the file for reading and writing<br>• **Write** – It allows opening the file for writing |

## Example

The following program demonstrates use of the **FileStream** class –

Live Demo

```
using System;
using System.IO;

namespace FileIOApplication {
   class Program {
      static void Main(string[] args) {
         FileStream F = new FileStream("test.dat", FileMode.OpenOrCreate,
            FileAccess.ReadWrite);

         for (int i = 1; i <= 20; i++) {
            F.WriteByte((byte)i);
         }
         F.Position = 0;
         for (int i = 0; i <= 20; i++) {
            Console.Write(F.ReadByte() + " ");
         }
         F.Close();
         Console.ReadKey();
      }
   }
}
```

When the above code is compiled and executed, it produces the following result –

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 -1
```

## Advanced File Operations in C#

The preceding example provides simple file operations in C#. However, to utilize the immense powers of C# System.IO classes, you need to know the commonly used properties and methods of these classes.

| Sr.No. | Topic & Description |
|---|---|
| 1 | Reading from and Writing into Text files<br><br>It involves reading from and writing into text files. The **StreamReader** and **StreamWriter** class helps to accomplish it. |
| 2 | Reading from and Writing into Binary files<br><br>It involves reading from and writing into binary files. The **BinaryReader** and **BinaryWriter** class helps to accomplish this. |
| 3 | Manipulating the Windows file system<br><br>It gives a C# programamer the ability to browse and locate Windows files and directories. |