# **📁File Handling**

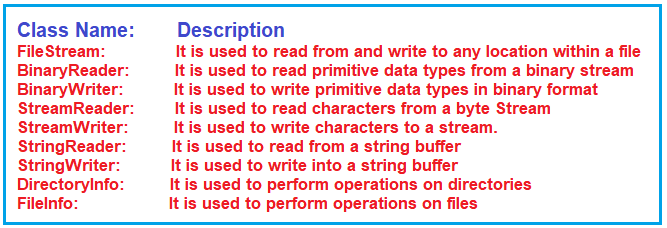
**File is collection of data stored on a disk, with specific name, extension, and directory path.**

Stream is when you open File using C# for reading and writing purposes.

Stream has two types :

* Input Stream : Used for read data from a file
* Output Stream : Used to write data into a file

## **🖱️**[**System.IO**](http://system.io/) **Namespace**



[System.IO](http://system.io/) Namespace contain required classes to handle I/O streams.

# **📂FileStream**

**FileStream class in C# provides a stream for file operations.**

It used to read and write to any location in a file.

## **❓How to use FileStream**

First of all, include the [System.IO](http://system.io/) namespace and create instance of FileStream object to create or open file.

Simplest way is :

| **public** **FileStream**(string path, FileMode mode); **public** **FileStream**(string path, FileMode mode, FileAccess access); **public** **FileStream**(string path, FileMode mode, FileAccess access, FileShare share); |
| --- |

path: absolute or relative path for the file

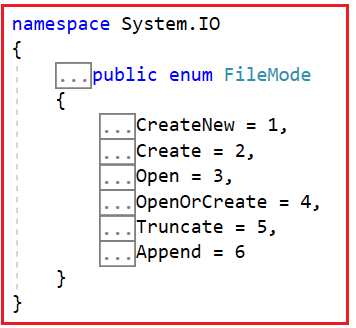
mode: determines to open or create the file

access: a constant that determines how the file can be accessed by the FileStream

share: determines how the file will be shared by processes

| **file**.Close(); //close fileStream **and** **release** any resources **file**.Flush(); //makesure **all** data **is** written **to** the **file**, even **buffer** **is** **not** full **file**.Dispose(); //**release** **all** resources **to** avoid mem leaks. |
| --- |

### **📳FileMode**



**CreateNew**: Create new file, if exist, [System.IO.IO](http://system.io.io/)Exception will be thrown

**Create**: Create new file, if exist, overwrite. If exist but hidden, System.UnauthorizedAccessException will be thrown

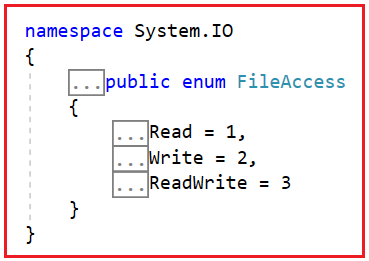
**Open**: Open existing file. If not found, thrown System.IO.FileNotFoundException

**OpenOrCreate** : Open existing, otherwise, create new file

**Truncate** : Open the file, make the size into zero bytes.

**Append** : Open the file if exist and seeks the end of file, if not exist, create new file.

### **♿FileAccess**

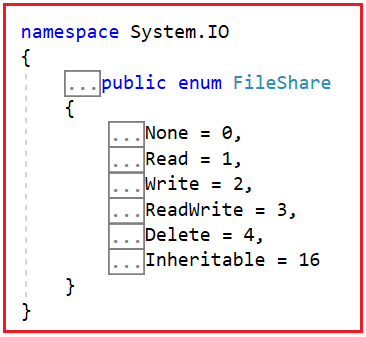


**Read** : Read access file

**Write** : Write access file

**ReadWrite** : Read and write access to the file

### **📤FileShare**



**None** : Decline sharing

**Read** : Allow subsequent opening of the file for reading

**Write** : Allow subsequent opening of the file writing

**ReadWrite** : Allow subsequent opening for writing and reading

**Delete** : Allow subsequent for delete the file

**Inheritable** : Makes the file handle inheritable by child process

## **🧪Example of using FileStream**

### **⚒️File Creation**

Example of code

| **using** System; **using** System.IO; **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  *//Set the File Path*  **string** FilePath = @"D:\MyFile.txt";  FileStream fileStream = **new** FileStream(FilePath, FileMode.Create);  fileStream.Close();  Console.Write("File has been created and the Path is D:\\MyFile.txt");  Console.ReadKey();  }  } } |
| --- |

### **📬File Open and File Write**

Example of code

| **using** System; **using** System.IO; **using** System.Text;  **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  *//Set the File Path*  **string** FilePath = @"D:\MyFile.txt";  FileStream fileStream = **new** FileStream(FilePath, FileMode.Append);  **byte**[] bdata = Encoding.Default.GetBytes("C# Is an Object Oriented Programming Language");  fileStream.Write(bdata, 0, bdata.Length);  fileStream.Close();  Console.WriteLine("Successfully saved file with data : C# Is an Object Oriented Programming Language");  Console.ReadKey();  }  } } |
| --- |

### **📖File Read**

Example of code

| **using** System; **using** System.IO; **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  **string** FilePath = @"D:\MyFile.txt";  **string** data;  FileStream fileStream = **new** FileStream(FilePath, FileMode.Open, FileAccess.Read);  **using** (StreamReader streamReader = **new** StreamReader(fileStream))  {  data = streamReader.ReadToEnd();  }  Console.WriteLine(data);  Console.ReadLine();  }  } } |
| --- |

# **🎏StreamWriter & StreamReader**

## **📝StreamWriter**

Its very helpful in writing text data in the file.

The StreamWriter writes the text data as individual characters, rather than must convert it first from data to binary.

**Constructor :**

StreamWriter() : used to initialize a new instance of the System.IO.StreamWriter

**Methods :**

Close() : Close the current StreamWriter

Flush() : Clears data from all buffers, write all left buffers to the file

Write() : Write data

WriteLine : Same like Write(), but add newline character at the end of the data

Dispose() : Release unmanaged resource

**Properties :**

AutoFlush : Get/set value

BaseStream : Gets underlying stream that interfaces with a backing store

Encoding : Gets System.Text.Encoding in which the output is written

Code to write user input into file

| **using** System; **using** System.IO; **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  *// This will create a file named MyFile.txt at the specified location i.e. in the D Drive*  *// Here we are using the StreamWriter constructor which takes the string path as an argument to create an instance of StreamWriter class*  StreamWriter sw = **new** StreamWriter("D://MyFile.txt");   *//Asking user to enter the text that we want to write into the MyFile.txt file*  Console.WriteLine("Enter the Text that you want to write on File");   *// To read the input from the user*  **string** str = Console.ReadLine();   *// To write the data into the stream*  sw.Write(str);   *// Clears all the buffers*  sw.Flush();   *// To close the stream*  sw.Close();  Console.ReadKey();  }  } } |
| --- |

Code for save variable data to file

| **using** System; **using** System.IO;  **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  **string** file = @"D:\MyTextFile1.txt";  **int** a, b, result;  a = 15;  b = 20;  result = a + b;   **using** (StreamWriter writer = **new** StreamWriter(file))  {  writer.Write("Sum of {0} + {1} = {2}", a, b, result);  }  Console.WriteLine("Saved");  Console.ReadKey();  }  } } |
| --- |

## **🕶️StreamReader**

Its very helpful to read text files easily.

**Constructor :**

StreamReader() : Initialize new instance

**Methods :**

Close() : Close StreamReader object, release any system resource

Peek() : Return next available character but does not consume it

Read() : Reads next character from input stream and advances the character position by one character

ReadLine() : Reads a line of character from current streams

Seek() : Read/write at specific location

**Properties :**

CurrentEncoding : Gets the current character encoding

EndOfStream : Indicate whether current stream position

BaseStream : It returns the underlying system

Code for read from a file

| **using** System; **using** System.IO; **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  *//Creating a new input stream i.e. MyFile.txt and opens it*  StreamReader sr = **new** StreamReader("D://MyFile.txt");   Console.WriteLine("Content of the File");   *// This is used to specify from where to start reading input stream*  sr.BaseStream.Seek(0, SeekOrigin.Begin);   *// To read line from input stream*  **string** str = sr.ReadLine();   *// To read the whole file line by line*  **while** (str != null)  {  Console.WriteLine(str);  str = sr.ReadLine();  }  Console.ReadLine();   *// to close the stream*  sr.Close();  Console.ReadKey();  }  } } |
| --- |

Code for read and write using StreamReader and StreamWriter

Code

| **using** System; **using** System.IO; **namespace** **FileHandlinDemo** {  **class** **Program**  {  **static** **void** **Main**(**string**[] args)  {  **string** FilePath = @"D:\MyFile.txt";  **string** data;  FileStream fileStream = **new** FileStream(FilePath, FileMode.Open, FileAccess.Read);  **using** (StreamReader streamReader = **new** StreamReader(fileStream))  {  data = streamReader.ReadToEnd();  }  Console.WriteLine(data);  Console.ReadLine();  }  } } |
| --- |

# **📁File Class**

1. **Copy**: This method is used to Copies an existing file to a new file. Overwriting a file of the same name is not allowed.
2. **Create**: This method is used to create or overwrite it in the specified path.
3. **Decrypt**: This method is used to Decrypt a file that was encrypted by the current account using the System.IO.File.Encrypt(System.String) method.
4. **Delete**: This method is used to delete the specified file.
5. **Encrypt**: This method is used to encrypt a file so that only the account used to encrypt the file can decrypt it.
6. **Open**: This method is used to Open a System.IO.FileStream on the specified path, having the specified mode with reading, write, or read/write access and the specified sharing option.
7. **Move**: This method is used to Move a specified file to a new location, providing the option to specify a new file name.
8. **Exists**: This method is used to determine whether the specified file exists.
9. **OpenRead**: This method is used to open an existing file for reading.
10. **OpenText**: This method is used to open an existing UTF-8 encoded text file for reading.
11. **OpenWrite**: This method is used to open an existing file or create a new file for writing.
12. **ReadAllBytes**: This method is used to open a binary file, read the contents of the file into a byte array, and then close the file.
13. **ReadAllLines**: This method is used to Open a file, reads all lines of the file with the specified encoding, and then close the file.
14. **ReadAllText**: This method is used to Open a text file, reads all the text in the file, and then close the file.
15. **ReadLines**: This method is used to read the lines of a file.
16. **Replace**: This method is used to replace the contents of a specified file with the contents of another file, delete the original file, and create a backup of the replaced file.
17. **WriteAllBytes**: This method is used to create a new file, write the specified byte array to the file, and then close the file. If the target file already exists, it is overwritten.
18. **WriteAllLines**: This method is used to create a new file, write the specified string array to the file, and then close the file.
19. **WriteAllText**: This method is used to create a new file, write the specified string to the file, and then close the file. If the target file already exists, it is overwritten.

Code Exist Method

| **using** System; **using** System.IO; **namespace** FileHandlinDemo {  class Program  {  **static** **void** Main(**string**[] args)  {  **string** path = @"D:\MyFile.txt";   if (File.Exists(path))  {  Console.WriteLine("MyFile.txt File Exists in D Directory");  }  else  {  Console.WriteLine("MyFile.txt File Does Not Exist in D Directory");  }  Console.ReadKey();  }  } } |
| --- |

Code ReadAllLines

| **using** System; **using** System.IO; **namespace** FileHandlinDemo {  class Program  {  **static** **void** Main(**string**[] args)  {  **string** path = @"D:\MyFile.txt";   if (File.Exists(path))  {  **string**[] lines= File.ReadAllLines(path);  foreach(var line in lines)  {  Console.WriteLine(line);  }  }  else  {  Console.WriteLine("MyFile.txt File Does Not Exists in D Directory");  }  Console.ReadKey();  }  } } |
| --- |

Code CopyMethod

| **using** System; **using** System.IO; **namespace** FileHandlinDemo {  class Program  {  **static** **void** Main(**string**[] args)  {  **string** SourceFilePath = @"D:\MyFile.txt";  **string** DestinationFilePath = @"D:\MyFile2.txt";   if (File.Exists(SourceFilePath))  {  File.Copy(SourceFilePath, DestinationFilePath);  **string** lines= File.ReadAllText(DestinationFilePath);  Console.WriteLine(lines);  }  else  {  Console.WriteLine("MyFile.txt File Does Not Exists in D Directory");  }  Console.ReadKey();  }  } } |
| --- |

Code Delete Method

| **using** System; **using** System.IO; **namespace** FileHandlinDemo {  class Program  {  **static** **void** Main(**string**[] args)  {  **string** FilePath = @"D:\MyFile2.txt";   if (File.Exists(FilePath))  {  File.Delete(FilePath);  Console.WriteLine("MyFile2.txt File Deleted");  }  else  {  Console.WriteLine("MyFile.txt File Does Not Exists in D Directory");  }  Console.ReadKey();  }  } } |
| --- |

Code CreateMethod

| **using** System; **using** System.IO; **namespace** FileHandlinDemo {  class Program  {  **static** **void** Main(**string**[] args)  {  *//Set the File Path*  **string** FilePath = @"D:\MyFile3.txt";   *//Create the File*  FileStream fs = File.Create(FilePath);  fs.Close();   if (File.Exists(FilePath))  {  *//Writing Content to File*  **string**[] content = { "Hello", "And", "Welcome" };  File.WriteAllLines(FilePath, content);  Console.WriteLine("MyFile3.txt File Created with the Following Data");   **string** fileContent = File.ReadAllText(FilePath);  Console.WriteLine(fileContent);  }  else  {  Console.WriteLine("MyFile.txt File Does Not Exists in D Directory");  }  Console.ReadKey();  }  } } |
| --- |

# **Compression Streams**

General purpose compression :

* DeflateStream : Same way like GZipStream, but better compression
* GZipStream : Compression using algorithm similar to ZIP format
* BrotliSteram : 10 times slower, but better compression