

Components of a file, different operation of the file, communication in files, creation of file streams, stream classes, header files, updating of file, opening and closing a file, file pointers and their manipulations, functions manipulation of file pointers, detecting end-of-file

---

## File Handling In C++

Files are used to store data in a storage device permanently. File handling provides a mechanism to store the output of a program in a file and to perform various operations on it.

A stream is an abstraction that represents a device on which operations of input and output are performed. A stream can be represented as a source or destination of characters of indefinite length depending on its usage.

In C++ we have a set of file handling methods. These include `ifstream`, `ofstream`, and `fstream`. These classes are derived from `fstreambase` and from the corresponding `iostream` class. These classes, designed to manage the disk files, are declared in `fstream` and therefore we must include `fstream` and therefore we must include this file in any program that uses files.

In C++, files are mainly dealt by using three classes **`fstream`**, **`ifstream`**, **`ofstream`**.

**`ofstream`**: This Stream class signifies the output file stream and is applied to create files for writing information to files

**`ifstream`**: This Stream class signifies the input file stream and is applied for reading information from files

**`fstream`**: This Stream class can be used for both read and write from/to files.

All the above three classes are derived from `fstreambase` and from the corresponding `iostream` class and they are designed specifically to manage disk files.

C++ provides us with the following operations in File Handling:

- Creating a file: `open()`
- Reading data: `read()`
- Writing new data: `write()`
- Closing a file: `close()`

## Opening a File

Generally, the first operation performed on an object of one of these classes is to associate it to a real file. This procedure is known to open a file.

We can open a file using any one of the following methods:

1. First is bypassing the file name in constructor at the time of object creation.

2. Second is using the open() function.

### To open a file use

open() function

Syntax

```
void open(const char* file_name,ios::openmode mode);
```

Here, the first argument of the open function defines the name and format of the file with the address of the file.

The second argument represents the mode in which the file has to be opened. The following modes are used

Modes	Description
In	Opens the file to read(default for ifstream)
Out	Opens the file to write(default for ofstream)
Binary	Opens the file in binary mode
App	Opens the file and appends all the outputs at the end
Ate	Opens the file and moves the control to the end of the file
Trunk	Removes the data in the existing file
nocreate	Opens the file only if it already exists
noreplace	Opens the file only if it does not already exist
In	Opens the file to read(default for ifstream)
Out	Opens the file to write(default for ofstream)
Binary	Opens the file in binary mode

### Example :

```
fstream new_file;
```

```
new_file.open("newfile.txt", ios::out);
```

In the above example, new\_file is an object of type fstream, as we know fstream is a class so we need to create an object of this class to use its member functions. So we create new\_file object and call open() function. Here we use out mode that allows us to open the file to write in it.

### Default Open Modes:

- fstream ios::in
- ofstream ios::out
- fstream ios::in | ios::out

We can combine the different modes using or symbol |

### Example :

```
ofstream new_file;
```

```
new_file.open("new_file.txt", ios::out | ios::app );
```

Here, input mode and append mode are combined which represents the file is opened for writing and appending the outputs at the end.

As soon as the program terminates, the memory is erased and frees up the memory allocated and closes the files which are opened.

But it is better to use the close() function to close the opened files after the use of the file.

Using a stream insertion operator << we can write information to a file and using stream extraction operator >> we can easily read information from a file.

### Example of opening/creating a file using the open() function

```
#include<iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
fstream new_file;
```

```
new_file.open("new_file",ios::out);
```

```
if(!new_file)
```

```
{
```

```
cout<<"File creation failed";
```

```
}
```

```
else
```

```
{
```

```
cout<<"New file created";
```

```
new_file.close(); // Step 4: Closing file
```

```
}
```

```
return 0;
```

```
}
```

### Output:

```
C:\Users\Lenovo\Desktop>g++ new_file.cpp -o new_file.exe
```

```
C:\Users\Lenovo\Desktop>new_file.exe
```

```
New file created
```

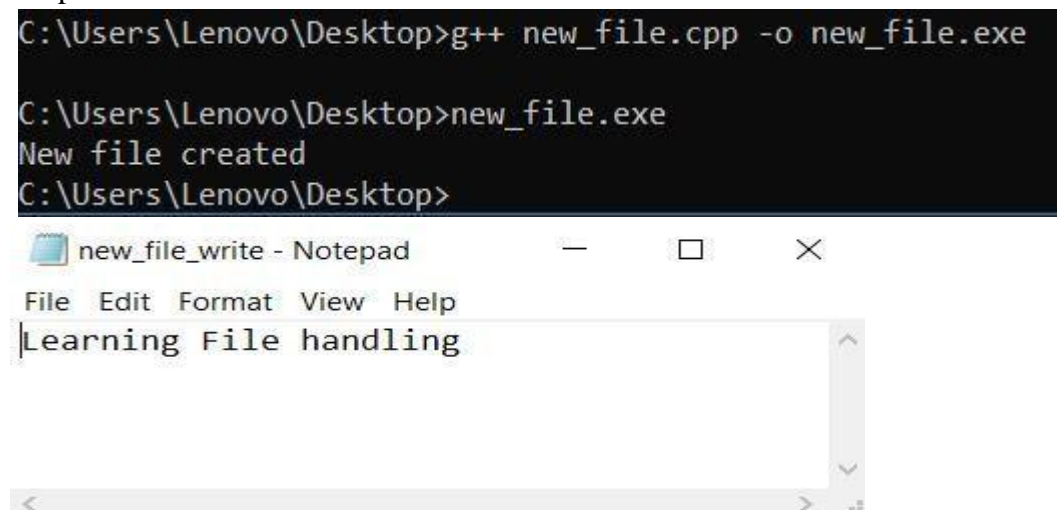
```
C:\Users\Lenovo\Desktop>
```

**Explanation :** In the above example we first create an object to class fstream and name it 'new\_file'. Then we apply the open() function on our 'new\_file' object. We give the name 'new\_file' to the new file we wish to create and we set the mode to 'out' which allows us to write in our file. We use a 'if' statement to find if the file already exists or not if it does exist then it will going to print "File creation failed" or it will gonna create a new file and print "New file created"

### Writing to a File

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    fstream new_file;
    new_file.open("new_file_write.txt",ios::out);
    if(!new_file)
    {
        cout<<"File creation failed";
    }
    else
    {
        cout<<"New file created";
        new_file<<"Learning File handling"; //Writing to file
        new_file.close();
    }
    return 0;
}
```

Output:



```
C:\Users\Lenovo\Desktop>g++ new_file.cpp -o new_file.exe

C:\Users\Lenovo\Desktop>new_file.exe
New file created
C:\Users\Lenovo\Desktop>
```

The screenshot shows a Windows command prompt window with the following text:

```
C:\Users\Lenovo\Desktop>g++ new_file.cpp -o new_file.exe

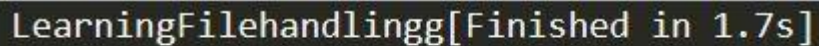
C:\Users\Lenovo\Desktop>new_file.exe
New file created
C:\Users\Lenovo\Desktop>
```

Below the command prompt is a Notepad window titled "new\_file\_write - Notepad". The menu bar includes "File", "Edit", "Format", "View", and "Help". The text area contains the string "Learning File handling".

### Reading from a File :

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    fstream new_file;
    new_file.open("new_file_write.txt",ios::in);
    if(!new_file)
        cout<<"No such file"; } else { char ch; while (!new_file.eof()) { new_file >>ch;
        cout << ch;
    }
    new_file.close();
    return 0;
}
```

Output:

A dark rectangular box with a light border containing the text "LearningFilehandling[Finished in 1.7s]" in a light-colored monospace font.

### Close a File :

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    fstream new_file;
    new_file.open("new_file.txt",ios::out);
    new_file.close();
    return 0;
}
```

**Output:** The file gets closed.