**Opening Files**

In MFC, the most common way to open a file is a two-stage process.

**To open a file**

1. Create the file object without specifying a path or permission flags.

You usually create a file object by declaring a [CFile](http://msdn.microsoft.com/en-us/library/60fh2b6f.aspx) variable on the stack frame.

1. Call the [Open](http://msdn.microsoft.com/en-us/library/hwbccf8z.aspx) member function for the file object, supplying a path and permission flags.

**The return value for Open will be nonzero if the file was opened successfully or 0 if the specified file could not be opened**. The **Open** member function is prototyped as follows:

virtual BOOL Open( LPCTSTR lpszFileName, UINT nOpenFlags, CFileException\* pError = NULL );

The open flags specify which permissions, such as read-only, you want for the file. The possible flag values are defined as enumerated constants within the **CFile** class, so they are qualified with "**CFile::**" as in **CFile::modeRead**. Use the **CFile::modeCreate** flag if you want to create the file.

The following example shows how to create a new file with read/write permission (replacing any previous file with the same path):

TCHAR\* pszFileName = \_T("c:\\test\\myfile.dat");

CFile myFile;

CFileException fileException;

if ( !myFile.Open( pszFileName, CFile::modeCreate |

CFile::modeReadWrite, &fileException ) )

{

TRACE( \_T("Can't open file %s, error = %u\n"),

pszFileName, fileException.m\_cause );

}

# Reading and Writing Files

If you've used the C run-time library file-handling functions, MFC reading and writing operations will appear familiar. This article describes reading directly from and writing directly to a **CFile** object. You can also do buffered file I/O with the [CArchive](http://msdn.microsoft.com/en-us/library/caz3zy5s.aspx) class.

### To read from and write to the file

1. Use the **Read** and **Write** member functions to read and write data in the file.
2. The **Seek** member function is also available for moving to a specific offset within the file.

**Read** takes a pointer to a buffer and the number of bytes to read and returns the actual number of bytes that were read. If the required number of bytes could not be read because end-of-file (EOF) is reached, the actual number of bytes read is returned. If any read error occurs, an exception is thrown. **Write**is similar to **Read**, but the number of bytes written is not returned. If a write error occurs, including not writing all the bytes specified, an exception is thrown. If you have a valid **CFile** object, you can read from it or write to it as shown in the following example:

TCHAR szBuffer[256];

UINT nActual = 0;

CFile myFile;

if ( myFile.Open( \_T("c:\\test\\myfile.dat"), CFile::modeCreate |

CFile::modeReadWrite ) )

{

myFile.Write( szBuffer, sizeof( szBuffer ) );

myFile.Flush();

myFile.Seek( 0, CFile::begin );

nActual = myFile.Read( szBuffer, sizeof( szBuffer ) );

}