DeleteFile function

Deletes an existing file.

To perform this operation as a transacted operation, use the **DeleteFileTransacted** function.

Syntax

C++

```
BOOL WINAPI DeleteFile(
  _In_ LPCTSTR lpFileName
);
```

Parameters

lpFileName [in]

The name of the file to be deleted.

In the ANSI version of this function, the name is limited to **MAX_PATH** characters. To extend this limit to 32,767 wide characters, call the Unicode version of the function and prepend "\\?\" to the path. For more information, see Naming a File.

Return value

If the function succeeds, the return value is nonzero.

If the function fails, the return value is zero (0). To get extended error information, call GetLastError.

Remarks

If an application attempts to delete a file that does not exist, the **DeleteFile** function fails with **ERROR_FILE_NOT_FOUND**. If the file is a read-only file, the function fails with **ERROR_ACCESS_DENIED**.

The following list identifies some tips for deleting, removing, or closing files:

- To delete a read-only file, first you must remove the read-only attribute.
- To delete or rename a file, you must have either delete permission on the file, or delete child permission in the parent directory.
- To recursively delete the files in a directory, use the SHFileOperation function.

- To remove an empty directory, use the RemoveDirectory function.
- To close an open file, use the CloseHandle function.

If you set up a directory with all access except delete and delete child, and the access control lists (ACL) of new files are inherited, then you can create a file without being able to delete it. However, then you can create a file, and then get all the access you request on the handle that is returned to you at the time you create the file.

If you request delete permission at the time you create a file, you can delete or rename the file with that handle, but not with any other handle. For more information, see File Security and Access Rights.

The **DeleteFile** function fails if an application attempts to delete a file that has other handles open for normal I/O or as a memory-mapped file (**FILE_SHARE_DELETE** must have been specified when other handles were opened).

The **DeleteFile** function marks a file for deletion on close. Therefore, the file deletion does not occur until the last handle to the file is closed. Subsequent calls to CreateFile to open the file fail with **ERROR_ACCESS_DENIED**.

Symbolic link behavior—

If the path points to a symbolic link, the symbolic link is deleted, not the target. To delete a target, you must call CreateFile and specify **FILE_FLAG_DELETE_ON_CLOSE**.

In Windows 8 and Windows Server 2012, this function is supported by the following technologies.

Technology	Supported
Server Message Block (SMB) 3.0 protocol	Yes
SMB 3.0 Transparent Failover (TFO)	Yes
SMB 3.0 with Scale-out File Shares (SO)	Yes
Cluster Shared Volume File System (CsvFS)	Yes
Resilient File System (ReFS)	Yes

Examples

For an example, see Locking and Unlocking Byte Ranges in Files.

Requirements

Minimum supported client	Windows XP [desktop apps | Windows Store apps]
Minimum supported server	Windows Server 2003 [desktop apps | Windows Store apps]
Minimum supported phone	Windows Phone 8
Header	FileAPI.h (include Windows.h); WinBase.h on Windows Server 2008 R2, Windows 7, Windows Server 2008, Windows Vista, Windows Server 2003, and Windows XP (include Windows.h)
Library	Kernel32.lib
DLL	Kernel32.dll
Unicode and ANSI names	**DeleteFileW** (Unicode) and **DeleteFileA** (ANSI)

See also

CloseHandle
CreateFile
DeleteFileTransacted
File Management Functions
Symbolic Links