FindFirstFile function

Searches a directory for a file or subdirectory with a name that matches a specific name (or partial name if wildcards are used).

To specify additional attributes to use in a search, use the **FindFirstFileEx** function.

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

Syntax

C++

```
HANDLE WINAPI FindFirstFile(
  _In_  LPCTSTR           lpFileName,
  _Out_ LPWIN32_FIND_DATA lpFindFileData
);
```

Parameters

lpFileName [in]

The directory or path, and the file name, which can include wildcard characters, for example, an asterisk (*) or a question mark (?).

This parameter should not be **NULL**, an invalid string (for example, an empty string or a string that is missing the terminating null character), or end in a trailing backslash (\).

If the string ends with a wildcard, period (.), or directory name, the user must have access permissions to the root and all subdirectories on the path.

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**.

lpFindFileData [out]

A pointer to the **WIN32_FIND_DATA** structure that receives information about a found file or directory.

Return value

If the function succeeds, the return value is a search handle used in a subsequent call to **FindNextFile** or **FindClose**, and the *lpFindFileData* parameter contains information about the first file or directory found.

If the function fails or fails to locate files from the search string in the *lpFileName* parameter, the return value is **INVALID_HANDLE_VALUE** and the contents of *lpFindFileData* are indeterminate. To get extended error information, call the GetLastError function.

If the function fails because no matching files can be found, the GetLastError function returns **ERROR_FILE_NOT_FOUND**.

Remarks

The **FindFirstFile** function opens a search handle and returns information about the first file that the file system finds with a name that matches the specified pattern. This may or may not be the first file or directory that appears in a directory-listing application (such as the dir command) when given the same file name string pattern. This is because **FindFirstFile** does no sorting of the search results. For additional information, see FindNextFile.

The following list identifies some other search characteristics:

- The search is performed strictly on the name of the file, not on any attributes such as a date or a file type (for other options, see FindFirstFileEx).
- The search includes the long and short file names.
- An attempt to open a search with a trailing backslash always fails.
- Passing an invalid string, **NULL**, or empty string for the *lpFileName* parameter is not a valid use of this function. Results in this case are undefined.

> **Note** In rare cases or on a heavily loaded system, file attribute information on NTFS file systems may not be current at the time this function is called. To be assured of getting the current NTFS file system file attributes, call the GetFileInformationByHandle function.

After the search handle is established, you can use it to search for other files that match the same pattern by using the FindNextFile function.

When the search handle is no longer needed, close it by using the FindClose function, not CloseHandle.

As stated previously, you cannot use a trailing backslash (\) in the *lpFileName* input string for **FindFirstFile**, therefore it may not be obvious how to search root directories. If you want to see files or get the attributes of a root directory, the following options would apply:

- To examine files in a root directory, you can use "C:*" and step through the directory by using FindNextFile.
- To get the attributes of a root directory, use the GetFileAttributes function.

> **Note** Prepending the string "\\?\" does not allow access to the root directory.

On network shares, you can use an *lpFileName* in the form of the following: "\\Server\Share*". However, you cannot use an *lpFileName* that points to the share itself; for example, "\\Server\Share" is not valid.

To examine a directory that is not a root directory, use the path to that directory, without a trailing backslash. For example, an argument of "C:\Windows" returns information about the directory "C:\Windows", not about a directory or file in "C:\Windows". To examine the files and directories in "C:\Windows", use an *lpFileName* of "C:\Windows*".

Be aware that some other thread or process could create or delete a file with this name between the time you query for the result and the time you act on the information. If this is a potential concern for your application, one possible solution is to use the CreateFile function with **CREATE_NEW** (which fails if the file exists) or **OPEN_EXISTING** (which fails if the file does not exist).

If you are writing a 32-bit application to list all the files in a directory and the application may be run on a 64-bit computer, you should call the Wow64DisableWow64FsRedirectionfunction before calling **FindFirstFile** and call Wow64RevertWow64FsRedirection after the last call to FindNextFile. For more information, see File System Redirector.

If the path points to a symbolic link, the WIN32_FIND_DATA buffer contains information about the symbolic link, not the target.

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

The following C++ example shows you a minimal use of **FindFirstFile**.

C++

```
#include <windows.h>
#include <tchar.h>
#include <stdio.h>

void _tmain(int argc, TCHAR *argv[])
{
   WIN32_FIND_DATA FindFileData;
   HANDLE hFind;

   if( argc != 2 )
   {
      _tprintf(TEXT("Usage: %s [target_file]\n"), argv[0]);
      return;
   }

   _tprintf (TEXT("Target file is %s\n"), argv[1]);
   hFind = FindFirstFile(argv[1], &FindFileData);
   if (hFind == INVALID_HANDLE_VALUE)
   {
      printf ("FindFirstFile failed (%d)\n", GetLastError());
      return;
   }
   else
   {
      _tprintf (TEXT("The first file found is %s\n"),
                FindFileData.cFileName);
      FindClose(hFind);
   }
}
```

For another example, see **Listing the Files in a Directory**.

Requirements

Minimum supported client	Windows XP [desktop apps only]
Minimum supported server	Windows Server 2003 [desktop apps only]

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	**FindFirstFileW** (Unicode) and **FindFirstFileA** (ANSI)

See also

File Management Functions
FindClose
FindFirstFileEx
FindFirstFileTransacted
FindNextFile
GetFileAttributes
SetFileAttributes
Symbolic Links
Using the Windows Headers
WIN32_FIND_DATA