# How do I create a shortcut whose target is specified by a relative path?

devblogs.microsoft.com/oldnewthing/20171019-00

October 19, 2017

Raymond Chen

Commenter Richard wonders if there's such a thing as a "relative" shortcut whose target is specified by a path relative to the shortcut itself.

Let's start with a program that creates a normal shortcut. This is a Little Program which does little to no error checking.

```
#define UNICODE
#define _UNICODE
#define STRICT
#include <windows.h>
#include <shlobj.h>
#include <atlbase.h>
#include <pathcch.h>

int __cdecl wmain(int, wchar_t **)
{
 CCoInitialize init;

 CComPtr<IShellLink> link;
 link.CoCreateInstance(CLSID_ShellLink);

 wchar_t path[MAX_PATH];
 GetModuleFileName(GetModuleHandle(nullptr), path, ARRAYSIZE(path));
 link->SetPath(path);

 PathCchRemoveFileSpec(path, ARRAYSIZE(path));
 PathCchAppend(path, ARRAYSIZE(path), L"Awesome.lnk");

 CComQIPtr<IPersistFile>(link)->Save(path, FALSE);

 return 0;
}
```

When run, this program creates a shortcut to itself in the same directory as the program.

Here are the changes necessary to make the shortcut remember that the target's location relative to the shortcut itself:

```
//
```

It's a trick. Normal shortcuts already remember the target's location relative to the shortcut itself.

However, the relative path is not used until other avenues have been exhausted. To give the relative path more prominence, let's disable the other avenues.

```
#define UNICODE
#define _UNICODE
#define STRICT
#include <windows.h>
#include <shlobj.h>
#include <atlbase.h>
#include <pathcch.h>

int __cdecl wmain(int, wchar_t **)
{
 CCoInitialize init;

 CComPtr<IShellLink> link;
 link.CoCreateInstance(CLSID_ShellLink);

 // Disable other ways of resolving the shortcut
 CComQIPtr<IShellLinkDataList> dataList(link);
 DWORD flags;
 dataList->GetFlags(&flags);
 flags |= SLDF_FORCE_NO_LINKINFO;
 flags |= SLDF_FORCE_NO_LINKTRACK;
 dataList->SetFlags(flags);

 wchar_t path[MAX_PATH];
 GetModuleFileName(GetModuleHandle(nullptr), path, ARRAYSIZE(path));
 link->SetPath(path);

 PathCchRemoveFileSpec(path, ARRAYSIZE(path));
 PathCchAppend(path, ARRAYSIZE(path), L"Awesome.lnk");

 CComQIPtr<IPersistFile>(link)->Save(path, FALSE);

 return 0;
}
```

The `SLDF_FORCE_NO_LINKINFO` flag disables the information the shell normally creates to identify the volume that contains the shortcut target. This is used, for example, if the target was on a network volume, so the shell can reconnect to that volume in order to find the target.

The `SLDF_FORCE_NO_LINKTRACK` flag disables the information the shell normally creates to identify the object with the assistance of the distributed link tracking service.

Deleting those two pieces of information means that the shell cannot use them to help find the shortcut. If the file doesn't exist at the specified absolute path, then the relative path will be applied to the location of the shortcut itself, and the shell will look for the file at the resulting location.

**Bonus chatter**: The `IShellLink::SetRelativePath` method is for the benefit of code which stores shortcuts in places other than files. You call `IShellLink::SetRelativePath` before saving the shortcut to memory, passing a path to a location you want to pretend the shortcut file is saved. The shell will remember the path to the shortcut target relative to the path you pass in. Conversely, after you load the shortcut from memory, you call `IShellLink::SetRelativePath` to specify the path to pretend the shortcut was loaded from. The relative path remembered when the shortcut was saved will be applied to this path to help find the target.

**Double bonus chatter**: You can use a simple pidl to make the shortcut target reside at any path you like.

Raymond Chen

**Follow**