Shortcuts are serializable objects, which means that they can be stored in places other than just a file

devblogs.microsoft.com/oldnewthing/20110224-00

February 24, 2011

Raymond Chen

It's true that the vast majority of the time, people consider the shell shortcut object as synonymous with the `.lnk` file it is normally saved into, shortcuts need not spend their time in a file. You can put a shortcut anywhere you can save a hunk of bytes. Here's a program that creates a shortcut to the file name passed on the command line (make sure it's a full path), and then serializes the shortcut to a blob of bytes (in the form of a `HGLOBAL`). Once that's done, it reconstitutes the bytes back into a shortcut object and sucks information out of it.

```
#define UNICODE
#define _UNICODE
#include <windows.h>
#include <shlobj.h>
#include <ole2.h>
#include <stdio.h>
#include <tchar.h>
#include <atlbase.h>
HGLOBAL CreateShellLinkInMemory(PCWSTR pszFile)
{
 BOOL fSuccess = FALSE;
 HGLOBAL hglob = GlobalAlloc(GMEM_MOVEABLE, 0);
 if (hglob) {
  CComPtr<IStream> spstm;
  if (SUCCEEDED(CreateStreamOnHGlobal(hglob, FALSE, &spstm))) {
   CComPtr<IShellLink> spsl;
   if (SUCCEEDED(spsl.CoCreateInstance(CLSID_ShellLink))) {
    if (SUCCEEDED(spsl->SetPath(pszFile))) {
     CComQIPtr<IPersistStream> spps(spsl);
     fSuccess = spps && SUCCEEDED(spps->Save(spstm, TRUE));
    }
   }
  }
 }
 if (fSuccess) return hglob;
 if (hglob) GlobalFree(hglob);
 return NULL;
}
```

After creating the shortcut object, we serialize it into a stream backed by a chunk of memory we record in a `HGLOBAL`. The shortcut object itself is no longer anywhere to be seen. It's been dehydrated into a pile of dust like in that old *Star Trek* episode.

But this time, we know how to bring it back.

```
IShellLink *CreateShellLinkFromMemory(HGLOBAL hglob)
{
 IShellLink *pslReturn = NULL;
 CComPtr<IStream> spstm;
 if (SUCCEEDED(CreateStreamOnHGlobal(hglob, FALSE, &spstm))) {
  CComPtr<IShellLink> spsl;
  if (SUCCEEDED(spsl.CoCreateInstance(CLSID_ShellLink))) {
   CComQIPtr<IPersistStream> spps(spsl);
   if (spps && SUCCEEDED(spps->Load(spstm))) {
    pslReturn = spsl.Detach();
   }
  }
 }
 return pslReturn;
}
```

We create a new shortcut object and tell it to restore itself from the chunk of memory we squirreled away. Bingo, the shortcut is back, ready for action.

```
int __cdecl wmain(int argc, WCHAR **argv)
{
 if (SUCCEEDED(CoInitialize(NULL))) {
  HGLOBAL hglob = CreateShellLinkInMemory(argv[1]);
  if (hglob) {
   CComPtr<IShellLink> spsl;
   spsl.Attach(CreateShellLinkFromMemory(hglob));
   if (spsl) {
    WCHAR szTarget[MAX_PATH];
    if (spsl->GetPath(szTarget, MAX_PATH, NULL, 0) == S_OK) {
     wprintf(L"Welcome back, shortcut to %s\n", szTarget);
    }
   }
   GlobalFree(hglob);
  }
  CoUninitialize();
 }
 return 0;
}
```

Since shortcuts can be stored anywhere, you can't rely on the file name to distinguish between shortcuts to files and shortcuts to folders because there may not be a file name at all! (What's the file name for our `HGLOBAL`?) Even if you decide that the convention applies only to shortcuts saved in a file, you've created an additional burden on people who manipulate shortcut files: They have to check whether the target is a file or folder before choosing the file

name, and if the target of the shortcut changes, they may have to rename the file as well. This is a real problem for the standard file property sheet: If you change the shortcut target from the *Shortcut* page, this might change the underlying file name. If you had also made changes to the *Security* page, it will try to update the security attributes on the old file name, even though the *Shortcut* page had renamed it. Oops, none of the other property sheet pages work, because they are now operating on a file that no longer exists!

Exercise: Under what conditions would it be useful to store a shortcut in memory rather than in a file? (Answer.)

Raymond Chen

Follow