Keep your eye on the code page: C# edition (warning about DllImport)

devblogs.microsoft.com/oldnewthing/20140812-00

August 12, 2014

Raymond Chen

Often, we receive problem reports from customers who failed to keep their eye on the code page.

> Does the `SHGetFileInfo` function support files with non-ASCII characters in their names? We find that the function either fails outright or returns question marks when asked to provide information for files with non-ASCII characters in their name.

```
using System;
using System.Runtime.InteropServices;
class Program
{
 static void Main(string[] args)
 {
  string fileName = "BgǒRồ.txt";
  Console.WriteLine("File exists? {0}", System.IO.File.Exists(fileName));
  // assumes extensions are hidden
  string expected = "BgǒRồ";
  Test(fileName, SHGFI_DISPLAYNAME, expected);
  Test(fileName, SHGFI_DISPLAYNAME | SHGFI_USEFILEATTRIBUTES, expected);
 }
 static void Test(string fileName, uint flags, string expected)
 {
  var actual = GetNameViaSHGFI(fileName, flags);
  Console.WriteLine("{0} == {1} ? {2}", actual, expected, actual == expected);
 }
 static string GetNameViaSHGFI(string fileName, uint flags)
 {
  SHFILEINFO sfi = new SHFILEINFO();
  if (SHGetFileInfo(fileName, 0, ref sfi, Marshal.SizeOf(sfi),
                    flags) != IntPtr.Zero) {
   return sfi.szDisplayName;
  } else {
   return null;
  }
 }
 [StructLayout(LayoutKind.Sequential)]
 struct SHFILEINFO {
  public IntPtr hIcon;
  public int iIcon;
  public uint dwAttributes;
  [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 260)]
  public string szDisplayName;
  [MarshalAs(UnmanagedType.ByValTStr, SizeConst = 80)]
  public string szTypeName;
 }
 const uint SHGFI_USEFILEATTRIBUTES = 0x10;
 const uint SHGFI_DISPLAYNAME = 0x0200;
 [DllImport("shell32.dll")]
 static extern IntPtr SHGetFileInfo(
    string path, uint fileAttributes, ref SHFILEINFO info, int cbSize,
    uint flags);
}
// Output:
// File exists? True
//  == Bg?R? ? False
// Bg?R? == Bg?R? ? False
```

> If we ask for the display name, the function fails even though the file does exist. If we also pass the `SHGFI_USEFILEATTRIBUTES` flag to force the system to act as if the file existed, then it returns the file name but with question marks where the non-ASCII characters should be.

The `SHGetFileInfo` function supports non-ASCII characters just fine, provided you call the version that supports non-ASCII characters!

The customer here fell into the trap of not keeping their eye on the code page. It goes back to an unfortunate choice of defaults in the System.Runtime.InteropServices namespace: At the time the CLR was originally being developed, Windows operating systems derived from Windows 95 were still in common use, so the CLR folks decided to default to *CharSet.Ansi*. This made sense back in the day, since it meant that your program ran the same on Windows 98 as it did in Windows NT. In the passage of time, the Windows 95 series of operating systems became obsolete, so the need to be compatible with it gradually disappeared. But too late. The rules were already set, and the default of *CharSet.Ansi* could not be changed.

The solution is to specify *CharSet.Unicode* explicitly in the *StructLayout* and *DllImport* attributes.

FxCop catches this error, flagging it as *SpecifyMarshalingForPInvokeStringArguments*. The error explanation talks about the security risks of unmapped characters, which is all well and good, but it is looking too much at the specific issue and not so much at the big picture. As a result, people may ignore the issue because it is flagged as a complicated security issue, and they will think, "Eh, this is just my unit test, I'm not concerned about security here." However, the big picture is

> This is almost certainly an oversight on your part. You didn't really mean to disable Unicode support here.

Change the lines

```
[StructLayout(LayoutKind.Sequential)]
[DllImport("shell32.dll")]
```

to

```
[StructLayout(LayoutKind.Sequential, CharSet=CharSet.Unicode)]
[DllImport("shell32.dll", CharSet=CharSet.Unicode)]
```

and re-run the program. This time, it prints

```
File exists? True
Bg?R? == Bg?R? ? True
Bg?R? == Bg?R? ? True
```

Note that you have to do the string comparison in the program because the console itself has a troubled history with Unicode. At this point, I will simply cue a Michael Kaplan rant and link to an article explaining how to ask nicely.

Raymond Chen

Follow