Keep your eye on the code page: C# edition (the mysterious third code page)

devblogs.microsoft.com/oldnewthing/20140813-00

August 13, 2014

Raymond Chen

A customer was having trouble manipulating the console from a C# program:

> We found that C# can read only ASCII data from the console. If we try to read non-ASCII data, we get garbage.
>
> ```
> using System;
> using System.Text;
> using System.Runtime.InteropServices;
> class Program
> {
>   [StructLayout(LayoutKind.Sequential)]
>   struct COORD
>   {
>     public short X;
>     public short Y;
>   }
>   [DllImport("kernel32.dll", SetLastError=true)]
>   static extern IntPtr GetStdHandle(int nStdHandle);
>   const int STD_OUTPUT_HANDLE = -11;
>   [DllImport("kernel32.dll", SetLastError=true)]
>   static extern bool ReadConsoleOutputCharacter(
>     IntPtr hConsoleOutput,
>     [Out] StringBuilder lpCharacter,
>     uint nLength,
>     COORD dwReadCoord,
>     out uint lpNumberOfCharsRead);
>   public static void Main()
>   {
>     // Write a string to a fixed position
>     System.Console.Clear();
>     System.Console.WriteLine("\u00C5ngstr\u00f6m");
>     // Read it back
>     COORD coord  = new COORD { X = 0, Y = 0 };
>     StringBuilder sb = new StringBuilder(8);
>     uint nRead = 0;
>     ReadConsoleOutputCharacter(GetStdHandle(STD_OUTPUT_HANDLE),
>                                sb, (uint)sb.Capacity, coord, out nRead);
>     // Trim off any unused excess.
>     sb.Remove((int)nRead, sb.Length - (int)nRead);
>     // Show what we read
>     System.Console.WriteLine(sb);
>   }
> }
> ```
>
> Observe that this program is unable to read the Å and ö characters. They come back as garbage.

Although there are three code pages that have special treatment in Windows, the CLR gives access to only two of them via `DllImport`.

- CharSet.Ansi = CP_ACP
- CharSet.Unicode = Unicode (which in Windows means UTF16-LE unless otherwise indicated).

Unfortunately, the console traditionally uses the OEM code page.

Since the `DllImport` did not specify a character set, the CLR defaults (unfortunately) to `CharSet.Ansi`. Result: The `ReadConsoleOutputCharacter` function stores its results in `CP_OEM`, the CLR treats the buffer as if it were `CP_ACP`, and the result is confusion.

The narrow-minded fix is to try to fix the mojibake by taking the misconverted Unicode string, converting it to bytes via the ANSI code page, then converting the bytes to Unicode via the OEM code page.

The better fix is simply to avoid the 8-bit code page issues entirely and say you want to use Unicode.

```
[DllImport("kernel32.dll", SetLastError=true, CharSet=CharSet.Unicode)]
static extern bool ReadConsoleOutputCharacter(...);
```

Raymond Chen

Follow