Is there a code page that matches ASCII and can round trip arbitrary bytes through Unicode?

devblogs.microsoft.com/oldnewthing/20200831-00

August 31, 2020

Raymond Chen

Is there a code page that matches ASCII for the first 128 values and can round trip arbitrary bytes through Unicode?

You may find yourself looking for such a code page when you have a chunk of binary data with embedded ASCII text. You want to be able to dig out and even manipulate the ASCII text, and treat the non-ASCII parts as mysterious characters that have no meaning, but you need to be able to convert them back into the original bytes.

For example, the format of the binary data might be an ASCII string followed by a 32-bit integer in big-endian format. You want to parse out the ASCII string, then take the next four characters, reverse them, and then convert them back to bytes so you can extract the integer.

The C# language has a lot of handy facilities for manipulating text, but they require C# `String` objects, which are expressed as a sequence of UTF16-LE code units. So what you want is a way to convert bytes into UTF16-LE code units, with the property that bytes less than 128 map to corresponding ASCII characters, and bytes 128 and above map to *something* in a reversible way.

The UTF-8 code page won't work because there are invalid byte sequences in UTF-8 which will not convert to text and back. Another choice you might go for is code page 1252, but it fails because there are many undefined code units, which means that a byte with one of those values may get converted to `U+FFFD` REPLACEMENT CHARACTER, which is a special Unicode character that means "There was a character here, but I can't express it in Unicode." It is commonly used to represent encoding errors.

Even though you are unlikely to consider it, I would just mention that double-byte code pages are also not going to work because they take pairs of bytes and convert them to Unicode. This means that reversing the Unicode code units and then converting back to binary is not going to work.

Okay, I'll cut to the chase. The code page I use for this sort of thing is code page 437. Every bytes is defined and maps to a unique Unicode code point, and it agrees with ASCII for the first 128 values.

```
using System;

class Program
{
    static public void Main()
    {
        var bytes = new byte[256];
        int i;
        for (i = 0; i < 256; i++)
        {
            bytes[i] = (byte)i;
        }
        var oem = System.Text.Encoding.GetEncoding(437);
        var text = oem.GetChars(bytes);
        Array.Reverse(text);
        bytes = oem.GetBytes(text);
        for (i = 0; i < 256; i++)
        {
            if (bytes[i] != (byte)(255 - i)) break;
        }
        System.Console.WriteLine(i); // should print 256
    }
}
```

I take the bytes 0 through 255 and convert them to a string via code page 437. I reverse the string, then convert back to bytes and verify that the resulting bytes are also reversed.

Raymond Chen

Follow