Keep your eye on the code page

devblogs.microsoft.com/oldnewthing/20050308-00

March 8, 2005

Raymond Chen

Remember that there are typically two 8-bit code pages active, the so-called “ANSI” code page and the so-called “OEM” code page. GUI programs usually use the ANSI code page for 8-bit files (though utf-8 is becoming more popular lately), whereas console programs usually use the OEM code page.

This means, for example, when you open an 8-bit text file in Notepad, it assumes the ANSI code page. But if you use the TYPE command from the command prompt, it will use the OEM code page.

This has interesting consequences if you switch between the GUI and the command line frequently.

The two code pages typically agree on the first 128 characters, but they nearly always disagree on the characters from 128 to 255 (so-called “extended characters”). For example, on a US-English machine, character 0x80 in the OEM code page is Ç, whereas in the ANSI code page it is €.

Consider a directory which contains a file named ç. If you type “dir” at a command prompt, you see a happy ç on the screen. On the other hand, if you do “dir >files.txt” and open files.txt in a GUI editor like Notepad, you will find that the Ç has changed to a €, because the 0x80 in the file is being interpreted in the ANSI character set instead of the OEM character set.

Stranger yet, if you mark/select the file name from the console window and paste it into Notepad, you get a Ç. That’s because the console window’s mark/select code saves text on the clipboard as Unicode; the character saved into the clipboard is not 0x80 but rather U+00C7, the Unicode code point for “Latin Capital Letter C With Cedilla”. When this is pasted into Notepad, it gets converted from Unicode to the ANSI code page, which on a US-English system encodes the Ç character as 0xC7.

But wait, there’s more. The command processor has an option (/U) to generate all piped and redirected output in Unicode rather than the OEM code page.

(Note that the built-in documentation for the command processor says that the /A switch produces ANSI output; this is incorrect. /A produces OEM output. This is one of those bugs that you recognize instantly if you are familiar with what is going on. It's so obviously OEM that when I see the documentation say "ANSI", my mind just reads it as "OEM". In the same way native English speakers often fail to notice misspellings or doubled words.)

If you run the command

```
cmd /U /C dir ^>files.txt
```

then the output will be in Unicode and therefore will record the Ç character as U+00C7, which Notepad will then be able to read back.

This has serious consequences for batch files.

Batch files are 8-bit files and are interpreted according to the OEM character set. This means that if you write a batch file with Notepad or some other program that uses the ANSI character set for 8-bit files, and your batch file contains extended characters, the results you get will not match the what you see in your editor.

Why the discrepancy between GUI programs and console programs over how 8-bit characters should be interpreted?

The reason is, of course, historical.

Back in the days of MS-DOS, the code page was what today is called the OEM code page. For US-English systems, this is the code page with the box-drawing characters and the fragments of the integral signs. It contained accented letters, but not a very big set of them, just enough to cover the German, French, Spanish, and Italian languages. And Swedish. (Why Swedish yet not Danish and Norwegian I don't know.)

When Windows came along, it decided that those box-drawing characters were wasting valuable space that could be used for adding still more accented characters, so out went the box-drawing characters and in went characters for Danish, Norwegian, Icelandic, and Canadian French. (Yes, Canadian French uses characters that European French does not.)

Thus began the schism between console programs (MS-DOS) and GUI programs (Windows) over how 8-bit character data should be interpreted.

Raymond Chen

Follow