How can I detect the language a run of text is written in?

devblogs.microsoft.com/oldnewthing/20120111-00

January 11, 2012

Raymond Chen

A customer asked, “I have a Unicode string. I want to know what language that string is in. Is there a function that can give me this information? I am most interested in knowing whether it is written in an East Asian language.” The problem of determining the language in which a run of text is written is rather difficult. Many languages share the same script, or at least very similar scripts, so you can’t just go based on which Unicode code point ranges appear in the string of text. (And what if the text contains words from multiple languages?) With heuristics and statistical analysis and a large enough sample, the confidence level increases, but reaching 100% confidence is difficult. I vaguely recall that there is a string of text which is a perfectly valid sentence in both Spanish and Portuguese, but with radically different meanings in the two languages! The customer was unconvinced of the difficulty of this problem. “Language detection of a single Unicode character should work with 100% accuracy. After all, the operating system already has a function to do this. When I pass the run of text to GDI, it knows to use a Chinese font to render the Chinese characters and a Korean font to render the Korean characters.” The customer has fallen into the trap of confusing scripts with languages. The customer in this case is an East Asian company, so they have entered the linguistic world with a mindset that each language has its own unique script, since that is true for the languages in their part of the world. It’s actually kind of interesting seeing a different set of linguistic assumptions. Whereas companies in the United States assume that every language is like English, it appears that companies in East Asia assume that every language is like English, Japanese, Chinese, Korean, or Thai. In this company’s world, the letter “A” is clearly English, since it never occurred to them that it might be German, Swedish, or French. When GDI is asked to render a run of text, it looks for a font that can render each specific character, and once it finds such a font, it tries to keep using that font until it runs into a character which that font doesn’t support, and then it begins a new search. You can see this effect when a non-Western character is inserted into a string when rendered on a system whose default code page is Western. GDI will switch to a font that supports the non-Western character, and it will keep using that font for the remainder of the string, even though the rest of the string uses just the letters A through Z. For example, the string might render like this: Dvořak. GDI switched to a different font to

render the “ř” and remained in that font instead of returning to the original font for the “ak”. Anyway, the answer to the customer’s question of language detection is to use the language detection capability of the Extended Linguistic Services.

If you are operating in the more constrained world of “I just want to know if it’s Chinese/Japanese/Korean/Thai or isn’t,” then you could fall back to checking Unicode character ranges. If you see characters in the ranges dedicated to characters from those East Asian scripts, then you found text which is (at least partially) in one of those languages. Note, however, that this algorithm requires continual tweaking because the Unicode standard is a moving target. For example, the range of characters which can be used by East Asian languages expanded with the introduction of the Supplemental Ideographic Plane. You’re probably best just letting somebody else worry about this, say, by asking `GetStringTypeEx` for `CT_CTYPE3` information, or using `GetStringScripts` (or its redistributable doppelgänger `DownlevelGetStringScripts`) or simply by asking ELS to do everything.

Raymond Chen

Follow