Using alternate locales to get more interesting case mapping than the C

devblogs.microsoft.com/oldnewthing/20250207-00

February 7, 2025

Last time, we saw that the default C locale is not a very interesting one. So how do you get a locale that does something better?

One way to get functions like `_strlwr` and `_wcslwr` to follow a specific locale is to set that other locale as the current C runtime locale.

```
// Set the C runtime locale for character
// classification (which includes case mapping)
// to the user's default locale
_wsetlocale(LC_CTYPE, L"");

// Now you can convert to lowercase in a locale-aware manner
wchar_t example[] = L"\x00C0" L"BC"; // ÀBC
_wcslwr_s(example); // Result: probably àbc
```

It is convenient that an empty string is interpreted by `_wsetlocale()` to mean "the user's default locale", as determined by `GetUserDefaultLocaleName`.[1]

A major problem with this approach is that it is using global state to solve a local problem. The C runtime locale is a process-wide setting, so you changed the locale not just for your call to `_wcslwr_s`, but for everybody else's call to `_wcslwr_s` as well.

Better would be to leave the global locale alone and just say "For this call to `_wcslwr`, use the user's default locale."

```
// Create a locale that represents the user's default locale
auto l = _wcreate_locale(LC_CTYPE, L"");

// Convert to lowercase according to that locale
wchar_t example[] = L"\x00C0" L"BC"; // ÀBC
_wcslwr_s_l(example, l); // Result: probably àbc
```

Even if you go all this trouble, you are still failing to handle the case where changing the case of a string changes its length. For that, you have to go to `LCMapStringEx` or the corresponding ICU function `u_strToLower` or `u_strToUpper`.

```
wchar_t example[] = L"\x00C0" L"BC"; // ÀBC

// Error checking elided for expository purposes
wchar_t lowercase[256];
LCMapStringEx(LOCALE_NAME_USER_DEFAULT,
    LCMAP_LOWERCASE, example, ARRAYSIZE(example),
    lowercase, ARRAYSIZE(lowercase),
    nullptr, 0);
// Result: probably àbc
```

Here's a dirty little secret: When you call `_wcslwr` and the locale is not the C locale, then the Visual C++ runtime just calls `LCMapStringEx`. So you're doing the same thing at the end of the day, just with the ability to accommodate strings that change length during a change of case.

Bonus chatter: Not all implementations of `wcslwr` or `towlower` are high quality.

[1] The user default locale may not be the best locale for your thread because the caller may have called a function like `SetThreadLocale` or `SetThreadPreferredUILanguages` to change the thread's preferred locale to something other than the user's default. You need to call a function like `GetThreadPreferredUILanguages` to see those thread custom locales and pick the one (probably the first one) to use for case mapping.