Subject: WinAPI UNICODE question
Posted by cbpporter on Sat, 15 Nov 2008 10:34:55 GMT
View Forum Message <> Reply to Message

I'm trying to obtain a version of Core that does not expose one single bit of platform specific code at an interface level (but in implementation can use as much as needed), including WinAPI. This is not that easy. Simply removing include <windows.h> would lead to days of tracking down missing definitions.

So I started filtering windows.h, to make the task a lot easier. I also noticed some unneeded headers this way and a couple of cases where I think there ares mall errors related to "A" and "W" suffixes. Let me say that it is impressive how few WinAPI function U++ does actually use .

And this is where is stumbled across my question. For a given function f, under WinAPI there is a fA version, which works on ANSI strings, and a fW function which works on Unicode strings. If you want you application to be ansi, you leave the macro UNIDODE undefined, and f will automatically be translated to fA through macro substitution. By declaring UNICODE, it will get translated to fW. Unicode versions work only starting with windows 2000, but you can install some .dlls under Win95/98 to handle a subset of Unicode. Did I understand this correctlly?

So there are tree options:
1. Leave UNICODE undefined and get apps that run on all systems.
2. Define UNICODE and only run on Win2000 and 95/98 if dll is available.
3. Detect if Unicode support is available, and then call A or W version of functions manually depending on case.

I really can't figure out which one U++ uses. If I remove the #define SendMessage SendMessageA (for example), I get compilation error, which hints that UNICODE is undefined and we are using case number one. Also, there are some explicit cases where a W version is called, which makes me think that we are still using case 1 as a compilation environment, but we are trying to achieve functionality of 2/3 with the explicit calls of W variants in some cases.