Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 17 Oct 2007 23:34:21 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 17 October 2007 22:54sergei wrote on Wed, 17 October 2007 15:58
Any chance you fix that (rename the method)?

No.

IMO, you require me to fix the code to compile using unstandard broken method. I do not see that as bright idea. You can break the compilation in many ways. Should I fix the code to support them all?

Maybe you should rather think about fixing the build system...

Mirek

So far, not many things got wrong... AFAIK this is the last one, excluding C (non-C++ plugins). And they all were at least slightly unusual - using macro as function name, using an API function as a member function, having 2 equal enum members in different enums. You have used code to fix MSVC6 bugs - to support non-standard behavior. But I'm not MS

Your BLITZ approach isn't standard either. It's used, because it works, and because it has its benefits - speed. I wouldn't use SCU, if I didn't see how great BLITZ's benefits are - it's like edit-and-continue for library modifications. My SCU approach isn't much different. It's an attempt to mimic BLITZ behavior. Not a perfect attempt, I agree. I used a different include order (first all headers, then all CPPs) than BLITZ (for each package, first header, then CPPs), and that was the reason for ReplaceText problem.

But it looks like you treat SCU as non-standard and BLITZ as "just fine" - it isn't so. They are slightly different speed-up solutions. BLITZ is smarter regarding macros, but it's still SCU. In a.cpp write int MyVar, in b.cpp write float MyVar. In standard C++ this would compile (variables are local), but both in BLITZ and SCU this would fail. I might be wrong, but I don't think you intended ReplaceText member function to be substituted with ReplaceTextA in all places - this sound like something that can potentially break compilation (like if you define another ReplaceTextA function thinking it's a different one). Something else might work with my SCU but not BLITZ (like using #defines from a header in CPP from another package) - neither good nor intended.

I can change include order in my SCU to make it resemble BLITZ even more, yet this won't make neither SCU nor BLITZ standard.

Fixing the build system does sound more reasonable regarding C plugins. These indeed could be a time-bomb in SCU, especially the updating ones. Except for zlib (which works and is a vital part of Core), others could be packaged as libs (3rd party plugins are usually lib-friendly). That shouldn't be bad for BLITZ too.