Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 09:18:12 GMT
View Forum Message <> Reply to Message

Why need dli? You already have all functions from #include <windows.h>. The trouble would only be explicitly calling A and W version. Thinking of it, it sounds nice - make PLATFORM_UNICODE a global boolean, initialized to true, unless OS is Win9x. But I'd prefer to finish the way I started to see everything work.

UTF-8 -> UTF-16 -> UTF-8 won't happen. FromFileCharset returns String if it's ASCII/UTF-8 and WString if it's UTF-16. It returns amount of bytes. 0 -> ASCII / String, 1 -> UTF-8 / String, 2-> UTF-16 / WString (4 -> UTF-32 / WString, but not implemented). What could happen is UTF-16 -> WString -> String, but UTF16 -> WString isn't expensive.

I wanted to compile UWord (now in ANSI, GUI Unicode isn't complete yet) to see if zlib work (UWord.iml), and found an interesting problem in PdfDraw:

ScreenDraw sd;

That causes a warning of statement is a reference not a function call. + error about sd definition. In Draw/DrawWin32, ScreenDraw is a class, but also:

```
ScreenDraw& ScreenDraw()
{
 return Single<ScreenInfoClass>();
}
```

That's a singleton? Whatever it is, it doesn't work - ScreenDraw sd; is recognized as a function name, not class type. Any suggestions how to fix?

P.S. Why does U++ use so many global functions? I prefer .Net-style - tree-like organization using namespaces/classes. After all, gathering functions into static classes should be realtively easy, and at the cost of some extra typing you (potentially) resolve naming conflicts, and make stuff easier to find. E.g. I may not know that there's a function named GetWinRegString hidden somewhere in Core/Win32Com. But if there was a class Registry, it would be more likely that I'd find it by typing Registry::. Plus that would be an OOP approach