Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sat, 15 Sep 2007 19:22:00 GMT
View Forum Message <> Reply to Message

Have you tried using MS Unicode Layer for Win9x? I never did, but from the description it doesn't look much complicated. That way all builds could be Unicode.

Actually TCHAR is supposed to work. I started to do it, didn't have the chance to complete (see previous post), but in theory my solution should just work.

I assumed that String always contain UTF-8, and WString always contain UTF-16. I also assumed that String.ToWString and WString.ToString convert between them properly. And I used Win32 functions - MultiByteToWideChar and WideCharToMultiByte (have no idea how stuff like that works on Linux).

TSTR is a class I made, that would be an interface between String/WString and Win32 functions taking strings. Example:

String s = "text"; WString ws = "cap";
MessageBox(NULL, TSTR(s), TSTR(ws), MB_OK);

If UNICODE is defined, MessageBox would be MessageBoxW, and TSTR would transform String and WString to WCHAR*. Otherwise MessageBox would be MessageBoxA, and TSTR would transform both to char* (ANSI charset, removing unknown characters). This should compile just fine on Win9x without UNICODE defined, and on WinXP with/without UNICODE defined.

In the file there's also UTFBOM, since I couldn't type in TheIDE (thanks for telling how) I wrote some file handling.
Usage (read any encoding with BOM into UTF-8 string without BOM):

String ftxt = LoadFile("File.txt");
String sf; WString wsf;
int res = UTFBOM::ReadBOM(ftxt, &sf, &wsf);
String strUTF8;
if (res < 2) strUTF8 = UTFBOM::WriteBOM(sf, 1, false);
else strUTF8 = UTFBOM::WriteBOM(wsf, 1, false);

There are probably bugs in the files (couldn't test thoroughly), yet I believe the concept is correct.

Hebrew is indeed RTL. RTL by itself isn't complicated. Flip your screen vertically - that's pretty much what you get. Fully RTL indeed flips icons, minimize|maximize|close becomes close|maximize|minimize on the left side of the screen. Cursor advances to the left when typing. Home key brings cursor to right, End to left (in LTR Home goes left and End goes right). Backspace deletes symbol to the right, Delete deletes symbol to the left. Thinking of it now, if you flip LTR screen, the only thing that will be different from RTL screen is the left/right keys. That is, left remains left and right remains right. Only that in RTL right actually goes towards beginning of

text, not end.

Now, that was the correct behaviour, at least what I'm used to (MS Word is quite good in Hebrew). Problems arise when combining RTL with LTR, especially if you also insert "neutral" symbols like !,.?
Then most text/word processors go crazy and results are rather unpleasant. MS Word is good, yet it isn't perfect.

(LTR TEXT)|(RTL TEXT)

| is the cursor. Press right - you'll probably get the cursor about here:
(LTR TEXT)(RTL TEX|T)

Because of the direction change. That also depends on paragraph orientation - whether the paragraph is RTL or LTR makes a difference. The behavior of Left/Right/Home/End/etc. keys is usually like that only in RTL paragraphs, and would be weird in LTR paragraph just like English would type weird in RTL paragraph (try it - right Ctrl+Shift in a textbox). Not exactly intuitive, and I'm not sure if that's standards-conforming behavior, but that often happens.

Does U++ work right with RTL-only (single language)? E.g. correct Left/Right/Home/End/Backspace/Delete behavior? That shouldn't be so difficult to implement.

P.S. please tell me what can be done about source packages. I don't know whether these are indeed mistakes, or I'm again doing something wrong.

File Attachments

1) TStr.h, downloaded 489 times