Subject: Re: 16 bits wchar
Posted by cbpporter on Thu, 25 Oct 2007 12:47:29 GMT
View Forum Message <> Reply to Message

I also wrote a conversion algorithm from Utf8 to Utf16, which is quite similar to my previous one.

Since I already done these, I would like to optimize them a little. I have a couple of questions though.

1. My code point extraction routine is a little to long and quite redundant. The same sequence that handles an incorrect value is called a lot of times. I would like to get rid of these repetitions. I could use a macro, but I don't like to expose a dangerous macro to the rest of the file, so I could undef it after the function. Or, this would be the perfect case to where the use of goto could be justified and almost needed. Can I use goto?

EDIT:
P.S.: The modifed version of utf8codepointEE optimized and using goto is 49 lines long, while the original was 115. And I still consider it pretty clear, maybe even more clear because I can it on one screen.

2. The algorithm is a little bit inefficient because it first calculates the length of the new buffer, and then it fills it. But by calculating the length, we already get enough info to populate it with the correct values, reducing the number of calculations by half. But if I do this, I would need to preallocate first a possibly bigger than necessary buffer, fill it directly, than copy it in the new string and free the buffer. This has one extra allocation, and I'm not sure how efficient allocations are in U++. AFAIK, you replaced the default allocator. If it has similar efficiency as the standard one, the price of the allocation is not that large, but maybe you are against this approach as it more STL like, with allocating a lot of extra data and doing copies. If the allocator is faster, or if you have a caching mechanism, than I think that it could be a lot faster this way.