Subject: Re: 16 bits wchar
Posted by mirek on Sat, 27 Oct 2007 09:11:46 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Thu, 25 October 2007 08:47I 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?

Of course. I have no problem with using anything in IMPLEMENTATION. I believe that the main task is to keep interfaces clear. If goto or macro are able to speedup or simplify things, go for it. No need to undefine macro either, as long as it is used in .cpp only.

(The only things I would ask in implementation: If you decide to use platform/machine/CPU specific things like "asm", be sure to provide crossplatform "default" implementation, or at least implement it for all supported platforms).

Quote:
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++.

I believe they are quite efficient. Most of time, about 20 CPU instructions have to be executed to allocate a memory block.

However, I am a little bit afraid that the "copy" will make it inefficient...

OTOH, IME, guessing never helps to resolve optimization issues. If you really want to play hard, benchmark

Also consider putting StringBuffer to the mix as well.

Quote:
If the allocator is faster, or if you have a caching mechanism, than I think that it could be a lot faster this way.

Well, it is as fast as to make the STL idea of speed optimized allocators in container templates obsolete

Mirek