Subject: Proposal: optional int64 size/indexes
Posted by Mindtraveller on Sun, 16 Aug 2015 08:08:34 GMT
View Forum Message <> Reply to Message

Hi,

Looking into U++ sources I've found many places where indexes of size variables are represented by int type members.
As we all know, 64-bit hardware is more and more widely used, so x64 support is crucial for U++. Yes, U++ is succesfully built with all latest C++ x64 compilers including MSVC, GCC and CLANG/LLVM. But 64-bit compatibility is more than just compilation. We need support for large indexes, large addresses and large sizes inside U++ containers and classes.

For example, FileMapping maps file using int64 as size (which is perfect), but FileMapping offset routines use int variable as index. Obviously, it doesn't support int64 indexes out-of-the-box.
The same is for U++ containers and other classes.

It would be too easy to say "OK, let's just switch to int64 everywhere we can". But it is not the best way.
Just because efficiency is a target. And more of that, U++ is used on embedded systems where simple int is preferrable.

What I propose is two possible options:

1) To add more int64-based member functions for U++ containers and classes.
For example: operator[](int64), etc.

2) To use something like INDEX typedef for internal U++ indexes and sizes to switch between int and int64. The switching is done with compilation flag like INT64SIZE.

Thanks
Pavel