Subject: Re: Cannot Compile TheIDE with MSVC12
Posted by mirek on Wed, 28 Jan 2015 19:25:20 GMT
View Forum Message <> Reply to Message

Silvan wrote on Tue, 27 January 2015 11:42What are icpp files?

Well, but now you have a new project, using U++ with Visual Studio, are you? :)

OK. Well, the .icpp is a problem there, one that init files are trying to solve.

The idea is this: We want some code to be 'autoregistered'. E.g. Draw module contains general interface for loading raster graphics. Then we have a module that loads particular image format, say 'tiff'. Now we want, when we add this module to the project, to get registered with Draw module, so that we can use "LoadImageDetectFormat" sort of function. And we want that to happen without calling some "RegisterTiffWithDraw" in main, we want to happend that automagically.

To that end, we are using global constructors (actually, we have a nice macros INIT_BLOCK/EXIT_BLOCKS that create a piece of code that gets run at the start and at the end of code. Those macros are implemented using C++ global constructors/destructors). But here comes the problem: All this only works when object file is linked into final .exe. And when we are building .lib, linker excludes object files that are not referenced from the rest of code. Unfortunately, that usually includes our registration code. Means that with usual building rules, global constructor/destructor trick does not work.

That is why we have invented ".icpp". This is like regular .cpp, but has guaranteed to be included in final binary - U++ build system understands this extension and takes appropriate steps to ensure that. Problem solved (as long as you are using theide or umk), problem created for Visual Studio...

'init' files are attempt to solve this issue.

Consider plugin/tif. It contains registration .icpp:

```
#include "tif.h"

NAMESPACE_UPP

INITBLOCK {
 StreamRaster::Register<TIFRaster>();
}

END_UPP_NAMESPACE
```

And then autogenerated (by theide) 'plugin/tif/init':

```
#ifndef _plugin_tif_icpp_init_stub
#define _plugin_tif_icpp_init_stub
#include "plugin\jpg/init"
#define BLITZ_INDEX__ F06a388f1e84b680d94787428bf67e5bb
#include "tifreg.icpp"
#undef BLITZ_INDEX__
#endif
```

As you can see, this code includes .icpp files, but also includes 'plugin/jpg/init'. That is because plugin/tif uses plugin/tiff (has it as 'uses' dependency). Now the idea how is this going to help with Visual Studio is that you can build everything as usual, only building .cpp, and in your main.cpp file, you would include all "init" files of directly dependent packages, like

main.cpp

```
#include <CtrlLib/CtrlLib.h>
#include <Something/Something.h>

// This part is only for visual studio:
#include <CtrlLib/init>
#include <Something/init>
// Includes in fact whole cpp files, thus must be include only in single .cpp file
```

Now, that is the nice theory, but AFAIK, nobody really tried this in practice... Be first! :)

Mirek
