Subject: Building & using U++ without TheIDE
Posted by sergei on Sun, 09 Sep 2007 20:03:06 GMT
View Forum Message <> Reply to Message

Hi! I wasn't sure where this belongs, so I posted here.

I was looking for a good cross-platform GUI instead of MFC/WTL/Win32API. Found many alternatives (wxWidgets, Qt, FOX, FLTK, VCF), but U++ seems to be closest to what I want - opensource, relatively light, but with enough features, and using C++ style (not macros).

The thing is, I just couldn't build the library. I'm a newcomer to opensource, but still I managed to build and use wxWidgets, FOX, FLTK (no success with VCF though). Using the default setup, programs work within TheIDE, but I want to work with Code::Blocks + MinGW, or maybe MSVC.

Might be a bit offtopic, but I'll say what was the first experience. While I've read people turn U++ down due to the way the code looks, I'll say that's the reason I want to use it. I like pointers (used them in 2D graphics, and implementation of data structures), but these simply don't belong to interface design - it should be nice, clean class code. Just the way it is in U++. Other toolkits just have lots of pointers to stuff an news/deletes everywhere.

But TheIDE and the layout designer are quite a disappointment (sorry, but that's my opinion). Build system is interesting, but the interface itself seems very unmature. Same goes for layout designer (especially inserting controls through right click, and not having splitter there). Although I might just be spoiled (VB / VB.Net have excellent IDEs and designers). I'd suggest nicer icons, a treeview to show packages and files, and a toolbox for widgets (all of them if possible).

So, I would like to use only the library part of U++, not TheIDE and layout designer. I tried to just include it, the way I did with WTL, but it didn't work - U++ has CPP files (WTL consists of several H files). I've read in another thread that the code cannot work as a DLL the way it is, so I tried to build a static library. That didn't work too - lots of warnings, and some errors. Started with some weird error about String's << operator (which I managed to fix), it continued with ambigous names (GetLocaleInfoA/W is name of Win32 API and of String's functions), and then it couldn't find classes definitions/declarations. That led me to finding that some CPPs (like Locale in Core) didn't have H files, and the classes they implemented were in H's of different names. I'm not sure just how legal is that from C++'s point of view, but I have no idea how to compile that.

Could anyone help me with building a static/dynamic U++ library without using TheIDE? I would really like to use U++ due to its potential, but there isn't much sense gaining cross-platformness at the cost of losing IDE/compiler freedom. If a static library / DLL would be available for download, I guess people would be more comfortable at trying U++ out with the IDE/compiler they're used to (IMHO most difficult part of using open-source projects is getting them to start working). Installing U++ isn't difficult, but TheIDE is very confusing at first.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sun, 09 Sep 2007 21:47:32 GMT

Search the forum, there was quite a lot of discussion about this topic.

Generally, it is possible to build U++ apps using generic make. So far, it is not possible to use other build systems to directly build it because of .icpp issue.

DLL version might not be the biggest problem; after all you can always use some tool to generate .def file.

My take is that library version is definitely possible, perhaps with some workarounds that to deal with things that theide build system deals with automatically.

What is missing is a devoted maintainer of such version; the problem would most likely involve rebuilding the tree to more classic form, maybe even using some automated source-changing procedures (like inserting registration routines as .icpp replacement).

The first iteration: You can use theide to get .obj files of everything, then simply gather them and make libraries...

Of course, the fundamental question is how you would reorganize U++ packages into perhaps bigger .lib files. If all you need is a single .lib per package, you can even generate .lib directly, but I am afraid that would yield too much .libs...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 09 Sep 2007 23:48:38 GMT

Thanks for the quick reply.

I did search the forum previously, all I found was outdated MSVC solution files. Now I've seen your new post about icpp, previously I didn't even understand what they were.

I renamed icpp to cpp, inserted dummy functions, but then I don't think these were the cause of my problems (yet?). Static libraries build every cpp, and include every h included, right? (that would explain locale.cpp with no locale.h).

I might be missing something here. Here's what I'm doing:

1) In Code::Blocks (recent nightly build) using MinGW (GCC 3.4.5), I created a static library project.
2) Copied everything from uppsrc folder to project folder, added all cpp (including icpp, renamed to cpp so they get compiled), h and rc.
3) Defined PLATFORM_WIN32 in project options.

4) Ran build all. Got a bunch of warnings and an error.

Warning are mostly signed/unsigned. One was more serious:
Core\Util.h:283: warning: `class Upp::CharFilterTextTest' has virtual functions but non-virtual destructor

Error:

CbGen\CppGen.cpp:: In function `void CallbackGen(Upp::String, Upp::String, int, Upp::String, Upp::String)':
CbGen\CppGen.cpp:27: error: no match for 'operator<<' in 'String(((const char*)"template <class OBJECT, class METHOD")) << If(Upp::String, Upp::String)(Upp::operator+(const char*, const Upp::String&)(((const Upp::String&)((const Upp::String*)(&classdef)))))'

That was on:

String cl_temp = String("template <class OBJECT, class METHOD") << If(classdef, ", " + classdef) << ">";

I changed that to (not sure if correct, just wanted to continue):

```
String cl_temp = String("template <class OBJECT, class METHOD");
 cl_temp <<= If(classdef, ", " + classdef);
 cl_temp <<= ">";
```

Tried to rebuild. Even more warnings - signed/unsigned, and virtual for PrintPageDraw. It also found that:

Item& AddMenu(const String& t, const UPP::Image& m, Callback c) { AddMenu(t, m, c); }

Didn't return a value though it should.

And again an error. This time in Core\t.h:

```
INITBLOCK_(COMBINE3(LNG_MODULE, LNG_VERB, LNG_VERA))
{
 static LngEntry__ langset[] = {
 #include TFILE
  { 0, NULL }
 };
#ifdef LNGMODULE
 AddModule(langset, #LNGMODULE);
#else
 AddModule(langset);
#endif
}
```

LngEntry__ and AddModule are undefined.

Included <Core/i18n.h> to this file. Looks like it helped.
Next error: String undefined in i18n, including <Core/String.h> helped.
Next - seems like due to my modifications String got included more than once. And looks like it doesn't have include guards.

OK, I might be able to go on "fixing" the source, but I guess something is wrong if I'm getting all these errors. Or maybe it's due to BLITZ putting all files together and thus solving dependancy problems...

I'd be glad to hear if there is an easy way to solve this. I didn't find a makefile to build U++. I guess taking *.o files from TheIDE could be a solution, but I'd prefer to be able to build a library without TheIDE.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 08:16:33 GMT
View Forum Message <> Reply to Message

Ah, I see.

This is the trouble - you are trying to compile everything. CbGen is not part of library, it is utility. This way, you would compile "ide" too into the library...

Actually, this is one of problems I have mentioned in previous post. TheIDE makes everything highly modular, beyond the capabilities of other systems (that was the primary motivation for theide).

If you want to "fix" things to work as libraries, you will have to choose how to reduce this modularity.

IMO, you will have to merge several packages into single .lib.

Possible starter: Load "examples/UWord" into theide and check what packages it uses. These are reasonable minimum for GUI development.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 08:30:38 GMT
View Forum Message <> Reply to Message

Also, activate SetUp/Verbose to find out what and how is really going to be compiled (will print compiler commandlines).

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 10 Sep 2007 10:25:16 GMT
View Forum Message <> Reply to Message

Actually the idea was to make one big lib of everything. I guessed it would be about 2MB or so, which IMHO is acceptable as an exe size. Linking to the lib and including whatever I need (e.g. CoreLib/CoreLib.h) would give me access to the library.

I removed some unnecessary stuff from building. I also realized that there is no documentation on what file / folder is what - and that sources are uncommented.
Now I'm building (folders): Core, Crypto, CtrlCore, CtrlLib, Draw, DropGrid, Geom, GLCtrl, PdfDraw, plugin, RichEdit, RichText, Web.

Still, extra files weren't the problem. It just skipped the previously first errors, right to the t.h and i18n.h undefineds. BLITZ problem or not, I just can't get the structure of the sources right. I set former icpp-files priority higher than the rest to make them compile first, but that didn't change a thing. I tried pragma once and ifdef guards, but no use. At least one thing is undefined/redefined.

I think the structure of the CPP/H is the root of the problem. I'm used to the classic CPP+H pairs, plus a main file. There's also a purely-H system (like STL and WTL). This structure is neither of them - there are CPPs without H, Hs without CPP. Moreover, any given file doesn't include its dependencies (e.g. i18n doesn't include String, although it uses it).

Is there any reason for such a structure? Can it be modified to be more "compileable"? If there are no further problems I'm unaware of, it should be possible to restructure most of the source within several days (all headers with pragma once, and include all dependencies). I could try to do it, if it would work and there is no better solution. Ideally, I'd wish it were a set of headers for immediate use, but since CPP are necessary, static library is just fine.

P.S. some warnings in non-core files (e.g. no return in AddItem) do seem to be errors, no?

P.S.2 why do files in plugin folder use short includes (#include "bmp.h" and not #include "plugin/bmp/bmp.h")?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 11:56:49 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 06:25
I removed some unnecessary stuff from building. I also realized that there is no documentation on what file / folder is what - and that sources are uncommented.

That is not quite true:

http://www.ultimatepp.org/srcdoc$Core$Packages$en-us.html

While documentation lacks & lags, most of GUI stuff and Core is actually documented. However, we are using TheIDE's ability to bind rich text documentation with the code, so instead of comments it is in separate files (just press the Help button).

Quote:
Is there any reason for such a structure?

Definitely It is the most optimal structure for very large projects.

Quote:
Can it be modified to be more "compileable"?

Do you in include path the root of package nest, in this case "uppsrc"? (like c:\upp\uppsrc ?).

Quote:
P.S. some warnings in non-core files (e.g. no return in AddItem) do seem to be errors, no?

AddItem was error, thank you, already fixed.

Quote:
P.S.2 why do files in plugin folder use short includes (#include "bmp.h" and not #include "plugin/bmp/bmp.h")?

OK. The idea is that usually, the package has the "main include", which is usually named the same as the package, e.g. CtrlLib/CtrlLib.h. Anyway, puttin all the stuff into single file would make it too big, that is why it is divided to parts.

But for .cpp files, the simplest way how to do #include is to use this main package header too.

#include "xxx.h" searches for the header in the same directory (as opposed to #include <xxx.h>) Less typing....

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 10 Sep 2007 13:53:53 GMT
View Forum Message <> Reply to Message

Include stuff is interesting. Does that mean that any #include <> could be replaced with #include ""?

I didn't add to include uppsrc dir. But I don't think I should - it's also the root of the project in Code::Blocks, so compiler should be able to find whatever it needs, right? Or maybe I have to since #include <> and not #include "" is used in many sources?

I don't quite understand why such a structure is optimal for very large projects (especially if precompiled headers exist), but my problem with it is that it seems (to me, at least) incorrect. Not in terms that I don't like the style, but due to the fact that there are files (headers) that use classes/structures defined in other files (headers), without including them. t.h uses LngEntry__ defined in i18n.h without including it. i18n.h uses String class without including Core/String.h (actually i18n.h only includes the miniature t_.h, but it also uses FileOut defined in Core/Stream.h). AFAIK this means that these files are supposed to be compiled later than the files declaring classes they use, but without referencing them through include, how is the compiler supposed to know that?

It's the first time I encounter such coding style. Are there other (more simple) examples of relying on existance of classes (without explicit including)? It just doesn't seem to be correct C++ coding (does it conform to C++ standards?).

And what exactly is the problem with the classic CPP+H pairs? Just headers (like WTL) is problematic due to, well, huge headers. But why not create a pair of CPP and H for every class / several classes, include the H in the CPP, and in the H use guarding ifndefs, and include all dependencies? Any header will be included at most once, and only the necessary pairs will be used.

One more thing, does BLITZ solve all these problems by finding what needs what and placing it in the correct order? Where could I find info on BLITZ (Google thinks I'm looking for Blitz Basic).

Subject: Re: Building & using U++ without TheIDE
Posted by Ulti on Mon, 10 Sep 2007 14:56:28 GMT
View Forum Message <> Reply to Message

If you like upp,then it is better to use TheIDE,I like upp too,but always feeling something missing.NTL? new theory?I don't know,but it is easy for using.
I hope someday I understand upp well,then I will try to fork a STL version upp.

Subject: Re: Building & using U++ without TheIDE
Posted by Zardos on Mon, 10 Sep 2007 15:17:06 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 15:53I don't quite understand why such a structure is optimal for very large projects (especially if precompiled headers exist), but my problem with it is that it seems (to me, at least) incorrect. Not in terms that I don't like the style, but due to the fact that there are files (headers) that use classes/structures defined in other files (headers), without including them. t.h uses LngEntry__ defined in i18n.h without including it. i18n.h uses String class without including Core/String.h (actually i18n.h only includes the miniature t_.h, but it also uses FileOut defined in Core/Stream.h). AFAIK this means that these files are supposed to be compiled later than the files declaring classes they use, but without referencing them through include, how is the compiler supposed to know that?

The trick is simple:

1.) Only .c / .cpp / ... files gets compiled.
2.) The exists a single central header filer for example. "Core.h"
3.) .cpp /.c /... files only include the central header file: Core.h
4.) .h / .hpp - files Include NO header files.

Imagine there would exists only one header file: Core.h
All cpp files only include this header file.

=> You don't have to include any other header files in a hedaer file even if you are using class and functions from other files in this header file, because these files don't get compiled. Only the cpp files and these include the central header file Core.h containing ALL classes / functions etc.

One important thing fur such a structure is the order of the included header files in Core.h, You have to include the most basic stuff not depending on other files. And step by step you climb down the dependency tree.

You don't need any include guards because, you only include each header file once in the central header file. And the other header files includes NOTHING!

After you have really understood the idea you might find the concept interesting, too. There are some obvious advantages: For example you don't have to think what you have to include in each ".h", ".cpp." files. Just include the central header in the cpp files and nothing in the header files.

This has nothing to do with BLITZ, btw.

But I prefer the "standard including style" , too

- Ralf

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 10 Sep 2007 15:57:54 GMT
View Forum Message <> Reply to Message

Oh, so that's what this is about. Thanks for the info. Quite similar to the approach with stdafx.h and precompiled headers in MSVC. Advantage is rather obvious, but so is the disadvantage - you take on yourself the job of defining compile order, and I'm not sure if that's easier than defining dependencies. Not sure if it affects build time (there shouldn't be any difference IMHO).

So I'll try to guess what the compiler does:
1) Pick a random CPP.
2) See that it needs Core.h, go to Core.h
3) Go from first to last include
4) For every include, define stuff in H and compile stuff in CPP
5) When done, Core.h and most CPPs will be compiled, and rest could be finished
That's the way it should work? If so, in a static lib there's no need for dummy functions in icpp, since nothing is thrown out, right?

Ok, let's see what's in Core.h:

#include <algorithm>
#include <string>
I thought U++ replaced STL with own containers...

I see that all includes are ordered, so it should've worked...
I'll try again at home, but I've found that XML.h, that is included just before Lang.h and i18n.h that I have trouble with, has #include <Core/Core.h>. XML.cpp includes "Core.h" too.
Why not add #pragma once to all H, just in case?

There are also some minor includes inside includes (like AString.hpp and t_.h), but these probably don't affect the big idea.

P.S. what's the difference between #ifdef and #if defined()? Both are used in Core.h.

P.S.2 I've found the tpp help files, is there any external viewer?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 15:57:55 GMT
View Forum Message <> Reply to Message

Zardos wrote on Mon, 10 September 2007 11:17
This has nothing to do with BLITZ, btw.

Well, actually, a little it has: with BLITZ it does not too much matter how much you include w.r.t. compilation speed, means the old rule "include as little as possible to keep compilation fast" is

irrelevant...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 16:09:59 GMT
View Forum Message <> Reply to Message

Sergei, .h never get compiled (in any C++ build system) and except some minor differencies (BLITZ and .icpp), U++ build system, w.r.t. compiler, is not really too different from any other build system.

C++ compiler is the same, rules for #include are defined by C/C++ standard (as are rules for #include "" vs #include <>, although these can be a little bit implementation specific).

It is therefore somewhat pointless to try to find out what "U++ compiler does", it is regular GCC or MSC compiler and it does nothing else than with any other C++ code.

My bet is that the main problem of your attempt is wrong include path.

Also, compile .c/.cpp/.icpp files only (I am not sure how codeblocks deals with .h files, if it invokes the compiler for them, you are certainly in trouble).

Quote:
#include <algorithm>
#include <string>
I thought U++ replaced STL with own containers...

Yes, that is correct, but people require certain level of compatibility. You can use STL algorithms with NTL (as long as elements satisfy STL requirements). Upp::String / Upp::WString have conversions from/to std::string.

Quote:
#pragma once

It is MSC specific, not defined by C++ standard. Main headers use normal define guards.

.hpp files are somewhat special, using this extension for template implementation stuff. Once again, this is about dividing long headers and grouping the stuff.

"#ifdef" and "#if defined" is the same thing.

.tpp: Just click the help button. Magenta button with question mark.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 16:12:02 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 11:57P.S.2 I've found the tpp help files, is there any external viewer?

Sorry, failed to notice "external viewer". Well, there is none (yet), but everything is exported to u++ website:

http://www.ultimatepp.org/www$uppweb$documentation$en-us.htm l

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 16:15:46 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 11:57I've found that XML.h, that is included just before Lang.h and i18n.h that I have trouble with, has #include <Core/Core.h>. XML.cpp includes "Core.h" too.

This include is a sort of benign bug, it was forgoten when moving the header from development package to Core. #include guards of Core.h made it effectively a NOP....

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 10 Sep 2007 18:59:29 GMT
View Forum Message <> Reply to Message

OK, this set some things straight. So I can think of BLITZ simply as a build time optimizer.

I made some progress. I "solved" the icpp issue by adding this main.cpp file:

```
#include "Core/Core.h"

#include "RichEdit/RichEdit.icpp"
#include "RichText/RichImage.icpp"
#include "Web/Web_init.icpp"
#include "plugin/bmp/BmpReg.icpp"
#include "plugin/gif/gif.icpp"
#include "plugin/jpg/jpgreg.icpp"
#include "plugin/png/pngreg.icpp"
#include "plugin/tif/tifreg.icpp"
```

#include "PdfDraw/PdfReport.icpp"
#include "CtrlLib/CtrlLib.icpp"
#include "Geom/Ctrl/GeomCtrl_init.icpp"
#include "Core/Core_init.icpp"
#include "CtrlCore/CtrlCore.icpp"

I also set it to high priority, to force it to compile first. Some of these icpp don't include Core.h first, which might be a problem, but since Code::Blocks only compiles C and CPP, it isn't.

This yielded an error of INITBLOCK being defined twice (actually, something inside it). Sounds reasonable, same macro used in both GIF and BMP icpp-s.

Reordering includes in main.cpp to:

#include "Core/Core.h"

#include "Core/Core_init.icpp"
#include "CtrlCore/CtrlCore.icpp"
#include "CtrlLib/CtrlLib.icpp"
#include "Geom/Ctrl/GeomCtrl_init.icpp"
#include "RichText/RichImage.icpp"
#include "RichEdit/RichEdit.icpp"
#include "PdfDraw/PdfReport.icpp"
#include "plugin/bmp/BmpReg.icpp"
#include "plugin/gif/gif.icpp"
#include "plugin/jpg/jpgreg.icpp"
#include "plugin/png/pngreg.icpp"
#include "plugin/tif/tifreg.icpp"
#include "Web/Web_init.icpp"

Resulted in the old error of LngEntry__ in t.h being undefined. But that struct is defined in Core/i18n.h, which is included in Core.h. Main.cpp is the first file compiled, so Core/Core.h should've been included. Weird.

And I did add project's dir to search directories of the compiler.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 10 Sep 2007 19:31:23 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 14:59OK, this set some things straight. So I can think of BLITZ simply as a build time optimizer.

I made some progress. I "solved" the icpp issue by adding this main.cpp file:

#include "Core/Core.h"

```
#include "RichEdit/RichEdit.icpp"
#include "RichText/RichImage.icpp"
#include "Web/Web_init.icpp"
#include "plugin/bmp/BmpReg.icpp"
#include "plugin/gif/gif.icpp"
#include "plugin/jpg/jpgreg.icpp"
#include "plugin/png/pngreg.icpp"
#include "plugin/tif/tifreg.icpp"
#include "PdfDraw/PdfReport.icpp"
#include "CtrlLib/CtrlLib.icpp"
#include "Geom/Ctrl/GeomCtrl_init.icpp"
#include "Core/Core_init.icpp"
#include "CtrlCore/CtrlCore.icpp"
```

I also set it to high priority, to force it to compile first. Some of these icpp don't include Core.h first, which might be a problem, but since Code::Blocks only compiles C and CPP, it isn't.

I am really not sure what priority is, but I bet it does not matter.

Quote:
This yielded an error of INITBLOCK being defined twice (actually, something inside it). Sounds reasonable, same macro used in both GIF and BMP icpp-s.

This is caused by the fact that INITBLOCK synthetises the name of some static variable based on the line number -> by including them into the single file, you are getting them defined twice...

I think the only correct solution is to add empty Init function to .icpp file (InitPluginPng) and call it from main. And rename .icpp to .cpp...

(Alternatively, you can create helper .cpp that includes .icpp and has this function - see technology section).

Quote:
Resulted in the old error of LngEntry__ in t.h being undefined. But that struct is defined in Core/i18n.h, which is included in Core.h. Main.cpp is the first file compiled, so Core/Core.h should've been included. Weird.

What is the actual error log?

BTW, you do not have to be so much obsessed about order of including - it does not really matter, as long as you include "highest-level" header for your application.

For GUI only, #include <CtrlLib/CtrlLib.h> is all you need. If you are doing SQL GUI app, #include <SqlCtrl/SqlCtrl> is all required (will include CtrlLib.h and CtrlCore.h and Draw.h and Core.h and

Sql.h...)

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 10 Sep 2007 20:35:29 GMT
View Forum Message <> Reply to Message

OK, I still don't understand the purpose of icpps. Some have extra includes, some INITBLOCKS and functions. Wouldn't it be easier to make one global function like InitUpp that would do the registers, and call it in GUI_APP_MAIN? I'm working on a static lib project, so empty functions probably don't matter - nothing should be thrown out in static lib.

Priority (in Code::Blocks at least) means that this CPP will get compiled earlier than the rest. In static lib there is no main CPP which contains function main, so I prioritized main.cpp to ensure it is the first one compiled.

I actually manged to solve the Lng error (or did I?). t.h was missing NAMESPACE_UPP at top and END_UPP_NAMESPACE at bottom.

Now, next errors:
#include <TCtrlLib/TCtrlLib.h> (in Geom/Ctrl/GeomCtrl.h)
#include <TCore/TCore.h> (in Geom/Coords/GeomCoords.h)
Didn't find these files in the sources.

Then RefBase is undefined somewhere in Geom (whole package seems problematic - what is it for - IDE or something else?) - that RefBase is defined in Web/util.h - how's that related to geometry...

I removed Geom from project to see if I can continue - then I reached, once again, the INITBLOCK. Assuming I give up modularity, and want to build a single lib containing everything, except from stuff needed only for TheIDE, can I do something about the icpps? I mean, there has to be a nicer solution than using INITBLOCKS in different CPPs so they don't see each other...

P.S. the documentation is incomplete, how can I know what is used for TheIDE and what isn't? UWord uses only a small subset, e.g. Sql, Web, etc. are part of the library but aren't used.

P.S.2 what do I need to define besides PLATFORM_WIN32? DEBUG? UNICODE?

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 11 Sep 2007 00:21:35 GMT
View Forum Message <> Reply to Message

Followup: without completely understanding why and how INITBLOCK works, I replaced it in icpps

with INITBLOCK_(BLK_###), where ### is the name of the file. It worked, and now I'm stuck on:

D:/Dev/0/1/libUpp/Core/Topt.h: In function `void Upp::AssertMoveable0(T*) [with T = Upp::wchar]':
D:/Dev/0/1/libUpp/Core/Topt.h:223:   instantiated from `void Upp::AssertMoveable(T*) [with T = Upp::wchar]'
D:/Dev/0/1/libUpp/Core/Vcont.h:89:   instantiated from `Upp::Vector<T>::~Vector() [with T = Upp::wchar]'
D:/Dev/0/1/libUpp/Core/Vcont.h:88:   instantiated from `Upp::Vector<T>::~Vector() [with T = Upp::Vector<Upp::wchar>]'
D:\Dev\0\1\libUpp\PdfDraw/PdfDraw.h:330:   instantiated from here
D:/Dev/0/1/libUpp/Core/Topt.h:214: error: invalid type argument of `unary *'

According to NTL requirements, wchar should be moveable. But that Assert fails.

```
template <class T>
inline void AssertMoveablePtr(T, T) {}

template <class T>
inline void AssertMoveable0(T *t) { AssertMoveablePtr(&**t, *t); }
```

T is wchar. That would be: &**(wchar*). **(wchar*) doesn't make sense. How does that check if the type is moveable?

Subject: Re: Building & using U++ without TheIDE
Posted by Zardos on Tue, 11 Sep 2007 07:25:51 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 10 September 2007 17:57Zardos wrote on Mon, 10 September 2007 11:17
This has nothing to do with BLITZ, btw.

Well, actually, a little it has: with BLITZ it does not too much matter how much you include w.r.t. compilation speed, means the old rule "include as little as possible to keep compilation fast" is irrelevant...

Mirek

Ah! I see your point! I think I change my position and find the used include style pattern, great - now!

Another advantage with BLITZ is: It might produce better code in "release mode".
I know it was written here in the forum that code produced with BLITZ is often larger than without BLITZ.
This can be considered a good thing (sometimes)! It probably means that the compilers can do more agressive inlining even on function calls that goes across different cpp files and that are not

explicit declared as inline.(With BLITZ there exists basically only one file which gets compiled). Well, some compilers (MSVC) can perform "Global optimization" across different object files. But I always have some doubts if they can use the same sophisticated optimization heuristic tthey use in a single file - optimize register allocation and all these nice things.

If I remeber right on the webpage on SQLite they claim a performance improvement of some percent (10?) if it gets compiled with the "one source file" version of it.

Unfortunately BLITZ has some problems on some packages in release mode. May be a another flag "No BLITZ in release mode" would help for some projects.

- Ralf

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 11 Sep 2007 20:16:32 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 16:35OK, I still don't understand the purpose of icpps. Some have extra includes, some INITBLOCKS and functions. Wouldn't it be easier to make one global function like InitUpp that would do the registers, and call it in GUI_APP_MAIN?

Keep in mind that this is modular system. The purpose of .icpp is to register modules that are not known at the time when registration procedure is created.

E.g. you can add new image formats, in the form of plugin package, this way and simply adding this new package to your project makes StreamRaster::LoadFileAny aware of this format.

Quote:
I'm working on a static lib project, so empty functions probably don't matter - nothing should be thrown out in static lib.

Sure the do matter. Linker only includes .obj from .lib that are referenced. That is the main and the only difference for .icpp - they are linked as .obj so they are always included.

Quote:
I actually manged to solve the Lng error (or did I?). t.h was missing NAMESPACE_UPP at top and END_UPP_NAMESPACE at bottom.

Means something else is wrong elsewhere: t.h is widely used without this problem.

Quote:
#include <TCtrlLib/TCtrlLib.h> (in Geom/Ctrl/GeomCtrl.h)
#include <TCore/TCore.h> (in Geom/Coords/GeomCoords.h)

You do not seem to listen...

Please, check UWord example in theide first.

If you are so much afraid about starting theide, ok. There are .upp files in all packages and they create a dependency tree. Check what is really needed for GUI application.

Quote:
I removed Geom from project to see if I can continue - then I reached, once again, the INITBLOCK. Assuming I give up modularity, and want to build a single lib containing everything, except from stuff needed only for TheIDE, can I do something about the icpps? I mean, there has to be a nicer solution than using INITBLOCKS in different CPPs so they don't see each other...

Rename to .cpp and insert referencing emty function. Call these in main.

Quote:
P.S. the documentation is incomplete, how can I know what is used for TheIDE and what isn't? UWord uses only a small subset, e.g. Sql, Web, etc. are part of the library but aren't used.

That is correct, but I think you should start with simple. I guess Sql will deserve separate library anyway - in fact, I am not sure how to handle differente RDBMS engines, because many of them require RDBMS clients to be installed. So perhaps each engine will require separate .lib too (but hey, this really is not my problem to solve - we have solved by creating theide, which solves these kinds of problems

Quote:
P.S.2 what do I need to define besides PLATFORM_WIN32? DEBUG? UNICODE?

theide, UWord example, Setup/Verbose, see the commandline...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 11 Sep 2007 20:21:39 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 10 September 2007 20:21Followup: without completely understanding why and how INITBLOCK works, I replaced it in icpps with INITBLOCK_(BLK_###), where ### is the name of the file. It worked, and now I'm stuck on:

D:/Dev/0/1/libUpp/Core/Topt.h: In function `void Upp::AssertMoveable0(T*) [with T = Upp::wchar]':

D:/Dev/0/1/libUpp/Core/Topt.h:223:   instantiated from `void Upp::AssertMoveable(T*) [with T = Upp::wchar]'
D:/Dev/0/1/libUpp/Core/Vcont.h:89:   instantiated from `Upp::Vector<T>::~Vector() [with T = Upp::wchar]'
D:/Dev/0/1/libUpp/Core/Vcont.h:88:   instantiated from `Upp::Vector<T>::~Vector() [with T = Upp::Vector<Upp::wchar>]'
D:\Dev\0\1\libUpp\PdfDraw/PdfDraw.h:330:   instantiated from here
D:/Dev/0/1/libUpp/Core/Topt.h:214: error: invalid type argument of `unary *'

According to NTL requirements, wchar should be moveable. But that Assert fails.

template <class T>
inline void AssertMoveablePtr(T, T) {}

template <class T>
inline void AssertMoveable0(T *t) { AssertMoveablePtr(&**t, *t); }

T is wchar. That would be: &**(wchar*). **(wchar*) doesn't make sense. How does that check if the type is moveable?

You cannot make a C++ automatic check here. Moveable types have to explicitly tagget by Moveable by deriving from Moveable<T> or by NTL_MOVEABLE macro. That overrides AssertMoveable and this way is not used.

What you see is used to make all pointers moveable. (Little bit confusing, but the only possible way).

Problem with wchar migth be caused by missing defines. Please, do what I asked - run UWord compilation with verbose mode active to see all defines...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 11 Sep 2007 20:22:32 GMT
View Forum Message <> Reply to Message

Zardos wrote on Tue, 11 September 2007 03:25Unfortunately BLITZ has some problems on some packages in release mode. May be a another flag "No BLITZ in release mode" would help for some projects.

Well, it is a problem of MSC compiler and that flag is there...

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 12 Sep 2007 00:25:07 GMT
View Forum Message <> Reply to Message

OK, I see that I've sounded a bit too ignorant and stubborn. Sorry for that. To clarify things:

1) I'm not afraid of TheIDE, I just don't like it. I just got myself comfortable with Code::Blocks, so I don't want to move to a new IDE to use a new library. I used it a bit, and found it less comfortable than Code::Blocks.
2) I want to build U++ without TheIDE not only for the reason that I don't want to use TheIDE. I want to ensure the library code is not tied to the development environment, and is pretty much standalone. In the hypotetical case of TheIDE ceasing to function, I want to still be able to build whatever I write with U++.
3) I've built UWord with TheIDE, with verbose. It just seemed that it included too few packages, so I guessed others also might be needed in other applications. Currently, apart from packages that are used in UWord, I added GLCtrl and GridCtrl (I hope none of these can cause trouble). I also added more plugins

Command line in TheIDE:

```
c++ -c  -I"C:\upp\examples" -I"C:\upp\uppsrc" -I"C:\upp\mingw\include" -DflagGUI -DflagGCC
-DflagDEBUG -DflagDEBUG_FULL -DflagBLITZ -DflagW
 IN32 -DbmYEAR=2007 -DbmMONTH=9 -DbmDAY=12 -DbmHOUR=0 -DbmMINUTE=30
-DbmSECOND=10 -g2 -static   -fexceptions  -D_DEBUG -O0 -x c++ "C:\up
 p\uppsrc\CtrlLib\ChWin32.cpp" -o "C:/upp/out/CtrlLib/MINGW.Debug_full.Gui\ChWin32.o"
```

I don't know what -D means (doesn't appear in c++ --help), but if these are defines, I need to define: flagGUI, flagGCC, flagDEBUG, flagDEBUG_FULL, flagBLITZ, flagWIN32. Should've done this last time (got overhauled with info due to verbose and missed these defines).

wchar problem is now gone (phew). Got some wrong stuff in Locale.cpp (I'm using dev2b from sourceforge, it might be outdated by now - there was and extra */ and missing :: to call Win32 functions). t.h is really weird. I've removed namespace from it, got back to the compiler error. Writing UPP::LngEntry__ solved the problem, but I still don't understand how INITBLOCK_ is defined there (t.h is included outside UPP namespace). Well, actually I don't understand how INITBLOCK works at all, it looks like some kind of lambda for C++, does it actually execute the code (register) upon program loading, without calling anything?

I'm getting closer to building U++. It took 43 minutes to get to the first error, which is in RichText (last package to be built). What bothers me is that the build time is so long, and that I get the same warnings over and over again. It's as if Core.h gets re-included for every CPP, and warnings from all its includes repeat (impossible since Core.h has include guards). I might later try to set Core.h to get precompiled and see if it helps.

The RichText error was in Para.cpp. It included an NText.h (non-existant file), and also used Paragraph (non-existant class). On second look, TheIDE used RichText in UWord, but that particular file wasn't used. I wonder how TheIDE knew it wasn't necessary, especially since Para.h is used. Removed the file, compilation continued.

```
#ifndef USE_MSDOS_MEMMGR /* make sure user got configuration right */
  You forgot to define USE_MSDOS_MEMMGR in jconfig.h. /* deliberate syntax error */
#endif
```

That's a funny way to say something's wrong
Defined USE_MSDOS_MEMMGR in jconfig.h, got an error in jmemansi.c:

```
METHODDEF(void)
read_backing_store (j_common_ptr cinfo, backing_store_ptr info,
      void FAR * buffer_address,
      long file_offset, long byte_count)
{
  if (fseek(info->temp_file, file_offset, SEEK_SET))
    ERREXIT(cinfo, JERR_TFILE_SEEK);
  if (JFREAD(info->temp_file, buffer_address, byte_count)
      != (size_t) byte_count)
    ERREXIT(cinfo, JERR_TFILE_READ);
}
```

Structure has no temp_file member.

OK, I made some progress, time to sleep
Accumulating changes to the original sources package (half of it, actually):

In all icpps using INITBLOCK, replace it with INITBLOCK_(BLK_###) - unique ### for every INITBLOCK.
In t.h, replace LngEntry__ with UPP::LngEntry__
In Locale.cpp, add NAMESPACE_UPP and END_UPP_NAMESPACE, define LOG(x) (what is this?), remove extra */, add :: to some Win32 function calls.

And add main.cpp with :

```
#include "Core/Core.h"

#include "Core/Core_init.icpp"
#include "CtrlCore/CtrlCore.icpp"
#include "RichEdit/RichEdit.icpp"
#include "CtrlLib/CtrlLib.icpp"
#include "RichText/RichImage.icpp"
#include "PdfDraw/PdfReport.icpp"
#include "plugin/bmp/BmpReg.icpp"
#include "plugin/gif/gif.icpp"
#include "plugin/jpg/jpgreg.icpp"
#include "plugin/png/pngreg.icpp"
#include "plugin/tif/tifreg.icpp"

void LinkUppInit() {}
```

P.S. I've found: http://en.wikipedia.org/wiki/Single_Compilation_Unit
Pros: only 1 file to compile, probably faster static lib compiling, also icpp issue solution
Cons: INITBLOCK-s will have to be replaced with unique INITBLOCK_(X)-s, possibly other similar changes, every new CPP will have to be added to that file that is compiled (it could be auto-generated, though).

P.S.2 Thinking of it now, RichImage.icpp has 2 INITBLOCK-s, how can TheIDE compile these in one source file? It's redefined in Code::Blocks, unless I replace it with INITBLOCK_(X) with different X.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 12 Sep 2007 07:20:31 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 11 September 2007 20:25
I don't know what -D means (doesn't appear in c++ --help), but if these are defines, I need to define: flagGUI, flagGCC, flagDEBUG, flagDEBUG_FULL, flagBLITZ, flagWIN32. Should've done this last time (got overhauled with info due to verbose and missed these defines).

Yes, check...

Quote:
t.h is really weird. I've removed namespace from it, got back to the compiler error.

What is the error? (Important part is from where t.h was included).

Quote:
Writing UPP::LngEntry__ solved the problem, but I still don't understand how INITBLOCK_ is defined there (t.h is included outside UPP namespace). Well, actually I don't understand how INITBLOCK works at all, it looks like some kind of lambda for C++, does it actually execute the code (register) upon program loading, without calling anything?

It creates a special class and single global object of that class; uses constructor of the class to insert initialization code. It is pure macro hackery; the name of class and of object is created based on the line number.

Quote:
I'm getting closer to building U++. It took 43 minutes to get to the first error, which is in RichText (last package to be built). What bothers me is that the build time is so long, and that I get the same warnings over and over again. It's as if Core.h gets re-included for every CPP, and warnings from all its includes repeat (impossible since Core.h has include guards). I might later try to set

Core.h to get precompiled and see if it helps.

Yes, it gets included all the time. Anyway, long build times is the thing that theide solves too (with theide, BLITZ and HYDRA - I can completely rebuild UWord, including U++ library, in 24s seconds with mingw and in 14s with MSC...)

Quote:
The RichText error was in Para.cpp. It included an NText.h (non-existant file), and also used Paragraph (non-existant class).

Once again, follow the suggestions. The list of files that really are part of project is displayed in theide and also listed in .upp files inside package directories (also there is the dependency). (OTOH, thanks, this looks like abandoned file that was forgot in the folder).

Quote:
#ifndef USE_MSDOS_MEMMGR /* make sure user got configuration right */
  You forgot to define USE_MSDOS_MEMMGR in jconfig.h. /* deliberate syntax error */
#endif

That's a funny way to say something's wrong
Defined USE_MSDOS_MEMMGR in jconfig.h, got an error in jmemansi.c:

```
METHODDEF(void)
read_backing_store (j_common_ptr cinfo, backing_store_ptr info,
      void FAR * buffer_address,
      long file_offset, long byte_count)
{
  if (fseek(info->temp_file, file_offset, SEEK_SET))
    ERREXIT(cinfo, JERR_TFILE_SEEK);
  if (JFREAD(info->temp_file, buffer_address, byte_count)
      != (size_t) byte_count)
    ERREXIT(cinfo, JERR_TFILE_READ);
}
```

Structure has no temp_file member.

Well, these files are not from us, but this is jpeg library. Anyway, I think the problem might be the same as with RichText - redundant file compiled. Please check in theide or in .upp file whether this file is part of package. Same for Local.cpp.

(OTOH, this is really good, as we are now able to remove forgotten files .

Quote:
P.S. I've found: http://en.wikipedia.org/wiki/Single_Compilation_Unit
Pros: only 1 file to compile, probably faster static lib compiling, also icpp issue solution

Cons: INITBLOCK-s will have to be replaced with unique INITBLOCK_(X)-s, possibly other similar changes, every new CPP will have to be added to that file that is compiled (it could be auto-generated, though).

Ah, nice, somebody else noticed the basic principle of BLITZ too. Anyway, the difference is that BLITZ does all things automagically, solving the .icpp problem, creating the "SCU" and managing this so that frequently modified files are compiled separately.

Quote:
P.S.2 Thinking of it now, RichImage.icpp has 2 INITBLOCK-s, how can TheIDE compile these in one source file? It's redefined in Code::Blocks, unless I replace it with INITBLOCK_(X) with different X.

See above. They get different names within single file, which is OK as the global variable is static.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 12 Sep 2007 10:31:52 GMT
View Forum Message <> Reply to Message

I hoped moving away from MFC would result in getting away from all the macros, but I see they're alive and kicking...
How does it replace anything with line number? TheIDE feature? I've never seen this available in standard C++. If it's non-standard, why not replace all INITBLOCK with INITBLOCK_(X)-s? There are only about 30 of them.

I don't want to kill that 43-mins build effort, but IIRC since main.cpp compiled first:

#include "Core/Core.h"

#include "Core/Core_init.icpp"
...

t.h was included from Core_init.icpp:

#include "Core.h"

#define TFILE <Core/Core.t>
#include <Core/t.h>

The error with the unmodified file was that LngEntry was undefined (quite understandable, since Core_init.icpp doesn't have NAMESPACE_UPP). Less understandable is that the INITBLOCK actually ceases to function if I add NAMESPACE_UPP to t.h.

BLITZ is pretty impressive if it can understand which CPPs are unused (especially for CPPs sucha s Locale.cpp, that don't have H-s). But that's what I like about static libs - everything has to be compiled, so you can be sure you have no code that is simply unreferenced.

I don't yet understand why Core.h gets reincluded all the time (don't include guards prevent this?). But I think a rather simple batch could be used to create main.cpp with all CPPs/icpps included, and compile just it. BTW, does the whole static lib always get linked, or only the referenced part of it (if only part may get linked, then single compilation unit isn't a good solution)?

Here are unreferenced files from jpg:

ansi2knr.c
cjpeg.c
ckconfig.c
djpeg.c
example.c
jmemdos.c
jmemmac.c
jmemname.c
jmemnobs.c
jpegtran.c
rdjpgcom.c
wrjpgcom.c

Here are unreferenced files from tif:

fax3sm_winnt.c
mkg3states.c
mkspans.c
mkversion.c
tif_acorn.c
tif_apple.c
tif_atari.c
tif_msdos.c
tif_stream.cxx
tif_unix.c
tif_vms.c
tif_win3.c
tif_win32.c

I removed zim plugin since it didn't want to compile and I didn't see an example using it.

Now I have a 408MB static debug library
What's the largest debug exe ever built with U++? Does it even come close?

I am somewhat worried about stuff I removed, some of it might be used for other platforms (I don't have unix/linux/macosx).

Now I can try single compilation unit, release, precompiled headers.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 12 Sep 2007 12:25:10 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 12 September 2007 06:31I hoped moving away from MFC would result in getting away from all the macros, but I see they're alive and kicking...
How does it replace anything with line number? TheIDE feature? I've never seen this available in standard C++.

__LINE__ as a pseudomacro defined by C and C++ standard. It returns the current line number. With a little bit of macro hackery magic, you can use this to build identifier with line number in it.

Quote:
If it's non-standard, why not replace all INITBLOCK with INITBLOCK_(X)-s? There are only about 30 of them.

Not using macros is better than using them. Using macros for things that repeat and cannot be done without macros is better than repeating...

Quote:
t.h was included from Core_init.icpp:

#include "Core.h"

#define TFILE <Core/Core.t>
#include <Core/t.h>

The error with the unmodified file was that LngEntry was undefined (quite understandable, since Core_init.icpp doesn't have NAMESPACE_UPP). Less understandable is that the INITBLOCK actually ceases to function if I add NAMESPACE_UPP to t.h.

Ha! So the same issue again, forgotten file. Thank explains it.

Quote:
BLITZ is pretty impressive if it can understand which CPPs are unused (especially for CPPs sucha s Locale.cpp, that don't have H-s). But that's what I like about static libs - everything has to be compiled, so you can be sure you have no code that is simply unreferenced.

Can you please carefully read again what I wrote about what files are to be compiled? Please...

Quote:
I don't yet understand why Core.h gets reincluded all the time (don't include guards prevent this?).

Yes, they do. That is why it can be included so many times.

Quote:
But I think a rather simple batch could be used to create main.cpp with all CPPs/icpps included, and compile just it. BTW, does the whole static lib always get linked, or only the referenced part of it (if only part may get linked, then single compilation unit isn't a good solution)?

Yes, that is why release mode does not use BLITZ SCU so that .libs are build from many .obj and unreferences .objs can be removed. That is also the solely reason for .icpp - these initialization blocks are not referenced from the rest of the code, that is why if they are put into .lib, linker simply excludes them and no initialization happens.. That is why theide build process does not put them into .lib, but links them directly as .obj -> that is the only difference.

Quote:
Now I have a 408MB static debug library
What's the largest debug exe ever built with U++? Does it even come close?

Optimized release mode about 10MB (that is really big app, about one million lines).

Debug mode (with debug info) theide has about 20-30MB with GCC.

Quote:
I am somewhat worried about stuff I removed, some of it might be used for other platforms (I don't have unix/linux/macosx).

So? Your task was to build libs for Win32, so what is the problem?

Thank you for trying. The fact is, for me and other core U++ developers, lib form of U++ is quite redundant, but I understand that it can be quite atractive for many people. Any effort here is highly welcome...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 12 Sep 2007 13:22:56 GMT
View Forum Message <> Reply to Message

Ohh.. right, .upp. So these 20 or so files from jpg and tif are indeed forgotten. I also removed these from plugin/z:
example.c
maketree.c
minigzip.c

I think I'll continue with adding packages I've thrown out and checking for forgotten files. The only actual serious change was a few bugfixes in Locale.cpp.

I've managed to set Core/Core.h as precompiled header (not using single compilation unit). Build time went below 10 mins (very reasonable, wxWidgets is more than half an hour). Hopefully this way unnecessary stuff will be thrown out when using the static lib (11MB release, 410MB debug). I'll check that later.

I'm attaching modified Locale.cpp (original is in dev2b from sourceforge), and Debug and Release build logs (in case anyone wants to see the warnings).

Thanks for all the help. I want to shrink modifications to a single main.cpp file, so that anyone would be able to build U++ even if it is updated (not drastically, though).

File Attachments

1) Locale.cpp, downloaded 556 times
2) Logs.zip, downloaded 402 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 12 Sep 2007 15:34:02 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 12 September 2007 09:22Ohh.. right, .upp. So these 20 or so files from jpg and tif are indeed forgotten. I also removed these from plugin/z:
example.c
maketree.c
minigzip.c

As for those removals... Well, if it is 3rd party plugin, I do not quite like removing open-source stuff... It should be dealt by using only some files, not removing.

Quote:
I think I'll continue with adding packages I've thrown out and checking for forgotten files. The only actual serious change was a few bugfixes in Locale.cpp.

Which is meaningless anyway - Locale.cpp is another forgoten file.

Quote:
I've managed to set Core/Core.h as precompiled header (not using single compilation unit). Build time went below 10 mins (very reasonable, wxWidgets is more than half an hour).

Hm, we should add gcc precompiled headers support to builders too...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 12 Sep 2007 16:00:21 GMT
View Forum Message <> Reply to Message

Duh, if only I knew that all the trouble was due to forgotten files

I understand the concerns in plugins, but could you clean up these files in main packages?

As I see it now, all .upp-s should be parsed, and only cpps/icpps present there should be compiled.

I tried to build examples using the library I've built. Console ones work, GUI don't I run the exe and nothing happens (neither is visible in task manager). Exes are about 3MB in release, 17MB in debug. Full rebuild (debug+release) for a minimal GUI app (which doesn't work, but anyway) is 24 sec.

Subject: Re: Building & using U++ without TheIDE
Posted by Novo on Wed, 12 Sep 2007 16:20:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 12 September 2007 03:20
Yes, it gets included all the time. Anyway, long build times is the thing that theide solves too (with theide, BLITZ and HYDRA - I can completely rebuild UWord, including U++ library, in 24s seconds with mingw and in 14s with MSC...)

This is really cool feature of TheIDE. Is it possible to use the build system of TheIDE as a standalone application? I have two reasons to ask for this.
1) I cannot build TheIDE at work because we use somewhat outdated distributives. (Probably I haven't tried too hard.)
2) I'd like to use it with other IDE (VIM in my case).

Thanks.

Subject: Re: Building & using U++ without TheIDE
Posted by Novo on Wed, 12 Sep 2007 16:31:58 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 12 September 2007 12:00

As I see it now, all .upp-s should be parsed, and only cpps/icpps present there should be compiled.

You might want to take a look at
http://www.ultimatepp.org/forum/index.php?t=msg&&th= 2097&goto=8518#msg_8518

Everything is already parsed ...

It can be used as a reference.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 12 Sep 2007 16:47:27 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 12 September 2007 12:20luzr wrote on Wed, 12 September 2007 03:20
Yes, it gets included all the time. Anyway, long build times is the thing that theide solves too (with theide, BLITZ and HYDRA - I can completely rebuild UWord, including U++ library, in 24s seconds with mingw and in 14s with MSC...)

This is really cool feature of TheIDE. Is it possible to use the build system of TheIDE as a standalone application? I have two reasons to ask for this.
1) I cannot build TheIDE at work because we use somewhat outdated distributives. (Probably I haven't tried too hard.)
2) I'd like to use it with other IDE (VIM in my case).

Thanks.

You can use "umk" - as long as you can stand with packages (but in reality, system of packages is a perfect match for BLITZ, as it creates natural groups of files).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by Novo on Wed, 12 Sep 2007 18:04:56 GMT

luzr wrote on Wed, 12 September 2007 12:47Novo wrote on Wed, 12 September 2007 12:20luzr wrote on Wed, 12 September 2007 03:20
Yes, it gets included all the time. Anyway, long build times is the thing that theide solves too (with theide, BLITZ and HYDRA - I can completely rebuild UWord, including U++ library, in 24s seconds with mingw and in 14s with MSC...)

This is really cool feature of TheIDE. Is it possible to use the build system of TheIDE as a standalone application? I have two reasons to ask for this.
1) I cannot build TheIDE at work because we use somewhat outdated distributives. (Probably I haven't tried too hard.)
2) I'd like to use it with other IDE (VIM in my case).

Thanks.

You can use "umk" - as long as you can stand with packages (but in reality, system of packages is a perfect match for BLITZ, as it creates natural groups of files).

Mirek

umk will run theide.exe (is it called theide.exe on Unix?). That is not exactly what I want, because I cannot build TheIDE on Linux. I want a pure command line utility. UPP seems to be something more than just a GUI library .

Is there a dedicated build system API, which I can use to develop such a tool without reimplementing everything by myself? That would save a lot of time.

I'm sorry, I probably shouldn't ask such stupid questions.

Thanks.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 12 Sep 2007 20:01:09 GMT

Novo wrote on Wed, 12 September 2007 14:04
Is there a dedicated build system API, which I can use to develop such a tool without reimplementing everything by myself? That would save a lot of time.

No. It is now unfortunately bound with GUI...

Perhaps we should separate this...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 12 Sep 2007 23:19:08 GMT
View Forum Message <> Reply to Message

I've encountered 2 problems:

1) GUI EXEs simply refuse to run. When I try to debug them, they "stall" before reaching WinMain. Executed, they just disappear (that is, launch and terminate immediately). Console programs work fine. I'd be glad to hear any ideas about what could go wrong. Just in case, here's how I reference my static lib (I made UppGUI.h):

```
#ifndef UPP_H_INCLUDED
#define UPP_H_INCLUDED

#define flagGUI
#define flagGCC
#define flagBLITZ
#define flagWIN32

#if defined(DEBUG) || defined(_DEBUG)
#define flagDEBUG
#define flagDEBUG_FULL
#endif

#include <CtrlLib/CtrlLib.h>

using namespace Upp;

void LinkUppInit();
#define APP_MAIN INITBLOCK { LinkUppInit(); } GUI_APP_MAIN

#endif // UPP_H_INCLUDED
```

2) The icpps solution Mirek suggested (dummy function for each) is indeed better than mine (main.cpp for all). It became obvious when removing LinkUppInit halved the EXE size (to 1.5MB)
UWord is 2.3MB with GCC, so without LinkUppInit my static lib approach doesn't seem to incur any size penalty. Yet a question arises - how a regular build environment should decide which icpp should be linked and which shouldn't? For instance, RichText is included in CtrlLib.h, but might not be referenced in the program and thus thrown out by the linker. But if I call its link, it definitely won't be thrown out, even if unused.
A solution could be to let the users decide which links to call. But that's not nice.
A better solution (IMHO) is to convert these icpps to regular CPPs, and call the dummy function in

the most important function of the package. This might not be simple, but there is logic basis to this idea - functions take care of their requirements. E.g. for PdfDraw, it would make sense to put that call in GetDrawingToPdfFn function (in Draw/DrawUtil), since it's there that the initialization of the static variable is needed. There should be no problem to put several dummy calls in places necessary.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Thu, 13 Sep 2007 00:21:51 GMT
View Forum Message <> Reply to Message

OK, I found what caused the first problem. Turns out merely linking to libUppd.a was enough to turn a simple Win32 app that displays a message box into a bzip2 compressor I was surprised to get this in the command line from a GUI app:

D:\Dev\0\1\GUITest\bin\Debug>guitest
guitest: I won't write compressed data to a terminal.
guitest: For help, type: `guitest --help'.

D:\Dev\0\1\GUITest\bin\Debug>

I never knew WinMain can be overridden by a lib. The source didn't reference U++ in any way. I indeed didn't remove some forgotten sources (including bzip2 commandline). I tried and indeed, placing main and WinMain in same file, precedence goes to main, regardless of which is first.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 13 Sep 2007 03:45:45 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 12 September 2007 19:19
A solution could be to let the users decide which links to call. But that's not nice.
A better solution (IMHO) is to convert these icpps to regular CPPs, and call the dummy function in the most important function of the package. This might not be simple, but there is logic basis to this idea - functions take care of their requirements. E.g. for PdfDraw, it would make sense to put that call in GetDrawingToPdfFn function (in Draw/DrawUtil), since it's there that the initialization of the static variable is needed. There should be no problem to put several dummy calls in places necessary.

Heh, this sounds like reproducing the route that ultimately has led to .icpp.

Sure, first we have placed just this - calls to dummy functions at strategic places.

Anyway, this is not possible when the mere presence of package in project should mean something.

For example, .icpp in PdfDraw: If you have PdfDraw in project, Report viewer has "Export to PDF..." button. If not, no button.

The same is true for image plugins. You just add image format plugin package you need in your application.

You do not have to do anything else than adding the package. Thanks to .icpp...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Thu, 13 Sep 2007 13:43:00 GMT
View Forum Message <> Reply to Message

I'm starting to see the big picture. There are 2 options:

1) Compiler decides what goes in and what doesn't.
2) User decides what goes in and what doesn't.

I wanted to go for option 1 - that way, if user uses PDF, PDF goes in. I understand that U++ works according to option 2 - what's in, is used. That is actually easier to implement.

My current idea goes like this:

1) Take the source, create a project (in any compiler/IDE) for a static library.
2) Add all files in all packages (according to .upp to exclude forgotten/unused files, and maybe exclude TheIDE-only packages).
3) Remove .icpps from the project (they will not be compiled).
4) Build debug/release/whatever libs.
5) Make a new project (that will use U++).
6) Include main headers of packages used, and icpps of packages used (only once, into one of the project's cpps).
7) Build the project - linker will throw out unused packages, and initialize used packages since the icpps were included.

I'd like to automate this process. I'll try to make a program to scan folder structure, parse .upps, create static lib project (e.g. include only used files), and create package headers.

By package headers, I mean it will be a header that handles icpps and dependencies inside. Example (upppkg/CtrlLib.h):

```
#ifndef UPPPKG_CTRLLIB_H
#define UPPPKG_CTRLLIB_H
```

```
// Uses
#include <upppkg/CtrlCore.h>
#include <upppkg/RichText.h>

// Uses (platform)
#ifdef flagLINUX
#include <upppkg/PdfDraw.h>
#endif
#ifdef flagFREEBSD
#include <upppkg/PdfDraw.h>
#endif
#ifdef flagOSX11
#include <upppkg/PdfDraw.h>
#endif

// Header
#include <CtrlLib/CtrlLib.h>

// ICPP
#include <CtrlLib/CtrlLib.icpp>

#endif
```

Linking to libUpp and including this file should basically provide a similar environment as TheIDE provides, when you write #include <CtrlLib/CtrlLib.h> and add this package.

What do you think of this idea? It would take a second to generate headers from U++ source, another ~20 mins to build the library, and U++ is ready to use.

Questions:
1) Is the first header in file section of .upp always the most important one of that package? If not, how can the main header be determined?
2) I've found files with other extensions (not h/hpp/c/cpp/icpp) that maybe should be handled somehow - .dli, .iml, .in, .lay, .patch, .t, .upt, .usc, .vc. How should I take care of these?
3) Having a static lib + correct includes, there should be no problem using them in any project - exe/dll/lib, right? I've seen in another thread that there are problems with using U++ DLL in U++ EXE - wouldn't static linking each to U++ just work (OK, 1MB or so wasted, but still)?

P.S. I've tried to start making that parser, encountered 3 problems:
1) ToUnixName is implemented in Path.cpp but not defined in Path.h - can't use it.
2) I didn't find anything looking like this in the sources:

```
#ifdef PLATFORM_WIN32
const char cDirSep = '\\';
#else
const char cDirSep = '/';
```

#endif
Is there any reason for constantly using the chars '\\' and '/' and checking the OS?
3) I don't understand how unicode is implemented. There is String, AString, WString, but there is no TString, or whatever the name, like there is TCHAR that expands into char or wchar_t, depending on whether UNICODE/_UNICODE is defined. How do I define whether I'm in unicode or not? I mean, MessageBox will expand into MessageBoxA or MessageBoxW? And why path handling routines use char - can I handle unicode filenames with U++?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 13 Sep 2007 21:58:31 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 13 September 2007 09:43
1) Is the first header in file section of .upp always the most important one of that package? If not, how can the main header be determined?

Usually it is first, but it definitely does not need to be determined - it is determined by the name of #include from another package.

Quote:
2) I've found files with other extensions (not h/hpp/c/cpp/icpp) that maybe should be handled somehow - .dli, .iml, .in, .lay, .patch, .t, .upt, .usc, .vc. How should I take care of these?

Ignore them, they are #included (if necessary).

Quote:
3) Having a static lib + correct includes, there should be no problem using them in any project - exe/dll/lib, right? I've seen in another thread that there are problems with using U++ DLL in U++ EXE - wouldn't static linking each to U++ just work (OK, 1MB or so wasted, but still)?

Frankly, I am not sure what might go wrong in that case... I think in principle, this should really work.

Quote:
1) ToUnixName is implemented in Path.cpp but not defined in Path.h - can't use it.

WinPath, UnixPath, NativePath "rework" slashes to the required direction.

Quote:
3) I don't understand how unicode is implemented. There is String, AString, WString, but there is no TString, or whatever the name, like there is TCHAR that expands into char or wchar_t, depending on whether UNICODE/_UNICODE is defined. How do I define whether I'm in unicode

or not?

This is sort of irrelevant. There is no UNICODE mode. All the time you have 8-bit and 16-bit String/WString.

Recommended approach is to use UTF-8 encoding. In that case, both strings can contain unicode and there is simple conversion between them. (You can however use one of 15 WIN/ISO encodings as 8-bit default.

Quote:
I mean, MessageBox will expand into MessageBoxA or MessageBoxW?

Always into MassageBoxA. However, in U++ you rather use Prompt, which can work with UTF8.

Quote:
And why path handling routines use char - can I handle unicode filenames with U++?

Unfortunately, there is drawback caused by fact that we still have to support win98, so we cannot use W variants . In practive, this is really minor trouble, but nothing to be happy about it.

Anyway, when you are using only functions from U++, there is automatic conversion between U++ default encoding and 8-bit encoding of Windows.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Fri, 14 Sep 2007 00:21:15 GMT
View Forum Message <> Reply to Message

luzr wrote on Fri, 14 September 2007 00:58Quote:
3) I don't understand how unicode is implemented. There is String, AString, WString, but there is no TString, or whatever the name, like there is TCHAR that expands into char or wchar_t, depending on whether UNICODE/_UNICODE is defined. How do I define whether I'm in unicode or not?

This is sort of irrelevant. There is no UNICODE mode. All the time you have 8-bit and 16-bit String/WString.

Recommended approach is to use UTF-8 encoding. In that case, both strings can contain unicode and there is simple conversion between them. (You can however use one of 15 WIN/ISO encodings as 8-bit default.

Quote:

I mean, MessageBox will expand into MessageBoxA or MessageBoxW?

Always into MassageBoxA. However, in U++ you rather use Prompt, which can work with UTF8.

Quote:
And why path handling routines use char - can I handle unicode filenames with U++?

Unfortunately, there is drawback caused by fact that we still have to support win98, so we cannot use W variants . In practive, this is really minor trouble, but nothing to be happy about it.

Anyway, when you are using only functions from U++, there is automatic conversion between U++ default encoding and 8-bit encoding of Windows.

Mirek

That would be one pretty serious drawback IMHO. I'm using Russian and Hebrew, and I've used so many apps with bad unicode support that I really want to avoid making another one. Who uses Win98 nowadays (and there's unicode layer for these)

I've tried entering non-English chars in TheIDE source editor, it didn't let me. So I made a UTF-8 file containing a name of a file (with both Hebrew and Russian characters). U++ was able to read the name but unable to load the file. I guess this is the reason:
handle = CreateFile(ToSystemCharset(name),

Since this is CreateFileA, it can't open files with unicode filenames.

I tried adding #define UNICODE and #define _UNICODE. After commenting some stuff (cAlternateFileName didn't exist) it built the program but didn't work. CreateFileW can't accept UTF-8, it needs unicode.

Could this be fixed by some creative editing of Util.h? And what about window title, and controls? I mean, any chance to bring this to the TCHAR style, so the program can be compiled in ANSI, and in Unicode (supporting several languages simultaneously)? This should be a matter of some defines and a function to convert UTF8 to Win32 unicode. Or are there other parts of U++ that rely on a specific String/WString encoding?

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sat, 15 Sep 2007 16:57:50 GMT
View Forum Message <> Reply to Message

I made some progress.

Tried to implement full Unicode (TCHAR/String/WString handling), but eventually TheIDE/debugger went crazy (stopped several lines after breakpoint, for the same #ifdef flagUNICODE in one place in file went to #ifdef, another place to #else). So I dropped the idea for a while and returned to static lib building.

I made a mini-parser for *.upp files. Used it to generate list of used files, deleted them, ended up with unused/forgotten files. I'm attaching a zip of persumably forgotten files, maybe they should be removed.

I also found several referenced (used) yet non-existant packages:

TCore (Geom/Coords)
TCtrlLib (Geom/Ctrl)
TCtrlLib/Calc (Ole/Ctrl/Calc)
TDraw (Geom/Draw)
VectorDes (ide/VectorDes - it actually references itself, and by short path)

Additionally, directory separators aren't consistent - in some files '/' is used, in others '\'.

I also noticed that in plugin/png, most files are "pseudo-forgotten" - they don't appear in *.upp, yet they are included in pnglib.c. Does this happen in any other package?

Are there any plans to clean up the source? These inconsistencies are somewhat preventing me from completing the project generator.

File Attachments

1) Unused.zip, downloaded 447 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 18:11:36 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 13 September 2007 20:21
That would be one pretty serious drawback IMHO. I'm using Russian and Hebrew, and I've used so many apps with bad unicode support that I really want to avoid making another one. Who uses Win98 nowadays (and there's unicode layer for these)

Well, I agree. Believe me, I would be more than happy to be finally able to cancel Win98 support and make everything UNICODE.

But even year ago there were people asking for *TheIDE* to run 98.... I can undestand that people want apps to be Win98 compatible, but developing on it?...

Quote:

I've tried entering non-English chars in TheIDE source editor, it didn't let me.

Actually, this is different reason. See FAQ. (In short, theide keeps track of encoding of file used and if keystroke does not match encoding, it is not inserted-> switch file encoding to UTF-8.)

Quote:
Could this be fixed by some creative editing of Util.h?

Yes. In fact, the code is there for PocketPC version.

Quote:
And what about window title, and controls?

Window title already works both in Win98 and XP UNICODE (it is quite easy to provide both way).

All widgets are UNICODE capable, there really is no trouble.

Quote:
I mean, any chance to bring this to the TCHAR style, so the program can be compiled in ANSI, and in Unicode (supporting several languages simultaneously)?

No. Forget about TCHAR style. But it can be easily done just right:)

Which makes me thing, this can even be done right while still supporting Win98, it is just much more work (need to test platform and dynamically link A or W version). .dli would help.

Quote:
This should be a matter of some defines and a function to convert UTF8 to Win32 unicode.

Well, in fact, these functions are already there and they work. In fact, U++ has no problem to handle chinesse filenames (because chinesse windows are using multibyte 8bit encoding of characters, something similar to UTF-8, and it is trivial to convert this to UTF-8 and back...).

Quote:
Or are there other parts of U++ that rely on a specific String/WString encoding?

Well, once you have String<->WString conversion and default encoding (SetDefaultCharset), the rest of code is pretty encoding ambivalent. The only real problem is the one you have encountered - the outside world interface. This is basically filenames and fonts; fonts are working UNICODE in Win98 out of the box, so really the remaining problem is filenames.

But once again, current pragmatic solution is just a result of fact that 2 years ago we had

customers that required Win98 support...

Mirek

PS.: BTW, Hebrew... Is not it RTL? (Another not yet resolved problem, this time because we just do not know how RTL is supposed to work...).

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 595 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 20:00:19 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 15 September 2007 15:22Have 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.

Well, but that is not out-of-box solution...

Quote:
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:

Actually, there is already existing TSTR there, it is called FromSystemCharset and ToSystemCharset (you need two ways of conversion). See in util.cpp how it is defined for PocketPC - in that case, ToSystemCharset is the same thing as your TSTR...

Anyway, I would still rather used dynamic loading.

Quote:
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):

What is BOM?

Quote:
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 !,.?

Yes, that is exactly the trouble I am afraid of...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 20:06:20 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 15 September 2007 12:57
TCore (Geom/Coords)

TCtrlLib (Geom/Ctrl)
TCtrlLib/Calc (Ole/Ctrl/Calc)
TDraw (Geom/Draw)
VectorDes (ide/VectorDes - it actually references itself, and by short path)

Not required.

Quote:
Are there any plans to clean up the source? These inconsistencies are somewhat preventing me from completing the project generator.

This happens all the time...

Anyway, the sort of trouble is that there are "canonical packages", which are maintained in good health, but recent versions contained also some specific packages which frankly should not have been there.

Well, the trouble here is that modularity is a bit seductive; there are many more packages that we have developed (for our bussines), so the "public release" contains just selection of them. Sometimes things go bad.

Anyway, I consider "canonical" the packages that are used in TheIDE (contains most of GUI) + everything in plugin + all SQL related packages.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sat, 15 Sep 2007 20:16:03 GMT
View Forum Message <> Reply to Message

BOM: http://en.wikipedia.org/wiki/Byte_Order_Mark

I don't know what dynamic loading is, but your ToSystemCharset always kills non-currentcharset symbols due to CP_ACP. The way I intended TSTR is to convert anything (either String or WString) to _TCHAR. And if UNICODE is defined, _TCHAR is WCHAR, and thus all symbols are preserved. Another problem is that ToSystemCharset always returns String, which is castable to char*, and TSTR is castable to _TCHAR - which would always be what Win32 functions expect.

Is UTF-8 correctly converted in String <-> WString conversions? And are other plane symbols also supported (symbols that take 4 bytes in UTF-16)?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 20:48:08 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 15 September 2007 16:16BOM: http://en.wikipedia.org/wiki/Byte_Order_Mark

Ah, thanks, I did not knew that there is one for UTF-8...

So, if I understand you well, EF BB BF at the start of UTF-8 sequence should be ignored, right?

Anyway, this rather looks like file issue... Not sure it should be part of basic UTF-8 code?

Quote:
Another problem is that ToSystemCharset always returns String, which is castable to char*, and TSTR is castable to _TCHAR - which would always be what Win32 functions expect.

Ah, I see. You just did not undestood me. Look at PocketPC versio - that one returns WString instead of String. The result of these goes always directly to system calls (or, for FromSystemCharset, directly uses value returned by system call), therefore this is possible.

IMO, all really need to do to try playing with UNICODE/TCHAR is to change #ifdefs to use PocketPC (WinCE) version for UNICODE too.

Quote:
Is UTF-8 correctly converted in String <-> WString conversions?

I hope so, for basic plane and except the BOM. There is one extension though: Wrong UTF-8 sequences are interpreted byte by byte as characters 0xEExx (where xx is the byte).

The purpose is simple: This way, you can convert ANY input data to WString and back without loosing any information. This proved the absolute neccessity if you have the editor capable of handling multiple encodings in single file.

(That is why we named this UTF-8EE, as "Error Escape" or "placing to 0xEExx")

Quote:
And are other plane symbols also supported (symbols that take 4 bytes in UTF-16)?

No, unfortunately, other planes are not implemented yet.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sat, 15 Sep 2007 23:43:10 GMT
View Forum Message <> Reply to Message

Yes, UTF-8 can have BOM. Actually it has, and all programs I've used write/recognize it, with the unfortunate exception of GCC (it takes UTF-8 without BOM - such files aren't always shown correctly in text editors).

There is no need to modify existing UTF-8 handling since I doubt BOM is used in strings (it's essential for files). Yet my UTFBOM might be useful since LoadFile/SaveFile aren't encoding-aware, and files that are saved in UTF-16 format (that's what I commonly use for non-English text) would be loaded as ANSI/UTF-8. Or maybe just add autodecode to LoadFile and optional encoding params to SaveFile.

Multiple encodings in one file? Any examples? I don't think any text editor would recognize such a file.

I tried your suggestion about WinCE. Is PocketPC Unicode-only (that would be awesome if MS actually made such good decision)? First of all, cAlternateFileName, from Path.h, is never defined. As such, GetMSDOSName can't be defined either, and can't be used in FileSystemInfo::Find. That's likely a bug - I just commented everything out, or is there a way to implement GetMSDOSName?
Well, it didn't really work. The same "craziness" returned. I replaced:
#ifdef PLATFORM_WINCE
with:
#if defined(PLATFORM_WINCE) || defined(UNICODE)

and define UNICODE in main.cpp:

#define UNICODE

#include <Core/Core.h>

using namespace Upp;

...

Still, in the debugger it insisted to go into the #else. Rebuild all didn't help. My guess is that the solution could've worked (WString should cast into WCHAR*).

BTW, I've found some mistakes in UTFBOM and fixed that. Now I seem to be able to load/save UTF-8/UTF-16 LE/BE with/without BOM. I'm attaching the updated code (class + demo that I tried to use to read unicode-filenamed file). UTF-32 is kinda rare, though I might add it too for sake of completeness. However, that would require a String/WString that can work with embedded nulls - can they?

As for RTL, if I have time, I'll read Unicode specs on how LTR and RTL is mixed in same

paragraph. The issue is quite interesting, but I'm afraid a standards-conforming solution might end up being an unusual one since many programs tend to ignore the existance of RTL languages.

P.S. is there a portable way to get a key from console (only key, without Enter)? Like _getch()?

P.S.2 String (UTF-8) seems to be the native U++ representation of text. Is WString used / recommended to be used for anything
besides Unicode OS API calls?

File Attachments

1) UniTest.cpp, downloaded 658 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 23:55:36 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 15 September 2007 19:43
Multiple encodings in one file? Any examples? I don't think any text editor would recognize such a file.

We had to deal with such files in theide...

E.g. it is not that unlikely to have several string literals in single file, each encoded in different encoding, e.g. Win1252, Win1250 and UTF-8. In this case, to edit them, you have to switch the encoding of editor; you will of course see and be able to correctly edit only some of them at one encoding session, but in principle it will be possible. But it is possible only with Utf-8EE extension, otherwise you cannot load the file as Win1250 encoded strings will produce invalid UTF8 sequences.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 15 Sep 2007 23:58:31 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 15 September 2007 19:43
P.S. is there a portable way to get a key from console (only key, without Enter)? Like _getch()?

No.

Quote:

P.S.2 String (UTF-8) seems to be the native U++ representation of text. Is WString used / recommended to be used for anything
besides Unicode OS API calls?

The fundamental problem of UTF-8 encoded strings is that you cannot easily address individual characters. E.g. GUI editors use WString.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by cbpporter on Mon, 17 Sep 2007 05:53:14 GMT
View Forum Message <> Reply to Message

Quote:The fundamental problem of UTF-8 encoded strings is that you cannot easily address individual characters. E.g. GUI editors use WString.
Is WString UTF-16 or UTF-32. If it is UTF-16 you still can't address individual characters outside the BMP.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 08:10:23 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Mon, 17 September 2007 01:53Quote:The fundamental problem of UTF-8 encoded strings is that you cannot easily address individual characters. E.g. GUI editors use WString.
Is WString UTF-16 or UTF-32. If it is UTF-16 you still can't address individual characters outside the BMP.

UTF-16. Yes, you are right.

However, for any practical scenario, this is sufficient for now - and also the problem is that Win32 API is only 16-bit too (also, I have never seen any font with character outside BMP).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by cbpporter on Mon, 17 Sep 2007 08:53:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 17 September 2007 10:10
UTF-16. Yes, you are right.

However, for any practical scenario, this is sufficient for now - and also the problem is that Win32 API is only 16-bit too (also, I have never seen any font with character outside BMP).

Well plane 2 is used for some rare Kanji. In normal circumstances this is not a problem, but U++ can't be used for special programs (that process historic asian texts for example).

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 17 Sep 2007 09:07:41 GMT
View Forum Message <> Reply to Message

It's UTF-16. And the problem is solved by ignoring it - non-main plane characters aren't supported.

My attempts of creating a static lib have slowly become attempts to simulate BLITZ outside TheIDE. Overally, they ended up successfully. This isn't BLITZ yet, but build times are better than with precompiled header. EXE size is bigger, though (the way I implemented this, packages are either not included, or included as a whole - so even a basic GUI app would have all widgets and lots of other packages compiled and linked).

Animated Hello example (full rebuild):
Code::Blocks svn4421 / MinGW 3.4.5 (Debug) : 1:18 / 241 warnings / 11.7MB.
Code::Blocks svn4421 / MinGW 3.4.5 (Release): 2:59 / 252 warnings / 3.3MB.
TheIDE 708dev2b / MinGW 3.4.2 (Debug): 1:11 / 0 warnings / 13.2MB.
TheIDE 708dev2b / MinGW 3.4.2 (Optimal): 2:39 / 1 warning / 1.6MB.

Modifying the source to work as SCU on a per-package basis isn't so difficult. There were several "cosmetic" changes - adding underscores / commenting to prevent "redefined" errors (there are very few conflicting symbol names in U++, yet they exist). But there was one large change - instead of the included 1.1.4 / 1.2.2 zlib, I had to use zlib 1.2.3 and seriously modify it. For one, in SCU all files are compiled as C++, and zlib is K&R C - modifying all function declarations was necessary. Then, there were many conflicts between different c files, more in overall than int whole U++ without zlib. I couldn't resolve the conflicts in bundled zlib version, but I managed to in zlib 1.2.3 (underscores, include guards, etc.).

I could upload modified files if anyone is interested in trying it out / modifying main source. Would be nice if main U++ source could include at least the minor changes (without zlib), that way only replacing zlib would be necessary.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 09:22:27 GMT

View Forum Message <> Reply to Message

I think this is all good. There are two question:

After all that work, are you able to use U++ with e.g. CodeBlocks?
(I mean, have you compiled any of examples?)

Would you maintain this "port" for us? Or help somebody else to maintain it?

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 17 Sep 2007 11:50:35 GMT
View Forum Message <> Reply to Message

I successfully compiled Animated Hello example, and it ran fine (jumping colorchanging letters). I'll try other examples later, though I'm afraid they might not work due to the way TheIDE treats .lay and .iml files (other IDEs probably won't know what to do with these).

With a few changes to main source (which IMHO don't break anything for people who use TheIDE), this won't be a port but a possibility of usage of source. The only significant change would be update of zlib package, but it might not be that bad since current U++'s version is outdated.

Maintenance would probably consist of watching that .upp files are correct (no broken references), and that there are no conflicting names in the whole U++. The latter is currently easy but might become more difficult in the future. That's because U++ has lots of functions and enums, that are members of the namespace yet aren't members of any class. That seriously increases the chance that 2 unrelated cpp files will have a function with the same name - erroneous when compiling as SCU. While not necessary, it might be a good idea to create static classes for non-class functions - e.g. GetCurrentDirectory would be Path::GetCurrentDirectory().

Please tell me if it's fine to update zlib to a newer yet modified version. Plus, it should get tested (any simple U++ programs that could be used to test if zlib works correctly?). I'll try to make smallest-possible changes list to the main sources.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 13:46:40 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 17 September 2007 07:50I successfully compiled Animated Hello example, and it ran fine (jumping colorchanging letters). I'll try other examples later, though I'm afraid they might not work due to the way TheIDE treats .lay and .iml files (other IDEs probably won't know what to do with these).

Should not be the problem, these files are just "compiled" using preprocessor. TheIDE has editors for them, but during build process, they are ignored (they do not have .cpp nor .c extension after all

Quote:
Maintenance would probably consist of watching that .upp files are correct (no broken references), and that there are no conflicting names in the whole U++. The latter is currently easy but might become more difficult in the future. That's because U++ has lots of functions and enums, that are members of the namespace yet aren't members of any class. That seriously increases the chance that 2 unrelated cpp files will have a function with the same name - erroneous when compiling as SCU.

a) This is unlikely. Also, just the same name is not enough, only same signature is the problem.
b) I do not quite understand this SCU issue. I thought that task is to make possible to use U++ with CodeBlocks and VisualC++. I believe that users rather expect .lib files?

Quote:
Please tell me if it's fine to update zlib to a newer yet modified version. Plus, it should get tested (any simple U++ programs that could be used to test if zlib works correctly?). I'll try to make smallest-possible changes list to the main sources.

Yes, it is OK. Alternatively, please put updated package here so that we can pick it up for the next dev release.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 17 Sep 2007 16:33:52 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 17 September 2007 15:46

a) This is unlikely. Also, just the same name is not enough, only same signature is the problem.
b) I do not quite understand this SCU issue. I thought that task is to make possible to use U++ with CodeBlocks and VisualC++. I believe that users rather expect .lib files?

Mirek

b) I started with the goal of .lib, yet now I find it rather unattractive due to the 410MB debug lib. With a lib there are 2 possibilities - redistribute it, or redistribute a project to easily build it. 410MB makes the former impossible, and the latter would require maintaining projects for different compilers/IDEs. + even with precompiled headers build time is about 10 mins vs ~1 min in TheIDE.

So what I did now, instead, is use SCU approach to drastically reduce compilation time (it's not much worse than BLITZ's now), partially at the cost of EXE size non-modularity (simple GUI and complex GUI apps will have the same big EXE since the whole CtrlLib package is linked).

I have an interface header for each package, implementing SCU, so when I do: #include <Upp/CtrlLib.h> it's like adding CtrlLib package in TheIDE. These headers are auto-generated from U++ source (using .upp files). So basically, user can work in Code::Blocks, without a static lib (saving space and better debugging), yet with similar fast compiles. Environment becomes quite similar to TheIDE, though there are drawbacks - no embedded help / .lay and .iml editors, and larger EXE size.

I believe these drawbacks aren't too big, and I don't rule out the possibility of building (based on PCH, not SCU) and redistributing a release-only lib to reduce final EXE size.

a) Unlikely? Yes. Yet possible and it happens. IsLeapYear is once a function, another time a macro. BINS is defined in heap and in draw palette, with different values. INITBLOCK/EXITBLOCK also cause conflicts since some happen to appear on the same line number. z.cpp (zlib) is for some reason in Core, and it redefines ASCII_FLAG, HEAD_CRC etc. already defined in plugin/z. RichText/Para.h has Code enum, yet Code is #defined in plugin/z/lib/deflate.h. These are the minor changes in main U++ source I was talking about. That, zlib, and a few casts in png to make it C++ compatible.

P.S. tried to compile on MSVC8. This piece from deflate.h:

```
#define Freq fc.freq
#define Code fc.code
#define Dad  dl.dad
#define Len  dl.len
```

Doesn't let me compile since Code and Len are names of parameters of functions in winnt.h. => RichText/Para.h is fine, zlib should be further modified.

I think I'll just prefix everything troublesome in zlib with "zlib_". Should work. Any demo projects to test zlib? I want to make sure that the library still works after my modifications.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 20:20:33 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 17 September 2007 12:33
I think I'll just prefix everything troublesome in zlib with "zlib_". Should work. Any demo projects to test zlib? I want to make sure that the library still works after my modifications.

Basically anything with images. .iml files are zlib compressed.

Then anything storing/loading .png files.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 20:28:09 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 17 September 2007 12:33
redistributing a release-only lib to reduce final EXE size.

I would rather follow this approach.

Or perhaps even better, you can make debug version - with runtime checks but without the debug info.

BTW, as I think you know, you can run TheIDE in commandline mode too, so automated generation tool does not rule out use TheIDE as build tool. I am even willing to add functions to generate .libs directly - but I do not really know how to group all the stuff to libs.

Mirek

P.S.: Looking at all troubles you have with this reminds me why we have created theide For the first 2 years, U++, named SQL++ back then, was developed with Visual Studio 6.0...

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 17 Sep 2007 21:47:13 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 17 September 2007 22:28sergei wrote on Mon, 17 September 2007 12:33
redistributing a release-only lib to reduce final EXE size.

I would rather follow this approach.

Or perhaps even better, you can make debug version - with runtime checks but without the debug info.

BTW, as I think you know, you can run TheIDE in commandline mode too, so automated generation tool does not rule out use TheIDE as build tool. I am even willing to add functions to

generate .libs directly - but I do not really know how to group all the stuff to libs.

Mirek

P.S.: Looking at all troubles you have with this reminds me why we have created theide For the first 2 years, U++, named SQL++ back then, was developed with Visual Studio 6.0...

Well, I want to keep SCU approach for debug. Debug EXEs are about 13MB - not too much. The heavy point in favor of this system is being able to step in U++ source during debug. + You could also easily modify U++ source while developing/debugging. So IMHO debug lib isn't necessary.

Release lib is also not necessary. It would be just an improvement for EXE size (I could create a lib with SCU approach, but it wouldn't make any difference - including a package would link all of it).

The problem with precompiled header vs SCU is the need to create project files. With SCU you can redistribute the source, user just includes necessary packages. With precompiled header you have to setup project files with all cpps + precompiled header to build a lib. Which isn't trivial, since not all cpps actually should be compiled. So, SCU gets a point for maintainability. Precompiled header is worth it only in release - smaller EXE. And only for programs that don't utilize most of the code in packages they use.

More interesting solution could be splitting packages, like instead of one huge CtrlLib, several with different kinds of widgets. But that's probably just too much work...

P.S. devpacks are released every 2-3 weeks, right? Can I get the current source tree somewhere?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 17 Sep 2007 22:08:41 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 17 September 2007 17:47
P.S. devpacks are released every 2-3 weeks, right?

In theory, they should be released each week

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 17 Sep 2007 22:32:45 GMT

sergei wrote on Mon, 17 September 2007 11:07

Animated Hello example (full rebuild):
Code::Blocks svn4421 / MinGW 3.4.5 (Debug) : 1:18 / 241 warnings / 11.7MB.
Code::Blocks svn4421 / MinGW 3.4.5 (Release): 2:59 / 252 warnings / 3.3MB.
TheIDE 708dev2b / MinGW 3.4.2 (Debug): 1:11 / 0 warnings / 13.2MB.
TheIDE 708dev2b / MinGW 3.4.2 (Optimal): 2:39 / 1 warning / 1.6MB.

Update:
I've somewhat refixed zlib, so now no compile errors on MSVC rebuild.
Same Animated Hello full rebuild:
MSVC 8.0 (Debug) : 0:23 / 73 warnings / 3.9MB
MSVC 8.0 (Release) : MS Linker crashed. First time this happened to me
--> fatal error LNK1000: Internal error during IMAGE::BuildImage

Google found this: http://connect.microsoft.com/VisualStudio/feedback/ViewFeedback.aspx?FeedbackID=100217
Should install SP1...

Subject: Re: Building & using U++ without TheIDE
Posted by tvanriper on Tue, 18 Sep 2007 00:35:58 GMT

Quote:
sergei wrote on Sat, 15 September 2007 15:22

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.

luzr wrote on Sat, 15 September 2007 16:00

Well, but that is not out-of-box solution...

I have had quite a bit of experience with using UNICOWS, and can at least comment on its use, and perhaps offer an alternative idea.

It is, and it isn't an out-of-box solution. To use it, you have to link to a .lib file provided by Microsoft in a special way (it has to be the first library linked, then subsequent libraries can link... otherwise the system won't work). And you have to distribute a 'unicows.dll' (if I remember the name properly) with the application on Win9x-derived OSes (but you don't have to distribute it on non-Win9x-derived OSes). Otherwise, the application will call the Unicode versions of the Win32

API calls, and Mysteriously Bad Things Will Happen.

So, yeah, you could distribute applications with it 'out-of-the-box' in many cases, but for Win9x, you'd have to drop in that DLL. And... well I don't know Microsoft's position on the distribution of this library (or the .lib file, for that matter). I wouldn't think they'd have a problem, but who really knows? At the very least, it's kind of painful to set up.

All of this said, they aren't really doing anything that's terribly mysterious. I haven't looked over all the Win32 API calls you're making, but chances are you could probably do something like unicows.dll yourself by reproducing the wide versions of the function calls yourself, having them call the ANSI versions on 9x systems (doing the Unicode-to-ANSI conversions between the wide/narrow API calls), or pass-through to the Unicode versions on other systems.

Depending on the volume of code involved, this might not be so bad. Or, it could be a nightmare. I don't know... I haven't looked too deeply into Ultimate++'s inner-workings to see how much of the Win32 API you're using.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 18 Sep 2007 01:04:03 GMT
View Forum Message <> Reply to Message

Update: I've found a flaw in my SCU method. Since main.cpp will contain all U++ code due to the includes, any change in main.cpp will require full rebuild - not nice. I've reworked the structure - now there are 2 additional files - UppBase.h and UppBase.cpp. To use U++, user should copy & add both to his project. UppBase.h may be included in all source files that use U++. Also, UppBase.h contains #includes of packages that should be used. UppBase.cpp is a helper source, that will be the SCU. Unless it's changed (and it will change only if UppBase.h changes - which probably happens only when packages set changes), full rebuild of U++ won't be necessary.

Surprisingly, this method allowed me to detect more bugs. That's because all U++ headers are included before the first U++ cpp. Example: IsClipboardFormatAvailable is used in Draw/MetaFile, and it is defined in CtrlCore/CtrlCore.h. Seems fine, usually is fine, but actually incorrect. Draw package doesn't declare in uses (Draw.upp) that it uses CtrlCore, yet it uses its function. Not sure what's the best solution (easiest is to add CtrlCore to Draw's uses).

P.S. .upp inconsistencies that still haven't been corrected (as of 709dev1):
ide/VectorDes uses VectorDes (itself?, and wrong folder)
Ole/Ctrl/Calc and some Geom packages use T??? packages (such don't exist)
coff/uar/uld/uar.upp - probably should delete whole folder...

Subject: Re: Building & using U++ without TheIDE
Posted by unodgs on Tue, 18 Sep 2007 06:30:00 GMT
View Forum Message <> Reply to Message

Quote:In theory, they should be released each week
But please forgive maintainers to do it less often sometimes. Life is life..

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 18 Sep 2007 16:05:05 GMT
View Forum Message <> Reply to Message

OK, the issue with clipboard had nothing to do with dependencies, it was an API call mistaken for U++ function by the compiler. :: solved it.

I've updates to SP1, yet the error didn't go away. At least now it's preceded with C1001 - internal error of compiler. Again, release-only problem, debug works. Compiler's error is on line 54 of RichText/txtop.cpp. Which is: if(update). Yeah, that's one complicated line
Any help would be welcome.

That's what I got:
d:\programming\upp\richtext\txtop.cpp(54) : fatal error C1001: An internal error has occurred in the compiler.
(compiler file 'F:\SP\vctools\compiler\utc\src\P2\main.c[0x10BF5F00:0x00000 02C]', line 182)
To work around this problem, try simplifying or changing the program near the locations listed above.

For the record, F: is a CD drive

Currently I'm rather pleased with the results (MS bug isn't my fault...). I'll test some more example projects (see how .iml and .lay work, test zlib), and retry unicode filenames. Then I'll upload everything.

For now, I'm attaching Animated Hello project. It won't compile without packages headers I've generated, but I'd like to know what you think about the way a typical U++ project on CodeBlocks/MSVC would look (I mean the sources).

P.S. I decided to try out ld and ar replacements.

Animated Hello full rebuild (debug + release):
MinGW 3.4.5 (original) : 4:27 / 562 warnings / 11.9MB + 3.3MB
MinGW 3.4.5 (U++ ld and ar) : 4:14 / 562 warnings / 11.1MB + 3.2MB

-> 5% reduction in build time, 7% reduction in debug EXE size, 1% reduction in release EXE size. Are these the common results? And is it safe to use these programs (I'm somewhat uncomfortable with patching the compiler).

File Attachments

1) UppTest.zip, downloaded 413 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 18 Sep 2007 17:04:51 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 18 September 2007 12:05OK, the issue with clipboard had nothing to do with dependencies, it was an API call mistaken for U++ function by the compiler. :: solved it.

I've updates to SP1, yet the error didn't go away. At least now it's preceded with C1001 - internal error of compiler. Again, release-only problem, debug works. Compiler's error is on line 54 of RichText/txtop.cpp. Which is: if(update). Yeah, that's one complicated line
Any help would be welcome.

That's what I got:
d:\programming\upp\richtext\txtop.cpp(54) : fatal error C1001: An internal error has occurred in the compiler.
(compiler file 'F:\SP\vctools\compiler\utc\src\P2\main.c[0x10BF5F00:0x00000 02C]', line 182)
To work around this problem, try simplifying or changing the program near the locations listed above.

For the record, F: is a CD drive

Hehe, that one is well known here:) The same result with MSC7.1 and MSC8.

For regular work with theide, it is non-issue as for release mode BLITZ is not recommended anyway as it produces longer .exes.

Quote:
P.S. I decided to try out ld and ar replacements.

Animated Hello full rebuild (debug + release):
MinGW 3.4.5 (original) : 4:27 / 562 warnings / 11.9MB + 3.3MB
MinGW 3.4.5 (U++ ld and ar) : 4:14 / 562 warnings / 11.1MB + 3.2MB

-> 5% reduction in build time, 7% reduction in debug EXE size, 1% reduction in release EXE size. Are these the common results?

Well, perhaps for rebuilding everything, reduction in build time is not that significant. But if you are rebuilding just single file while developing, which usually takes about 2-3s, that 13s difference is quite welcome.

Quote:
And is it safe to use these programs (I'm somewhat uncomfortable with patching the compiler).

These really are not patches. These replacements are rewrites from the scratch, using U++ Core (NTL). That is where the speed comes from

As for bugs, who knows... But originals are not completely bug-free either.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 18 Sep 2007 22:43:26 GMT
View Forum Message <> Reply to Message

Update:

I'm done implementing unicode. I don't exactly like the way I did it, but it works. I prefer some global strings manager like the TSTR I suggested, but since Mirek said he doesn't want TCHARs I've updated all API calls manually. I added PLATFORM_UNICODE define, and replaced many #ifdef PLATFORM_WINCE with it. Conversion mostly implemented through ToSystemCharset and FromSystemCharset.

In the process I've also found and eliminated a "security vulnerability" (that's what memory bugs are called nowadays) in Log.h:

```
sprintf(h, "* %s %02d.%02d.%04d %02d:%02d:%02d, user: %s\n",
  (const char*)FromSystemCharset(exe),
  t.day, t.month, t.year, t.hour, t.minute, t.second, (const char*)FromSystemCharset(user));
```

I added the (const char*), otherwise I got segmentation fault in debug. Probably without the explicit cast sprintf thought that String is a char array.

I received another segmentation fault earlier (but maybe it was this one...), and that one was solved by replacing all To/FromSysChrSet with To/FromSystemCharset. Not a big deal IMHO, they were all used in Win32-specific code anyway...

I've tested unicode filenames - it did open a multilingually-named file and read its content successfully. I've also tested registry - successfully wrote that filename to a REG_SZ key, it's fine. Didn't try to create unicode-named keys (don't wanna kill my windows).

Next (final) step - zlib/lay/iml/images testing. If that goes well, I'll upload everything.

P.S. I'm working from Code::Blocks when editing U++ source, rebuild for console project is half a minute. So I don't want a static lib for debug Though GDB / Code completion don't work that well - GDB reports most stuff in U++ source as incomplete type (so I have to Cout whatever I want to see), and Code completion usually can't find definition of things (but it does know function

prototypes).

P.S.2 since you know about the MSVC bug, did anyone report to MS? There might be a chance that they fix it...

Edit: I think UTFBOM class I posted above, or something else implementing that functionality, should be added to some place in U++. That way unicode support will be complete - unicode filenames + unicode text.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 07:34:05 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 18 September 2007 18:43Update:

I'm done implementing unicode. I don't exactly like the way I did it, but it works. I prefer some global strings manager like the TSTR I suggested, but since Mirek said he doesn't want TCHARs I've updated all API calls manually. I added PLATFORM_UNICODE define, and replaced many #ifdef PLATFORM_WINCE with it. Conversion mostly implemented through ToSystemCharset and FromSystemCharset.

In the process I've also found and eliminated a "security vulnerability" (that's what memory bugs are called nowadays) in Log.h:

```
sprintf(h, "* %s %02d.%02d.%04d %02d:%02d:%02d, user: %s\n",
  (const char*)FromSystemCharset(exe),
  t.day, t.month, t.year, t.hour, t.minute, t.second, (const char*)FromSystemCharset(user));
```

I added the (const char*), otherwise I got segmentation fault in debug. Probably without the explicit cast sprintf thought that String is a char array.

I received another segmentation fault earlier (but maybe it was this one...), and that one was solved by replacing all To/FromSysChrSet with To/FromSystemCharset. Not a big deal IMHO, they were all used in Win32-specific code anyway...

Uh oh, that is no good. There definitely should be FromSysCharset, which returns a pointer to static char * array.

The reason is that Log is supposed to work independently from the rest of U++, so that it can be used in situation when the rest failed or is not available. By using vanilla FromSystemCharset, you make it dependent on String and in turn on memory allocator. Therefor it will not work when heap crashes or cannot be used to debug memory allocator...

Quote:
I've tested unicode filenames - it did open a multilingually-named file and read its content successfully. I've also tested registry - successfully wrote that filename to a REG_SZ key, it's fine. Didn't try to create unicode-named keys (don't wanna kill my windows).

Sort of redundant work. The final solution will use dynamic .dll loading to choose between W and A variants.

Quote:
P.S.2 since you know about the MSVC bug, did anyone report to MS? There might be a chance that they fix it...

No.

Quote:
Edit: I think UTFBOM class I posted above, or something else implementing that functionality, should be added to some place in U++. That way unicode support will be complete - unicode filenames + unicode text.

Well, the question is the definition. Thinking about it, I just fail to see what is the exact description. If detection of utf-8 files is the purpose, that can be easily done withou specific code. Also, what is going to happen if there is no UTFBOM?

Should it be available as

bool SkipUTFBOM(Stream& in);
const char *SkipUTFBOM(const char *s);

?

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 08:12:23 GMT
View Forum Message <> Reply to Message

I see now... I knew there should've been a good reason for a second set of charset functions. OK, I'll reverse the changes and see if that works.

About registry:

bool SetWinRegString(const String& string, const char *value, const char *path, HKEY base_key)
{

```
 HKEY key = 0;
 if(RegCreateKeyEx(base_key, ToSystemCharset(path), 0, NULL,
REG_OPTION_NON_VOLATILE,
  KEY_ALL_ACCESS, NULL, &key, NULL) != ERROR_SUCCESS)
  return false;
#ifdef PLATFORM_UNICODE
 WString wstring = string.ToWString(); wstring.Cat(0, 1);
 bool ok = (RegSetValueEx(key, ToSystemCharset(value), 0, REG_SZ, (const byte*)(const
wchar*)wstring, (wstring.GetLength() + 1)*2) == ERROR_SUCCESS);
#else
 bool ok = (RegSetValueEx(key, value, 0, REG_SZ, (const byte*)(const char*)string,
string.GetLength() + 1) == ERROR_SUCCESS);
#endif
 RegCloseKey(key);
 return ok;
}
```

The #else part is what was prevoiusly the code (I added the casts, though). Linking that to W version wouldn't be possible - defining UNICODE without #ifdef would cause an error of cast of char* to WCHAR*.

I found that I didn't modify everything, since console apps don't include much Then I found that already exists a macro L_(). Thus using TCHAR instead of char/wchar, + L_() and To/FromSystemCharset might indeed remove these #ifdefs. But that would be later, first to ensure everything works.

I'm not sure how you want to use dynamic dll loading. Change all #ifdefs into if/elses, and explicitely call W and A versions, to enable runtime switching between ANSI/Unicode?

UTFBOM: Skip BOM of UTF-8 / UTF-16 LE / UTF-16 BE files (not only UTF-8), and read ASCII/UTF-8 (if there's no BOM, it's considered ASCII) into String, UTF-16 LE/BE into WString. Convert UTF-8 String into UTF-8 / UTF-16 LE/BE with/without BOM. I guess it should be:

```
int FromFileCharset(const String& s, String* os, WString* ows);
String ToFileCharset(const String& s, int bytes, bool BOM = true, bool LE = true);
String ToFileCharset(const WString& s, int bytes, bool BOM = true, bool LE = true);
```

(maybe should add ASCII -> UTF-8 conversion if there's no BOM, since chars > 127 could cause invalid UTF-8, being just system-charset chars)

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 08:45:45 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 19 September 2007 04:12
I'm not sure how you want to use dynamic dll loading. Change all #ifdefs into if/elses, and explicitely call W and A versions, to enable runtime switching between ANSI/Unicode?

Yes. With .dli, it is not as much trouble as it seems. In fact, you forced me to work on it right now

Quote:
UTFBOM: Skip BOM of UTF-8 / UTF-16 LE / UTF-16 BE files (not only UTF-8), and read ASCII/UTF-8 (if there's no BOM, it's considered ASCII) into String, UTF-16 LE/BE into WString. Convert UTF-8 String into UTF-8 / UTF-16 LE/BE with/without BOM. I guess it should be:

int FromFileCharset(const String& s, String* os, WString* ows);
String ToFileCharset(const String& s, int bytes, bool BOM = true, bool LE = true);
String ToFileCharset(const WString& s, int bytes, bool BOM = true, bool LE = true);

(maybe should add ASCII -> UTF-8 conversion if there's no BOM, since chars > 127 could cause invalid UTF-8, being just system-charset chars)

I see. In fact you suggest something like LoadUnicodeAny, which detects the kind of file and loads UTF-8 or UTF-16LE or UTF-16BE.
Returning WString (it is easy to convert it to String).

Hm, perhaps there should be two variants after all to avoid unnecessarry UTF-8 -> UTF-16 -> UTF-8 conversion...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 09:18:12 GMT
View Forum Message <> Reply to Message

Why need dli? You already have all functions from #include <windows.h>. The trouble would only be explicitly calling A and W version. Thinking of it, it sounds nice - make PLATFORM_UNICODE a global boolean, initialized to true, unless OS is Win9x. But I'd prefer to finish the way I started to see everything work.

UTF-8 -> UTF-16 -> UTF-8 won't happen. FromFileCharset returns String if it's ASCII/UTF-8 and WString if it's UTF-16. It returns amount of bytes. 0 -> ASCII / String, 1 -> UTF-8 / String, 2-> UTF-16 / WString (4 -> UTF-32 / WString, but not implemented). What could happen is UTF-16 -> WString -> String, but UTF16 -> WString isn't expensive.

I wanted to compile UWord (now in ANSI, GUI Unicode isn't complete yet) to see if zlib work (UWord.iml), and found an interesting problem in PdfDraw:

ScreenDraw sd;

That causes a warning of statement is a reference not a function call. + error about sd definition. In Draw/DrawWin32, ScreenDraw is a class, but also:

```
ScreenDraw& ScreenDraw()
{
 return Single<ScreenInfoClass>();
}
```

That's a singleton? Whatever it is, it doesn't work - ScreenDraw sd; is recognized as a function name, not class type. Any suggestions how to fix?

P.S. Why does U++ use so many global functions? I prefer .Net-style - tree-like organization using namespaces/classes. After all, gathering functions into static classes should be realtively easy, and at the cost of some extra typing you (potentially) resolve naming conflicts, and make stuff easier to find. E.g. I may not know that there's a function named GetWinRegString hidden somewhere in Core/Win32Com. But if there was a class Registry, it would be more likely that I'd find it by typing Registry::. Plus that would be an OOP approach

---

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 09:33:49 GMT
View Forum Message <> Reply to Message

... It was way too tempting to see if UWord works, so I removed PDF and built without. 5:36, full debug+release rebuild (without PDF). 12.1MB debug, 4.2MB release. It works, and most importantly, I see the icons on toolbar, these look just fine. It means new zlib works, right?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 09:39:12 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 19 September 2007 05:18Why need dli? You already have all functions from #include <windows.h>.

Yes, but it would not start on Win98 ("missing dll call").

Quote:
UTF-8 -> UTF-16 -> UTF-8 won't happen. FromFileCharset returns String if it's ASCII/UTF-8 and WString if it's UTF-16.

How? Sure, you can make it return something more complex, but I would prefer two functions: one returning String and other WString.

Quote:
I wanted to compile UWord (now in ANSI, GUI Unicode isn't complete yet) to see if zlib work (UWord.iml), and found an interesting problem in PdfDraw:

ScreenDraw sd;

That causes a warning of statement is a reference not a function call. + error about sd definition. In Draw/DrawWin32, ScreenDraw is a class, but also:

```
ScreenDraw& ScreenDraw()
{
 return Single<ScreenInfoClass>();
}
```

That's a singleton? Whatever it is, it doesn't work - ScreenDraw sd; is recognized as a function name, not class type. Any suggestions how to fix?

Now this is really interesting. It is obvious bug (minor, this is just something forgotten), just tell why in theide it compiles without a error, in both MSC and all versions of mingw?

Maybe different way of SCU? (Your SCU is bigger than mine?

Quote:
P.S. Why does U++ use so many global functions? I prefer .Net-style - tree-like organization using namespaces/classes. After all, gathering functions into static classes should be realtively easy, and at the cost of some extra typing you (potentially) resolve naming conflicts, and make stuff easier to find.

Actually, if you look a bit more carefully, then either these are really very globally used names (e.g. AsString) - there is a limited number of such cases, or they are carefully bound to parameters so that C++ overloading resolves the conflicts.

Believe or not, function name conflicts are not an issue. (Moreover, everything is still in U++ namespace).

Quote:
Plus that would be an OOP approach

I like multi-paradigm approaches more

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 09:40:10 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 19 September 2007 05:18

```
ScreenDraw& ScreenDraw()
{
 return Single<ScreenInfoClass>();
}
```

Forgot to say: Just delete it..

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 10:22:10 GMT
View Forum Message <> Reply to Message

Regarding UTF - I understand that. I made it take 2 pointers - to String and WString. That was for optimal conversion. I'll split that, and add a function that returns encoding bytes according to BOM - so if user wants to go for the faster conversion, he would be able to choose between String and WString.

Regarding SCU - no idea... My SCU first includes all headers, then all sources (of packages used).

Tested PDF export, now works. Should complete unicode, and test layouts and more images.

P.S. tried Hebrew support in UWord. Pretty bad While it does display correctly what I type, everything else doesn't work. E.g. no switch to RTL button, home/end keys work as in LTR, left/right work for some reason... Worse yet - it doesn't position the cursor where I see it:
(TEX|T)
If I type a letter I get:
(TELX|T)
Same goes for copy/paste of selected text, it doesn't select the correct text (selects same amount

from other side).

Even worse in textboxes (like in save file). Not only inserting is wrong, but there's also a sliding problem - when you select text, it changes while you're selecting it (slides into selection from other side). I've seen that in a few apps before, makes editing text impossible.

Could at least partial support be improved? I mean, ignore keys and RTL layout. At least should be WYSIWYG in insert/delete/select/copy. BTW, export to PDF reversed Hebrew text. Expectable, but not nice.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 10:39:12 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 19 September 2007 06:22
E.g. no switch to RTL button, home/end keys work as in LTR, left/right work for some reason...
Worse yet - it doesn't position the cursor where I see it:
(TEX|T)
If I type a letter I get:
(TELX|T)
Same goes for copy/paste of selected text, it doesn't select the correct text (selects same amount from other side).

Even worse in textboxes (like in save file). Not only inserting is wrong, but there's also a sliding problem - when you select text, it changes while you're selecting it (slides into selection from other side). I've seen that in a few apps before, makes editing text impossible.

Could at least partial support be improved? I mean, ignore keys and RTL layout. At least should be WYSIWYG in insert/delete/select/copy. BTW, export to PDF reversed Hebrew text. Expectable, but not nice.

Hey, Sergei, I have told you that RTL is not implemented.

In fact, so far there was nobody capable about even commenting this issue. We could base it on some generic info available, but I guess this would not really help... We need a real person who knows how this is supposed to work;)

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 12:22:50 GMT
View Forum Message <> Reply to Message

Well, I just hoped that it might simply work... I could explain whatever you want to know but I doubt that I'll be able to code handling myself.

I'm trying to compile HomeBudget - GridCtrl/GridDisplay.h didn't have NAMESPACE_UPP.
sqllite3 has stuff like this:
typedef struct sqlite3_stmt sqlite3_stmt;
That have to be commented out to work in C++. Now, do I have to install SQL or something to test this project? I don't know how SQL works...

More: GridCtrl contains stuff like this: LG("speedx %d, speedy %d", speedx, speedy). LG is:
#define LG(x) s << x << '\n'. So I believe something is wrong about the syntax.

I'm not sure if I'll be able to make sqllite work with SCU, since it's 2MB of source. Actually, the biggest package of U++... Better finish unicode (testing UWord).

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 19 Sep 2007 21:06:17 GMT
View Forum Message <> Reply to Message

Quote:
In Debug.cpp, sLogFile, FromSysChrSet causes a segmentation fault on program (correct) shutdown in Code::Blocks/MinGW (with UNICODE, otherwise it isn't called).

I do not get this... Something else must be broken.

Quote:
For things such as clipboard, To/From Ansi/Unicode charset probably should be defined separately. It's possible that Ansi strings need to be posted on clipboard from a unicode app (maybe vice versa too, but rather unlikely).

Win32API explicitly states that ANSI - UNICODE conversions are performed.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 19 Sep 2007 21:56:09 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 19 September 2007 23:06Quote:
In Debug.cpp, sLogFile, FromSysChrSet causes a segmentation fault on program (correct) shutdown in Code::Blocks/MinGW (with UNICODE, otherwise it isn't called).

I do not get this... Something else must be broken.

Quote:
For things such as clipboard, To/From Ansi/Unicode charset probably should be defined separately. It's possible that Ansi strings need to be posted on clipboard from a unicode app (maybe vice versa too, but rather unlikely).

Win32API explicitly states that ANSI - UNICODE conversions are performed.

Mirek

OK about clipboard...

In Debug.cpp, did FromSysChrSet do Unicode->Ansi conversion? Try to add this to sLogFile function:

```
static char out[1024];
FromUnicode(out, s, wstrlen(s), CHARSET_DEFAULT);
```

With some wchar[] as s.

Though it could be just a GDB problem...

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 20 Sep 2007 03:39:28 GMT
View Forum Message <> Reply to Message

Sergei, now I am a bit confused. Please, what is crashing, the current U++ code or your Log.cpp patched to use UNICODE?

(BTW, there is IMO no need to use UNICODE there... the names of .log files are simple and fixed, as long as you do not insist about puttin hebrew characters to .exe file names - and even then it would work).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Thu, 20 Sep 2007 11:29:29 GMT

I didn't want to use Unicode, I really don't care how wrong it writes in debug, that's why I initaially left SysChrSet and didn't replace with SsytemCharset like everywhere else. But:

::GetModuleFileName(NULL, fn, 512);

In UNICODE fn has to be const WCHAR*, and it's char*. So I used the WinCE code:

```
wchar wfn[256];
::GetModuleFileName(NULL, wfn, 512);
strcpy(fn, FromSysChrSet(wfn));
```

What crashes is this function in Debug.cpp:

```
#ifdef PLATFORM_WIN32
static void sLogFile(char *fn, const char *app = ".log")
{
#ifdef PLATFORM_WINCE
 wchar wfn[256];
 ::GetModuleFileName(NULL, wfn, 512);
 strcpy(fn, FromSysChrSet(wfn));
#else
 ::GetModuleFileName(NULL, fn, 512);
#endif
 char *e = fn + strlen(fn), *s = e;
 while(s > fn && *--s != '\\' && *s != '.')
  ;
 strcpy(*s == '.' ? s : e, app);
}
#endif
```

It's called after APP_MAIN finishes execution. It crashes if you change to this (don't define UNICODE or anything else, crashes "out of the box"):

```
#ifdef PLATFORM_WIN32
static void sLogFile(char *fn, const char *app = ".log")
{
#ifdef PLATFORM_WINCE
 wchar wfn[256];
 ::GetModuleFileName(NULL, wfn, 512);
 strcpy(fn, FromSysChrSet(wfn));
#else
 wchar wfn[256];
 ::GetModuleFileNameW(NULL, wfn, 512);
 FromUnicode(fn, wfn, wstrlen(wfn), CHARSET_DEFAULT);
#endif
 char *e = fn + strlen(fn), *s = e;
 while(s > fn && *--s != '\\' && *s != '.')
```

```
  ;
 strcpy(*s == '.' ? s : e, app);
}
#endif
```

OK. Not "out of the box". Yesterday it crashed, today GDB just gets stuck and doesn't terminate the program. But the problem is there, since in MSVC it also crashes in debug (unhandled exception in vcont.h). It's the same bug/something, in Code::Blocks it also got segmentation fault somewhere in vectors from FromUnicode call. And both times it happened inside here:

```
inline
static CharSetData& s_cset(byte charset)
{
 return s_map()[ResolveCharset(charset)];
}
```

Hope that helps.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 20 Sep 2007 12:51:57 GMT
View Forum Message <> Reply to Message

Ah, I see, obviously!

Well, what about using A version (GetModuleFileNameA) of function instead of converting?

I guess the reason of crash has a lot to do with what I have stated before: LOG has to work ALWAYS.

I believe that you have altered it in a way that makes it depend on Charset system. And it gets called when the Charset system is destructed (at the end of program).

BTW, if you fix this using "A" versions, please check the log. IMO it is quite likely that there is some diagnostic info there logged during the program exit (what likely happened: Some problem during exit/destriction, being logged -> log crashed).

Mirek

P.S.: Actually, yes, now I remeber that back when I was testing WinCE U++ ("experimental"), it was crashing at the end of program

---

Subject: Re: Building & using U++ without TheIDE

Posted by sergei on Thu, 20 Sep 2007 13:11:09 GMT
View Forum Message <> Reply to Message

Hehe, known yet forgotten bug

I changed to A version, obviously it works (it's the same it was before - no unicode), and I see no log.

Of course debug could use A versions everywhere, but why not just save the possible log filename in both char[] and wchar[] variants on load? That way no conversion would be necessary. Or does log filename change during execution?

P.S. Stumbled upon "Panic" message. How about something more professional-looking, like "Exception" or "Fatal Error"?

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Thu, 20 Sep 2007 19:57:50 GMT
View Forum Message <> Reply to Message

Update: I'm nearly done (at least for the first release) making U++ SCU-compatible. Console and GUI work, MSVC seems to work in Debug. .lay, .iml, .t work. zlib was updated to 1.2.3 and modified to work, png was slightly modified (probably should update later, it's 2001 version, there's 2007 version on the website) and works, gif and bmp worked out of the box, jpg was seriously modified (I hate that C-style polymorphism...) but now works. Left is tif. Other stuff is untested, but sqlite3 definitely does not work .

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Fri, 21 Sep 2007 01:36:14 GMT
View Forum Message <> Reply to Message

I'm testing which packages compile as SCU and which don't. I've encountered some problems:

Inner Geom packages use T??? includes. Commenting them doesn't help, since geomcoord has this:

```
struct Tree : RefBase
 {
  Tree(pick_ Node& root) : root(root) {}

  Node root;
 };
```

RefBase is only defined in Web. Is that the same RefBase? If yes, GeomCoord should have uses Web. If not, what RefBase does it need, and there is then naming conflict.

Ole Ctrl has member of type Thread. But I'm not compiling multithreaded. Is multithreading a

requirement for Ole Ctrl, or should there be some kind of #ifdef?

MySql didn't compile. I don't have #include <Sql.h>, and I don't know where MYSQL is defined (I need to install MySQL?). PostgreSQL wants #include <libpq-fe.h>, I have to install something too, right?

Web/SSL wants #include <openssl/ssl.h> and #include <openssl/err.h>...

Zim, and removed stuff in png, gif and Draw/hrr use AlphaArray, which is never defined -> can't build Zim.

List of compileable packages (in MSVC) ATM (commented ones don't compile):

```
#include <UppPkg/Core.h>
#include <UppPkg/CppBase.h>
#include <UppPkg/Crypto.h>
#include <UppPkg/CtrlCore.h>
#include <UppPkg/CtrlLib.h>
#include <UppPkg/Draw.h>
#include <UppPkg/Esc.h>
//#include <UppPkg/Geom_Coords.h>
//#include <UppPkg/Geom_Ctrl.h>
//#include <UppPkg/Geom_Draw.h>
#include <UppPkg/Geom.h>
#include <UppPkg/GLCtrl.h>
#include <UppPkg/GridCtrl.h>
//#include <UppPkg/MySql.h>
//#include <UppPkg/Ole_Ctrl.h>
#include <UppPkg/Ole.h>
#include <UppPkg/OleDB.h>
#include <UppPkg/Oracle.h>
#include <UppPkg/PdfDraw.h>
#include <UppPkg/plugin_bmp.h>
//#include <UppPkg/plugin_bz2.h>
#include <UppPkg/plugin_dbf.h>
#include <UppPkg/plugin_ftp.h>
#include <UppPkg/plugin_gif.h>
#include <UppPkg/plugin_jpg.h>
//#include <UppPkg/plugin_ndisasm.h>
//#include <UppPkg/plugin_pcre.h>
#include <UppPkg/plugin_png.h>
//#include <UppPkg/plugin_sqlite3.h>
//#include <UppPkg/plugin_tif.h>
#include <UppPkg/plugin_z.h>
//#include <UppPkg/plugin_zim.h>
//#include <UppPkg/PostgreSQL.h>
#include <UppPkg/Report.h>
```

```
#include <UppPkg/RichEdit.h>
#include <UppPkg/RichText.h>
#include <UppPkg/Sql.h>
#include <UppPkg/SqlCtrl.h>
//#include <UppPkg/Web_SSL.h>
#include <UppPkg/Web.h>
```

I believe ndisasm should be excluded unless building TheIDE, since it's GPL, right?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Fri, 21 Sep 2007 07:49:25 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 20 September 2007 21:36I'm testing which packages compile as SCU and which don't. I've encountered some problems:

Inner Geom packages use T??? includes. Commenting them doesn't help, since geomcoord has this:

```
struct Tree : RefBase
 {
  Tree(pick_ Node& root) : root(root) {}

  Node root;
 };
```

RefBase is only defined in Web. Is that the same RefBase? If yes, GeomCoord should have uses Web. If not, what RefBase does it need, and there is then naming conflict.

Sorry for the mess. Geom is not "canonical".

Quote:
Ole Ctrl has member of type Thread. But I'm not compiling multithreaded. Is multithreading a requirement for Ole Ctrl, or should there be some kind of #ifdef?

Yes. Which makes us aware abot a new problem (Not only debug/release, but also single/multithreaded).

Quote:
MySql didn't compile. I don't have #include <Sql.h>, and I don't know where MYSQL is defined (I need to install MySQL?). PostgreSQL wants #include <libpq-fe.h>, I have to install something too, right?

Yes and yes. I guess this rules out the possibility to have them in the same library.

Quote:
Web/SSL wants #include <openssl/ssl.h> and #include <openssl/err.h>...

Same issue.

Quote:
Zim, and removed stuff in png, gif and Draw/hrr use AlphaArray, which is never defined -> can't build Zim.

Noncanonical mess On the positive note, your efforts will finally detect all these problems

List of compileable packages (in MSVC) ATM (commented ones don't compile):
[/quote]

OK, these commented out packages are canonical for non-SQL development:

//#include <UppPkg/plugin_bz2.h>
//#include <UppPkg/plugin_pcre.h>
//#include <UppPkg/plugin_tif.h>

Quote:
I believe ndisasm should be excluded unless building TheIDE, since it's GPL, right?

Yes.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Fri, 21 Sep 2007 11:41:58 GMT
View Forum Message <> Reply to Message

When you say non-canonical, should I understand that as "not-yet-working" and exclude it for now?

Multithreading is a problem, turns out... So there's a need for 2 release libs, at the very least. SQL too, can't it be turned into stand-alone? Just install whatever is needed and add the necessary headers to the packages. Or simply compile a lib on a computer that has everything installed. Having separate libs for SQL would be too many libs...

The three packages you marked seem canonical for me too. It's just that they don't work in SCU

out-of-the-box. Naming conflicts etc. (should be resolveable). bz2, libtiff, libpng could/should be updated.

Others that don't work in SCU are ndisasm and sqlite3.

I'll rebuild, check out warnings, and package what I've done. What doesn't work can be fixed later. I don't want to get out of sync with the main sources.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Fri, 21 Sep 2007 22:16:10 GMT
View Forum Message <> Reply to Message

sergei wrote on Fri, 21 September 2007 07:41When you say non-canonical, should I understand that as "not-yet-working" and exclude it for now?

Well, they are working, but are not core libs and are not polished.

Quote:
I'll rebuild, check out warnings, and package what I've done. What doesn't work can be fixed later. I don't want to get out of sync with the main sources.

OK.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Fri, 21 Sep 2007 22:18:59 GMT
View Forum Message <> Reply to Message

OK, I thought everything worked, I was wrong

I've built an example with all the "working" packages under MSVC, debug was built, it worked. Building the same project under MinGW in release worked. But in MinGW/debug it didn't. Instead of working, debugger just hanged without reaching the first line of the program. Executing resulted in a memory could not be written error.

This scenario works in debug:

```
#include <UppPkg/Core.h>
#include <UppPkg/CppBase.h>
#include <UppPkg/Crypto.h>
#include <UppPkg/CtrlCore.h>
#include <UppPkg/CtrlLib.h>
```

```
#include <UppPkg/Draw.h>
#include <UppPkg/Esc.h>
#include <UppPkg/Geom.h>
#include <UppPkg/GLCtrl.h>
#include <UppPkg/GridCtrl.h>
#include <UppPkg/Ole.h>
#include <UppPkg/PdfDraw.h>
#include <UppPkg/plugin_bmp.h>
//#include <UppPkg/plugin_dbf.h>
//#include <UppPkg/plugin_ftp.h>
#include <UppPkg/plugin_gif.h>
//#include <UppPkg/plugin_jpg.h>
#include <UppPkg/plugin_png.h>
#include <UppPkg/plugin_z.h>
#include <UppPkg/Report.h>
#include <UppPkg/RichEdit.h>
#include <UppPkg/RichText.h>
//#include <UppPkg/Web.h>
```

Uncommenting any package results in the bug I mentioned above. I tried to see what happens there. Uncommented dbf, excluded the class DbfStream itself - works. Seems like having the class in the code results in this weird bug in debug mode. Other packages are bigger and more difficult to debug, but I doubt there's any serious error since it works in release and in MSVC's debug.

Becoming more interesting: removing GridCtrl makes it possible to use dbf, ftp and jpg (but not Web), without the bug. With GridCtrl none of them can be used.

OK, I had an idea and removed #define flagDEBUG and #define flagDEBUG_FULL (while still using MinGW/debug). Guess what, it works. Removing only debug_full wasn't enough, but removing flagDEBUG too solved the problem. So, what exactly does flagDEBUG do that could cause thing go wrong?

P.S. I have a feeling it has something to do with LG/LOG/LLOG/... #defines. GridCtrl redefines LG, but removing it and replacing all LG with LGR in it didn't help...

Edit: I found that I incorrectly used GridCtrl. As Mirek said, although usually the main header is the first file, it's not always the case. Well, that's the example... But unfortunately fixing it and renaming LG to LGD (so that it wouldn't conflict with CtrlCore's log) didn't make any difference. Uncommenting either GridCtrl or Web makes debug break.

Subject: Re: Building & using U++ without TheIDE

Posted by mirek on Sun, 23 Sep 2007 08:54:17 GMT
View Forum Message <> Reply to Message

sergei wrote on Fri, 21 September 2007 18:18
P.S. I have a feeling it has something to do with LG/LOG/LLOG/... #defines. GridCtrl redefines LG, but removing it and replacing all LG with LGR in it didn't help...

Ah, of course. Manual SCU approach.

Do you undefine macros defined in .cpp?

(BLITZ does this automatically).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 23 Sep 2007 14:22:48 GMT
View Forum Message <> Reply to Message

luzr wrote on Sun, 23 September 2007 10:54sergei wrote on Fri, 21 September 2007 18:18
P.S. I have a feeling it has something to do with LG/LOG/LLOG/... #defines. GridCtrl redefines LG, but removing it and replacing all LG with LGR in it didn't help...

Ah, of course. Manual SCU approach.

Do you undefine macros defined in .cpp?

(BLITZ does this automatically).

Mirek

No. That's why I'm trying to ensure there are no naming conflicts. Macros are evil

But since that works in MSVC I doubt that would be a naming clash bug or something similar. Maybe I should try MinGW 4.2 instead of 3.4.5.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 23 Sep 2007 18:51:01 GMT
View Forum Message <> Reply to Message

OK, I've finalized everything I did. Not perfect, but works. Everything is in the zip, there's

readme.txt and changelog.txt. Any comments/suggestions/questions are welcome. Would be interesting to see times/sizes on other computers.

File Attachments

1) U++.zip, downloaded 427 times

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 25 Sep 2007 09:44:37 GMT
View Forum Message <> Reply to Message

I think it would be better to solve INITBLOCK SCU issue by using

BLITZ_INDEX__

macro.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 25 Sep 2007 14:18:29 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 September 2007 11:44I think it would be better to solve INITBLOCK SCU issue by using

BLITZ_INDEX__

macro.

But where/how is BLITZ_INDEX__ defined? Especially if BLITZ isn't present.

Did you try to use lib or SCU? Did any change make it into the main source?

P.S. since U++ and many plugins are BSD-style licensed and programs are statically linked, what exactly should be present in the documentation of a program using U++? One copyright notice per package/plugin? Is there a compilation of licenses to be included?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 25 Sep 2007 21:07:42 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 25 September 2007 10:18luzr wrote on Tue, 25 September 2007 11:44I think it would be better to solve INITBLOCK SCU issue by using

BLITZ_INDEX__

macro.

But where/how is BLITZ_INDEX__ defined? Especially if BLITZ isn't present.

Did you try to use lib or SCU? Did any change make it into the main source?

You have to define it in SCU.

The imporant thing is that INITBLOCK will use this macro to make its internally generated static function name unique.

Quote:
P.S. since U++ and many plugins are BSD-style licensed and programs are statically linked, what exactly should be present in the documentation of a program using U++? One copyright notice per package/plugin? Is there a compilation of licenses to be included?

Actuallly, I have this in ToDo - it should be easily possible to generate license info by TheIDE for the current project

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 25 Sep 2007 22:52:36 GMT
View Forum Message <> Reply to Message

But how am I supposed to define that? In general case it would have to be redefined before every file, since __LINE__ isn't unique, and stuff like counter in MSVC aren't portable. The headers would be full of #undef and #define. Once replacing all these INIT/EXITBLOCK and manually adding them unique identifiers like I did should solve the issue (2 seconds more typing in case you'll add another INITBLOCK to give it a name - filename usually is unique enough).

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 26 Sep 2007 11:41:57 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 25 September 2007 18:52But how am I supposed to define that? In general case it would have to be redefined before every file, since __LINE__ isn't unique, and stuff like counter in MSVC aren't portable. The headers would be full of #undef and #define.

Yes. But you do not need to fix existing code.

Quote:
Once replacing all these INIT/EXITBLOCK and manually adding them unique identifiers like I did should solve the issue (2 seconds more typing in case you'll add another INITBLOCK to give it a name - filename usually is unique enough).

Ah, so you do think I should remove nameless INITBLOCK?

I am not very happy about such idea... I hate inventing unique ids with the only purpose - being unique.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 26 Sep 2007 13:13:56 GMT
View Forum Message <> Reply to Message

I'd love a unique ID generator, but that wouldn't be portable. This is the one and only reason I don't like the parameterless INITBLOCK. I don't say you should remove it, it might be useful when you're sure line numbers won't clash. But when you're not, just supply a unique name. Besides the existing INITBLOCKs, how many more you want to add? 10? 20? That would be 2 minutes of work to make the code portable.

P.S. I personally dislike the idea of INITBLOCK. Such code would be difficult to debug in case something goes wrong, since it executes before main. And it's scattered in the source.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 26 Sep 2007 16:25:26 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 26 September 2007 09:13I'd love a unique ID generator, but that wouldn't be portable.

Current implmentation IS FULLY PORTABLE. What makes it unportable is SCU. You cannot simply include everything into a single file and expect it to work - that is not 100% C/C++ compatible.

Quote:
P.S. I personally dislike the idea of INITBLOCK. Such code would be difficult to debug in case something goes wrong, since it executes before main. And it's scattered in the source.

That is the point. Module intialization is placed into the module. No action is required by client besides adding the module.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 26 Sep 2007 16:44:29 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 26 September 2007 18:25sergei wrote on Wed, 26 September 2007 09:13I'd love a unique ID generator, but that wouldn't be portable.

Current implmentation IS FULLY PORTABLE. What makes it unportable is SCU. You cannot simply include everything into a single file and expect it to work - that is not 100% C/C++ compatible.

Quote:
P.S. I personally dislike the idea of INITBLOCK. Such code would be difficult to debug in case something goes wrong, since it executes before main. And it's scattered in the source.

That is the point. Module intialization is placed into the module. No action is required by client besides adding the module.

Mirek

I was talking about your BLITZ_INDEX__ suggestion - that would work only with BLITZ. INITBLOCK without BLITZ, is indeed portable without SCU. I agree that SCU isn't C/C++ compatible. But once it's made to work, it can be maintained relatively easily. + If code works with SCU you don't have to worry about naming if you wish to transfer some code from one file to another. But why argue? Just tell me if it's fine to change the INITBLOCKs to INITBLOCK_s like I did, or I'll have to find another way around.

I'm not against the idea that the only action required is adding the module, I actually like that. What I dislike is the code scattered among constructors. There is no easy way to go through these initializations while debugging, one by one.

I'm probably unclear, so here's my idea: create a global array of pointers to functions (get void, return void) - like 1000 cells, and an index saying how many pointers are actually stored. In INITBLOCKs, instead of doing something, the only action would be to append the initialization function to the global array (and increment index). That way, when main is started, you'll have an array with pointers to functions you have to call to initialize everything. Add a for loop there, that would initialize things. The pro is that you then can debug these initializations by stepping in during the loop. You'll also be able to do something before the initializations, in case that will ever

be necessary (who knows, charset setup or something). Additionally, INITBLOCK could look like INITBLOCK(MyPkgInitFunc), and as long as the initialization function name is meaningful, it probably would be unique. MyPkgInitFunc would contain all the register stuff, and it could be debugged. Con is that you don't know how many INITBLOCKs will be, so you'll have to statically allocate a big enough array of function pointers.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 26 Sep 2007 17:26:25 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 26 September 2007 12:44luzr wrote on Wed, 26 September 2007 18:25sergei wrote on Wed, 26 September 2007 09:13I'd love a unique ID generator, but that wouldn't be portable.

Current implmentation IS FULLY PORTABLE. What makes it unportable is SCU. You cannot simply include everything into a single file and expect it to work - that is not 100% C/C++ compatible.

Quote:
P.S. I personally dislike the idea of INITBLOCK. Such code would be difficult to debug in case something goes wrong, since it executes before main. And it's scattered in the source.

That is the point. Module intialization is placed into the module. No action is required by client besides adding the module.

Mirek

I was talking about your BLITZ_INDEX__ suggestion - that would work only with BLITZ.

Why? You can use it in any SCU system. Actually, it just makes U++ sources more usable with SCU approach.

Quote:
But once it's made to work, it can be maintained relatively easily.

Once you create some release system, it can be as easily maintained as it is.

Quote:
Just tell me if it's fine to change the INITBLOCKs to INITBLOCK_s like I did, or I'll have to find another way around.

Well, I wanted to tell you that I am not going to avoid this feature (I mean, unnamed INITBLOCK) only because of SCU experiment. Means, I do not have a problem with it, but expect to fix these for each release...

Quote:
There is no easy way to go through these initializations while debugging, one by one.

Maybe not one by one, but breakpoints still work....

Quote:
I'm probably unclear, so here's my idea: create a global array of pointers to functions (get void, return void) - like 1000 cells, and an index saying how many pointers are actually stored. In INITBLOCKs, instead of doing something, the only action would be to append the initialization function to the global array (and increment index). That way, when main is started, you'll have an array with pointers to functions you have to call to initialize everything.

Actually, that is exactly how global constructors (and therefore INITBLOCKs) are implemented

Quote:
Add a for loop there, that would initialize things. The pro is that you then can debug these initializations by stepping in during the loop. You'll also be able to do something before the initializations, in case that will ever be necessary (who knows, charset setup or something). Additionally, INITBLOCK could look like INITBLOCK(MyPkgInitFunc), and as long as the initialization
function name is meaningful, it probably would be unique.

Yes, great. So instead of calling initializations directly, we will have an array of functions (global constructors). This array will be performed by C++ runtime to create another array of functions and that one will be performed in main?

Also, if you need to do something more, you simply cannot use INITBLOCK.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 26 Sep 2007 19:01:47 GMT
View Forum Message <> Reply to Message

OK, never mind...

I'll just add BLITZ_INDEX defines everywhere.

Did the other changes (like IsLeapYear) make it into the main source?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 26 Sep 2007 20:56:28 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 26 September 2007 15:01OK, never mind...

I'll just add BLITZ_INDEX defines everywhere.

Did the other changes (like IsLeapYear) make it into the main source?

Working on it... But majority is INITBLOCK anyway...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 09 Oct 2007 17:14:16 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 26 September 2007 22:56sergei wrote on Wed, 26 September 2007 15:01OK, never mind...

I'll just add BLITZ_INDEX defines everywhere.

Did the other changes (like IsLeapYear) make it into the main source?

Working on it... But majority is INITBLOCK anyway...

Mirek

I see that 710dev1 is released, but AFAIK most of my proposed changes weren't applied
Nevermind the INITBLOCKs, but why the other changes didn't make it?

I didn't have time yet to thoroughly test it, but these changes seem to still apply:

Info:
Packages coff/uar/uld/uar.upp and Geom/Coords/Ctrl/Ctrl.upp should be removed from source tree.

Core:
TimeDate.cpp - Removed #define IsLeapYear.
TimeDate.h - Added #define IsLeapYear since it has to be visible is CtrlLib/DateTimeCtrl.
z.cpp - removed enum, ASCII_FLAG, HEAD_CRC, EXTRA_FIELD, ORIG_NAME, COMMENT,

RESERVED already defined in plugin/z,
GZ_MAGIC1, GZ_MAGIC2 were replaced with gz_magic[0], gz_magic[1] defined in plugin/z.

CtrlLib:
DateTimeCtrl.h - commented IsLeapYear function - already #defined in Draw/TimeDate.

Draw:
MetaFile.cpp - IsClipboardFormatAvailable replaced with ::IsClipboardFormatAvailable to resolve naming conflict.
Palette.cpp - BINS, BINSHIFT replaced with palBINS, palBINSHIFT to resolve naming conflict with BINS in Core/lheap.

Esc
Esc.h - #ifndef/#define ESC_H replaced with _ESC_H_ to resolve naming conflict with plugin/pcre.

GridCtrl:
GridCtrl.upp - reordered files to ensure first file is the main header of the package.
(something was changed about LG, might/might not work now)

PdfDraw:
PdfDraw.upp - removed Test.cpp - test program, not part of the package.

RichEdit:
Find.cpp - ReplaceText() renamed to ReplacementText() to resolve naming conflict with Win32 API function.
RichEdit.h - ReplaceText() renamed to ReplacementText() to resolve naming conflict with Win32 API function.

plugin/png:
png.h - marked modification.
pngread.c - added casts.
pngrutil.c - added casts.
pngset.c - added casts.

plugin/jpg:
Whole plugin rework.
Appended filename to functions marked LOCAL to resolve naming conflicts.
Added include guards.
Modified #define/typedef names to resolve naming conflicts.

plugin/z:
Whole plugin rework. Updated to version 1.2.3.
Changed function declarations from K&R to ASNI C style.
Added include guards.
Modified #define/typedef names to resolve naming conflicts.

Though I'm really glad to see unicode (W versions) implemented so quickly. I'll try my unicode files/registry test later.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 09 Oct 2007 21:39:13 GMT
View Forum Message <> Reply to Message

Sergei, I am very sorry about it. Too much happening now, I only remembered I have a job to do in this thread after Daniel annouced the release..

Going to do that right now not to forget again.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 09 Oct 2007 22:02:26 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 09 October 2007 13:14
Info:
 Packages coff/uar/uld/uar.upp and Geom/Coords/Ctrl/Ctrl.upp should be removed from source tree.

Core:
 TimeDate.cpp - Removed #define IsLeapYear.
 TimeDate.h - Added #define IsLeapYear since it has to be visible is CtrlLib/DateTimeCtrl.

CtrlLib:
 DateTimeCtrl.h - commented IsLeapYear function - already #defined in Draw/TimeDate.

Draw:
 MetaFile.cpp - IsClipboardFormatAvailable replaced with ::IsClipboardFormatAvailable to resolve naming conflict.
 Palette.cpp - BINS, BINSHIFT replaced with palBINS, palBINSHIFT to resolve naming conflict with BINS in Core/lheap.

Esc
 Esc.h - #ifndef/#define ESC_H replaced with _ESC_H_ to resolve naming conflict with plugin/pcre.

GridCtrl:
 GridCtrl.upp - reordered files to ensure first file is the main header of the package.
(something was changed about LG, might/might not work now)

PdfDraw:
 PdfDraw.upp - removed Test.cpp - test program, not part of the package.

Above are used.

Quote:
RichEdit:
 Find.cpp - ReplaceText() renamed to ReplacementText() to resolve naming conflict with Win32 API function.
 RichEdit.h - ReplaceText() renamed to ReplacementText() to resolve naming conflict with Win32 API function.

I see no naming conflict; it is a method, not a global function. Well, it might get macro-replace with ReplaceTextA, but who cares?

Quote:
plugin/png:
 png.h - marked modification.
 pngread.c - added casts.
 pngrutil.c - added casts.
 pngset.c - added casts.

plugin/jpg:
 Whole plugin rework.
 Appended filename to functions marked LOCAL to resolve naming conflicts.
 Added include guards.
 Modified #define/typedef names to resolve naming conflicts.

Core
 z.cpp - removed enum, ASCII_FLAG, HEAD_CRC, EXTRA_FIELD, ORIG_NAME, COMMENT, RESERVED already defined in plugin/z,
  GZ_MAGIC1, GZ_MAGIC2 were replaced with gz_magic[0], gz_magic[1] defined in plugin/z.

plugin/z:
 Whole plugin rework. Updated to version 1.2.3.
 Changed function declarations from K&R to ASNI C style.
 Added include guards.
 Modified #define/typedef names to resolve naming conflicts.

Is it really that good idea to change 3rd party code?

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 09 Oct 2007 22:51:58 GMT
View Forum Message <> Reply to Message

Quote:Well, it might get macro-replace with ReplaceTextA, but who cares?

Exactly. ReplaceText is a macro and thus doesn't care what is a global function and what isn't. In my case it replaced it with ReplaceTextA, instead of using method, in this call for example: Insert(cursor, ReplaceText(), false);

Quote:Is it really that good idea to change 3rd party code?

Not as a general rule. But U++ 3rd party plugins are mostly under BSD-style license, so, unlike GPL, there should be absolutely no problems modifying it (that's the point of BSD, isn't it). + Last zlib release is from 2005, jpeg from 1998, so it's rather unlikely any of them will be changed anytime soon (ever?). Thus IMHO modifying these two shouldn't be a problem - one time change and that's it. While SCU reasoning probably won't appeal to you, I'll note that some modifications were also due to non-C++ code (K&R style, missing casts, use of this as variable name, defines conflicting with code in other packages).

Some other libraries (tiff, png) are still maintained, these should be examined each. PNG should be used IMHO since the modifications are merely casts required by C++ but not by C. TIFF and others, I haven't checked yet.

Some "tough" libraries like sqlite could possibly be built as a separate lib. For non-core plugins (like zlib) this makes sense - every time such a library is updated, plugin could be rebuilt. Can TheIDE use libs internally?

P.S. Checked unicode on files, works out of the box Now U++ has 99% unicode support (except for surrogate pairs).

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 11 Oct 2007 03:40:26 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 09 October 2007 18:51Quote:Well, it might get macro-replace with ReplaceTextA, but who cares?

Exactly. ReplaceText is a macro and thus doesn't care what is a global function and what isn't. In my case it replaced it with ReplaceTextA, instead of using method, in this call for example: Insert(cursor, ReplaceText(), false);

As macro, it should be substituted

Insert(cursor, ReplaceTextA(), false);

But the method name is ReplaceTextA() too...

I just wonder, was it some real problem to solve?

Quote:
P.S. Checked unicode on files, works out of the box Now U++ has 99% unicode support (except for surrogate pairs).

Well, I am afraid there are more details to care about... But time will tell (also, network names are not unicode(d) yet, but in fact, NetNode is not yet used anywhere).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 14 Oct 2007 22:35:49 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 11 October 2007 05:40sergei wrote on Tue, 09 October 2007 18:51Quote:Well, it might get macro-replace with ReplaceTextA, but who cares?

Exactly. ReplaceText is a macro and thus doesn't care what is a global function and what isn't. In my case it replaced it with ReplaceTextA, instead of using method, in this call for example:
Insert(cursor, ReplaceText(), false);

As macro, it should be substituted

Insert(cursor, ReplaceTextA(), false);

But the method name is ReplaceTextA() too...

I just wonder, was it some real problem to solve?

Mirek

Purely compilation problem. ReplaceText() is a member function. ReplaceText is a macro expanding to ReplaceTextA. In the Insert call, the member function call was supposed to be executed but actually the macro was used, and thus ReplaceTextA. Since signatures didn't match - compilation error. My suggested solution was to call member function ReplacementText(), or maybe call it explicitely (this->ReplaceText()).

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 15 Oct 2007 05:27:51 GMT
View Forum Message <> Reply to Message

sergei wrote on Sun, 14 October 2007 18:35luzr wrote on Thu, 11 October 2007 05:40sergei wrote on Tue, 09 October 2007 18:51Quote:Well, it might get macro-replace with ReplaceTextA, but who cares?

Exactly. ReplaceText is a macro and thus doesn't care what is a global function and what isn't. In my case it replaced it with ReplaceTextA, instead of using method, in this call for example: Insert(cursor, ReplaceText(), false);

As macro, it should be substituted

Insert(cursor, ReplaceTextA(), false);

But the method name is ReplaceTextA() too...

I just wonder, was it some real problem to solve?

Mirek

Purely compilation problem. ReplaceText() is a member function. ReplaceText is a macro expanding to ReplaceTextA. In the Insert call, the member function call was supposed to be executed but actually the macro was used, and thus ReplaceTextA. Since signatures didn't match - compilation error. My suggested solution was to call member function ReplacementText(), or maybe call it explicitely (this->ReplaceText()).

Sorry, I still do not get it.

ReplaceText is defined by Win32 as ReplaceTextA. EVERYWHERE. Means in .h too.

Therefore, the method name, after macro replacement, is ReplaceTextA too. Call is to ReplaceTextA.

Means signatures DO MATCH.

(Note that this obviously compiles without a problem with TheIDE).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 15 Oct 2007 08:48:20 GMT

[quote title=luzr wrote on Mon, 15 October 2007 01:27]sergei wrote on Sun, 14 October 2007 18:35luzr wrote on Thu, 11 October 2007 05:40sergei wrote on Tue, 09 October 2007 18:51Quote:Well, it might get macro-replace with ReplaceTextA, but who cares?

Exactly. ReplaceText is a macro and thus doesn't care what is a global function and what isn't. In my case it replaced it with ReplaceTextA, instead of using method, in this call for example: Insert(cursor, ReplaceText(), false);

As macro, it should be substituted

Insert(cursor, ReplaceTextA(), false);

But the method name is ReplaceTextA() too...

I just wonder, was it some real problem to solve?

Mirek

Purely compilation problem. ReplaceText() is a member function. ReplaceText is a macro expanding to ReplaceTextA. In the Insert call, the member function call was supposed to be executed but actually the macro was used, and thus ReplaceTextA. Since signatures didn't match - compilation error. My suggested solution was to call member function ReplacementText(), or maybe call it explicitely (this->ReplaceText()).

Ah, I think I know where the problem is. I beleive it is your ruthless SCU approach again - if I preprocess that file in theide, I do not even get ReplaceTextA, because corresponding Win32 header is not included at global level.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Mon, 15 Oct 2007 10:18:24 GMT

That's because you use MSVC, where you include Windows' stuff separately. In MinGW, you include <windows.h> and thus the macro's defined. Unfortunately this->ReplaceText() doesn't help (becomes this->ReplaceTextA()), so I suggest to change the function's name to something like ReplacementText, or TextReplace.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 17 Oct 2007 17:12:06 GMT
View Forum Message <> Reply to Message

sergei wrote on Mon, 15 October 2007 06:18That's because you use MSVC, where you include Windows' stuff separately. In MinGW, you include <windows.h> and thus the macro's defined.

Nope, in MinGW, it stays "ReplaceText". ReplaceTextA is defined in "Commdlg.h" - and that is not visible in RichText.

The real trouble is perhaps the Commdlg.h in your SCU is included AFTER RichEdit.h but BEFORE RichText RichEdit::ReplaceText().

If it would be included before RichEdit.h, everything would work too - both occurences of ReplaceText, in the header and in the .cpp would replaced by ReplaceTextA....

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 17 Oct 2007 19:58:57 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 17 October 2007 19:12sergei wrote on Mon, 15 October 2007 06:18That's because you use MSVC, where you include Windows' stuff separately. In MinGW, you include <windows.h> and thus the macro's defined.

Nope, in MinGW, it stays "ReplaceText". ReplaceTextA is defined in "Commdlg.h" - and that is not visible in RichText.

The real trouble is perhaps the Commdlg.h in your SCU is included AFTER RichEdit.h but BEFORE RichText RichEdit::ReplaceText().

If it would be included before RichEdit.h, everything would work too - both occurences of ReplaceText, in the header and in the .cpp would replaced by ReplaceTextA....

Mirek

Aha... there had to be a reason why it worked in TheIDE

Indeed in SCU RichEdit.h is included way before Find.cpp (which uses ReplaceText). And using MinGW in TheIDE, it doesn't stay ReplaceText, it becomes ReplaceTextA in both places (so I guess whatever defined ReplaceText macro got included earlier), and thus works.

Any chance you fix that (rename the method)?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Wed, 17 Oct 2007 20:54:48 GMT
View Forum Message <> Reply to Message

sergei wrote on Wed, 17 October 2007 15:58
Any chance you fix that (rename the method)?

No.

IMO, you require me to fix the code to compile using unstandard broken method. I do not see that as bright idea. You can break the compilation in many ways. Should I fix the code to support them all?

Maybe you should rather think about fixing the build system...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Wed, 17 Oct 2007 23:34:21 GMT
View Forum Message <> Reply to Message

luzr wrote on Wed, 17 October 2007 22:54sergei wrote on Wed, 17 October 2007 15:58
Any chance you fix that (rename the method)?

No.

IMO, you require me to fix the code to compile using unstandard broken method. I do not see that as bright idea. You can break the compilation in many ways. Should I fix the code to support them all?

Maybe you should rather think about fixing the build system...

Mirek

So far, not many things got wrong... AFAIK this is the last one, excluding C (non-C++ plugins). And they all were at least slightly unusual - using macro as function name, using an API function as a member function, having 2 equal enum members in different enums. You have used code to fix MSVC6 bugs - to support non-standard behavior. But I'm not MS

Your BLITZ approach isn't standard either. It's used, because it works, and because it has its benefits - speed. I wouldn't use SCU, if I didn't see how great BLITZ's benefits are - it's like edit-and-continue for library modifications. My SCU approach isn't much different. It's an attempt to mimic BLITZ behavior. Not a perfect attempt, I agree. I used a different include order (first all

headers, then all CPPs) than BLITZ (for each package, first header, then CPPs), and that was the reason for ReplaceText problem.

But it looks like you treat SCU as non-standard and BLITZ as "just fine" - it isn't so. They are slightly different speed-up solutions. BLITZ is smarter regarding macros, but it's still SCU. In a.cpp write int MyVar, in b.cpp write float MyVar. In standard C++ this would compile (variables are local), but both in BLITZ and SCU this would fail. I might be wrong, but I don't think you intended ReplaceText member function to be substituted with ReplaceTextA in all places - this sound like something that can potentially break compilation (like if you define another ReplaceTextA function thinking it's a different one). Something else might work with my SCU but not BLITZ (like using #defines from a header in CPP from another package) - neither good nor intended.

I can change include order in my SCU to make it resemble BLITZ even more, yet this won't make neither SCU nor BLITZ standard.

Fixing the build system does sound more reasonable regarding C plugins. These indeed could be a time-bomb in SCU, especially the updating ones. Except for zlib (which works and is a vital part of Core), others could be packaged as libs (3rd party plugins are usually lib-friendly). That shouldn't be bad for BLITZ too.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Thu, 18 Oct 2007 12:44:11 GMT
View Forum Message <> Reply to Message

Well, but BLITZ is at this stage well defined and integral part of U++. It is sort of stardard SCU approach for U++. Why should we support another SCU based build system?

BTW, would not it much easier to just reuse BLITZ for the task? AFAIK, you already have the code that takes uppsrc and generates SCU out of it. So what is the difference from using genuine BLITZ code which in fact does something similar and can even do the same (after a bit of fixing of the code)?

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Thu, 18 Oct 2007 15:49:12 GMT
View Forum Message <> Reply to Message

luzr wrote on Thu, 18 October 2007 14:44Well, but BLITZ is at this stage well defined and integral part of U++. It is sort of stardard SCU approach for U++. Why should we support another SCU based build system?

BTW, would not it much easier to just reuse BLITZ for the task? AFAIK, you already have the code that takes uppsrc and generates SCU out of it. So what is the difference from using genuine BLITZ code which in fact does something similar and can even do the same (after a bit of fixing of the code)?

Mirek

Why not? I can't just reuse BLITZ, since my code doesn't generate a big CPP file. It generates headers, that through the use of #includes implement SCU in any given compiler. If I reused BLITZ, I'd have to regenerate uppsrc on every modification - which would require some custom build setup - for every IDE. In my SCU, only if there is a serious change to the filesystem - add/remove file/package headers have to be regenerated. The SCU part is performed by the compiler's preprocessor. But as I said, if ReplaceText is the problem I think I can get around that.

I'm trying to make it simpliest to get running - it should look just like a library that you include/link, not an intimidating framework requiring own build process.

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 21 Oct 2007 00:54:20 GMT
View Forum Message <> Reply to Message

Update: I manually modified 710dev1 tree with the accepted changes, plus I had to add my modified z and png plugins (they're referenced in CtrlLib). I updated pkggen, now it handles BLITZ_INDEX__ and so the INITBLOCKs work the way they should.

Regarding ReplaceText, this error now happens only in MSVC, not MinGW. That's because in MinGW, <windows.h> is included. There, ReplaceText is defined and both declaration and definition become ReplaceTextA. In MSVC however, <windows.h> is split into several definitions, and <commdlg.h> is only included in Win32.cpp, in CtrlLib. Thus, in my SCU, declaration is ReplaceText, because only headers have been processed, and definition is ReplaceTextA, since Win32.cpp already was included. This incompatibility should be easily fixed by moving the include out of Win32.cpp into CtrlLib.h or some header included by it. Or by using <windows.h> in MSVC too. Why neither of these happened in the first place (and include was placed in CPP)?

By the way, NetNode requires mpr.lib, and it isn't Unicodized though a .dli exists. Why is it part of Core (adding another dependency)?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sun, 21 Oct 2007 20:26:59 GMT
View Forum Message <> Reply to Message

sergei wrote on Sat, 20 October 2007 20:54
Regarding ReplaceText, this error now happens only in MSVC, not MinGW. That's because in MinGW, <windows.h> is included. There, ReplaceText is defined and both declaration and definition become ReplaceTextA. In MSVC however, <windows.h> is split into several definitions, and <commdlg.h> is only included in Win32.cpp, in CtrlLib. Thus, in my SCU, declaration is ReplaceText, because only headers have been processed, and definition is ReplaceTextA, since Win32.cpp already was included. This incompatibility should be easily fixed by moving the include out of Win32.cpp into CtrlLib.h or some header included by it. Or by using <windows.h> in MSVC too. Why neither of these happened in the first place (and include was placed in CPP)?

Actually, include was placed in CPP exactly to avoid problems you are having. There is no need to polute global namespace with more macros than necessarry. This way it polutes only CtrlLib, which is easily handled.

Quote:
By the way, NetNode requires mpr.lib, and it isn't Unicodized though a .dli exists. Why is it part of Core (adding another dependency)?

Well, you have to give some oportunity to develop things too

Anyway, NetNode is being developed as we do not have any network browsing capabilities yet. It does not support UNICODE yet because

a) it was somewhat difficult to achieve
b) the chances that network nodes use UNICODE characters is not as high as for regular file names.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 21 Oct 2007 23:39:00 GMT
View Forum Message <> Reply to Message

Heh, that attempt backfired at me. Why doesn't it work that way in MinGW? This causes a certain compiler incompatibility. Unfortunately I don't know how to make the "good" MSVC version work. I could go back to previous include fashion that resembles BLITZ, but that would cause full rebuild on every build - not good. The problem is being unable to add CPPs to compiler on the fly.

...
Will you add my zlib (and png) plugins to the source tree? Without them, even the simpliest program won't work with my SCU, so the whole effort would be pointless to continue (Lib is good, but I'd like SCU too, especially since it almost works). If you will add them, I might add a scanner to pkggen to add necessary #undefs (though renaming ReplaceText would've been a lot easier).

Out of curiosity, am I right that ReplaceText wasn't supposed to become ReplaceTextA in RichEdit, and it just happened to work fine that way?

Regarding NetNode, it's good to see it developing, I just didn't like it being part of Core, requiring another library for even a trivial U++ app.

P.S. I can't properly preprocess RichEdit.h is MSVC, C:\upp710dev1\uppsrc\Core/Core.h(120) : fatal error C1189: #error : RTTI must be enabled !!! Find.cpp can be preprocessed. MinGW works too.

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sat, 27 Oct 2007 17:40:33 GMT
View Forum Message <> Reply to Message

sergei wrote on Sun, 21 October 2007 19:39
Will you add my zlib (and png) plugins to the source tree?

I am hesitating a lot. I see no problem with updating those to the newer version; but I do not like fixing 3rd party code too much...

(I believe there are often even legal issues involved...)

BTW, what about suggesting your patches to original authors? That, IMO, would be the correct approach.

Quote:
Out of curiosity, am I right that ReplaceText wasn't supposed to become ReplaceTextA in RichEdit, and it just happened to work fine that way?

The truth is we do not care. If any identifier * is replaced by *A everywhere, it makes little difference.

Quote:
Regarding NetNode, it's good to see it developing, I just didn't like it being part of Core, requiring another library for even a trivial U++ app.

Well, the library is a standard part of Win32 API. U++ links to many Win32 .dlls, one more or less does not make any difference.

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 28 Oct 2007 00:04:13 GMT
View Forum Message <> Reply to Message

There aren't any legal issues with BSD licences - you only have to mark them as modified (and open-source them too, but that will happen anyway).

I agree that as a general practice that would be bad, but I'm only talking about these few:

zlib - used in Core, thus simply has to be modified for SCU to work. Last update in 2005 - unlikely to change.
jpg - Often used. Last update in 1998 - won't change.
png, tif, bz2 - these are still rather active, so changes could be submitted to them. However, I doubt my modifications will be accepted - they are neither bugfixes nor functionality improvements, simply modifications for the plugin to make it work in another way.

I think that only z and png are used in all GUI apps. Thus they are a must. Rest aren't vital for trying out / working with U++. Modification to the (old) png version are simply adding casts. I'll try the latest version later. As I said, "big" ones like sqlite could be compiled in a separate lib.

In short - vital is zlib and png (hopefully new version will just work).

I still have no idea how to solve ReplaceText. That's really a technical challenge - how to make it work with headers and a single CPP file that has to include all of U++. Any help would be welcome.

Options:
1) The way I do it now - #include all packages headers into a single header (UppBase.h) included into program file. #include first all headers, then all sources of all packages into a single CPP file (UppBase.cpp). On MinGW that makes ReplaceText declared in UppBase.h become ReplaceTextA, and implementation in UppBase.cpp becomes ReplaceTextA too. On MSVC, ReplaceText in UppBase.h stays ReplaceText, yet implementation in UppBase.cpp becomes ReplaceTextA - error.
2) The way I did it before - #include everything into a single header (UppBase.h) - can be done per-package (pkg1hdr, pkg1src, pkg2hdr, pkg2src, ...). Then it will work on MinGW and MSVC (all will become ReplaceTextA). But then, every build, even a tiny change in program, will trigger full rebuild of U++ source (in first option that didn't happen since U++ source was in a never-changing UppBase.cpp).
3) Combine option 1 with option 2 - #include U++ per-package into UppBase.cpp, and only the headers into UppBase.h. ReplaceText will become ReplaceTextA in UppBase.cpp in both declaration and implementation, both MinGW and MSVC. However, in UppBase.h, ReplaceText will become ReplaceTextA only on MinGW, and will stay ReplaceText in MSVC - error.

Funny, but polluting the global namespace helps SCU

Hmm, this could also be hacked by #defining ReplaceText ReplaceTextA in RichEdit.h... Is it better than changing the function name?

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Sun, 28 Oct 2007 23:38:16 GMT
View Forum Message <> Reply to Message

OK, I've tried the new PNG. It works, without a single change to the original code. I'm even stricter than you were - you changed #include "zlib.h" to #include <plugin/z/zlib.h>, I instead added a dummy zlib.h to plugin dir that points to z plugin

Added these defines:

#define PNG_NO_MMX_CODE
#define PNG_USE_GLOBAL_ARRAYS

Didn't compile otherwise, and AFAIK MMX is disabled in currently used PNG too. I didn't really test the code, but since it's original unmodified and it compiles, it should work.

So, ReplaceText and zlib are the only showstoppers now. I added this to RichEdit.h:

#ifndef ReplaceText
#define ReplaceText ReplaceTextA
#endif

And used my zlib mod. Surprisingly, SCU EXE size is 2.0MB - same as Lib. I must've done something terribly wrong about the Lib, but I just can't figure out what. (OTOH, does anyone really care? 2MB for a GUI app isn't much, I've seen 20MB apps).

P.S. These few warnings in Core keep repeating many many times during Lib build, making it difficult to otice any other warnings, could they be fixed? (0 errors 0 warnings would be cool):

D:\Programming\Upp\Core/Topt.h(428) : warning C4311: 'type cast' : pointer truncation from 'const void *' to 'int'
D:\Programming\Upp\Core/Index.h(81) : warning C4244: 'argument' : conversion from 'intptr_t' to 'const int', possible loss of data
d:\programming\upp\core\Cbgen.h(805) : warning C4312: 'type cast' : conversion from 'int' to 'Upp::GateAction *' of greater size
D:\Programming\Upp\Core/Format.h(15) : warning C4311: 'type cast' : pointer truncation from 'const void *' to 'int'
D:\Programming\Upp\Core/Format.h(16) : warning C4311: 'type cast' : pointer truncation from 'const void *' to 'int'

File Attachments

1) png.zip, downloaded 446 times

Subject: Re: Building & using U++ without TheIDE

Posted by sergei on Thu, 01 Nov 2007 11:30:47 GMT
View Forum Message <> Reply to Message

Any feedback? Will this PNG be accepted to source tree?
And what regarding ReplaceText, zlib, warnings?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Sun, 04 Nov 2007 11:34:29 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 01 November 2007 07:30Any feedback? Will this PNG be accepted to source tree?

Apologies, too many things going on...

It is now accepted. Thanks, good job. (I only put zlib.h directly into /lib, as I believe #include "" would not be able to fetch it from plugin/png with GCC).

Quote:
And what regarding ReplaceText

I think I was clear enough. I do not think it is the fault of existing U++ code. I am not going to rename ReplaceText.

(Note that there are more possible name clashes in U++, all of them are resolved as long as the normal ("makefile") or U++ BLITZ is used.).

Quote:
, zlib, warnings?

Well, if only the exactly the same thing would be possible for zlib

Warnings: I will look into that soon (remind me if I do not).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 05 Nov 2007 08:06:38 GMT
View Forum Message <> Reply to Message

sergei wrote on Thu, 01 November 2007 07:30Any feedback? Will this PNG be accepted to source tree?

Update: Unfortunately, the updated PNG package does not compile *sometimes*.

Frankly, this is most puzzling, it worked on my desktop, but does not work on my notebook, with otherwise identical settings.

I have to investigate.... (meanwhile, I had to rollback it).

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Mon, 05 Nov 2007 10:10:46 GMT
View Forum Message <> Reply to Message

Update: So it seems nearly impossible to solve that without chnaging #include "zlib.h" in png.h.

If I put "fake" zlib.h as you did into plugin/png, it does not seem to work with mingw. With zlib.h in lib, it does not work with MSC...

So with this ugly thing, it is now back again...

Mirek

Subject: Re: Building & using U++ without TheIDE
Posted by sergei on Tue, 06 Nov 2007 13:07:41 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 05 November 2007 12:10Update: So it seems nearly impossible to solve that without changing #include "zlib.h" in png.h.

If I put "fake" zlib.h as you did into plugin/png, it does not seem to work with mingw. With zlib.h in lib, it does not work with MSC...

So with this ugly thing, it is now back again...

Mirek

How about 2 fake zlib.h? One in lib for MinGW, one in png for MSVC.

ReplaceText - OK. I can add the define in my SCU headers, that way uppsrc doesn't change.

zlib - there's absolutely no way to make K&R style fucntions work in C++. Though I might add another UppBase.c file, to compile C plugins there. That way it *might* work.

jpg - what about it, same as zlib? It's 1998, they aren't gonna change it

Edit: thinking of it, how does BLITZ handle C plugins? They aren't BLITZ-approved, so each file is compiled separately as a C-file?

Subject: Re: Building & using U++ without TheIDE
Posted by mirek on Tue, 06 Nov 2007 20:19:01 GMT
View Forum Message <> Reply to Message

sergei wrote on Tue, 06 November 2007 08:07luzr wrote on Mon, 05 November 2007 12:10Update: So it seems nearly impossible to solve that without changing #include "zlib.h" in png.h.

If I put "fake" zlib.h as you did into plugin/png, it does not seem to work with mingw. With zlib.h in lib, it does not work with MSC...

So with this ugly thing, it is now back again...

Mirek

How about 2 fake zlib.h? One in lib for MinGW, one in png for MSVC.

ReplaceText - OK. I can add the define in my SCU headers, that way uppsrc doesn't change.

zlib - there's absolutely no way to make K&R style fucntions work in C++. Though I might add another UppBase.c file, to compile C plugins there. That way it *might* work.

jpg - what about it, same as zlib? It's 1998, they aren't gonna change it

Edit: thinking of it, how does BLITZ handle C plugins? They aren't BLITZ-approved, so each file is compiled separately as a C-file?

.c are excluded from BLITZ.

Mirek
