Subject: theide/umk support for .lib/.a generation
Posted by mirek on Sun, 22 Nov 2020 16:45:26 GMT
View Forum Message <> Reply to Message

Well, people keep asking about U++ as library.

Thinking about it, I have figured out it should not be that hard to produce required libraries.

So there are now 2 new flags builders react to:

MAKE_LIB creates .lib file from the package, including all dependencies. This is meant e.g. for CtrlLib or Core.

MAKE_MLIB creates .lib file only from the main package. This is meant for e.g. plugins, like plugin/zip.

I have already tested, producing Core.lib and after a bit of fighting with Visual Studio 2019 I have managed to produce working console application (first one after 20 years - yes U++ development started in Visual Studio).

Now the hard part is to decide how to "parcel" the U++ into libraries. I am also afraid there will have to be like 12 versions of each library:

- USEMALLOC (not using U++ allocator)
- STDNEW (using standard new/delete, but U++ allocator where possible)
- with upp allocator

- Debug
- Release

- Linked with dynamic CRT
- Linked with static CRT

That is 3x2x2 = 12.... All those memories that made us create theide are coming back... :)

My current idea is that library generation should become a part of standard install. There would be a batch file / shell script that would generate it all. Maybe we can even make it part of theide or umk to automatize the process as much as possible.

I hope somebody will step in here and start parceling U++ and creating the script or maybe just "CreatLibs" package in uppbox or something... In the end, maybe the only input information needed is which packages are to be converted with MAKE_LIB and which are not to be converted at all. Rest should be made with MAKE_MLIB....

Subject: Re: theide/umk support for .lib/.a generation
Posted by Klugier on Sun, 22 Nov 2020 19:24:02 GMT

Hello Mirek,

That's great news!

Quote:
Now the hard part is to decide how to "parcel" the U++ into libraries. I am also afraid there will have to be like 12 versions of each library:

- USEMALLOC (not using U++ allocator)
- STDNEW (using standard new/delete, but U++ allocator where possible)
- with upp allocator

- Debug
- Release

- Linked with dynamic CRT
- Linked with static CRT

My point of view here is simple - the less variants we provide the better. In case of allocator I would choose only one option in context of .lib. Upp allocator should be exclusive for upp as a platform. So, I opt that we should use STDNEW only. Very good performance on library level, but user space is untouched. I think this is what most users want and do not create headache what option should be choose. In the very first version we could go with USEMALLOCK only.

In initial version we should only limit to RELEASE. In context of DEBUG works properly there is a need to distribute sources as well. In the first version the debugging of upp library should be only reserved to platform (TheIDE). I think initially there will not be many requests to add support for debug mode.

Providing CRT should be application responsibility. So, probably dynamic option is fine for us at least at the beginning. Mirek, do you see any objections difficulties here? The problem with static is that sometimes the CRT runtime is LGPL (Linux), so if we would like to go that path then we will need to make sure that we will do not link with libraries with that kind of license. I know that there is clang library that has BSD license. Anyway, this adds additional point of complexity, so initially I would focus only on dynamic version.

So, backing to math we could create only one variant instead of 12 and make whole solution simple :) This will also allow us to see how market will response and tune up in the future. KISS :)

In context of includes - will include works like this:

#include <Upp/CtrlLib/CtrlLib.h>

or do you plan to

#include <CtrlLib/CtrlLib.h>

I know that people might have objections for the second variant, so we should avoid it.

Klugier

Subject: Re: theide/umk support for .lib/.a generation
Posted by mirek on Sun, 22 Nov 2020 19:31:38 GMT
View Forum Message <> Reply to Message

Klugier wrote on Sun, 22 November 2020 20:24Hello Mirek,

That's great news!

Quote:
Now the hard part is to decide how to "parcel" the U++ into libraries. I am also afraid there will have to be like 12 versions of each library:

- USEMALLOC (not using U++ allocator)
- STDNEW (using standard new/delete, but U++ allocator where possible)
- with upp allocator

- Debug
- Release

- Linked with dynamic CRT
- Linked with static CRT

My point of view here is simple - the less variants we provide the better. In case of allocator I would choose only one option in context of .lib. Upp allocator should be exclusive for upp as a platform. So, I opt that we should use STDNEW only. Very good performance on library level, but user space is untouched. I think this is what most users want and do not create headache what option should be choose. In the very first version we could go with USEMALLOCK only.

In initial version we should only limit to RELEASE. In context of DEBUG works properly there is a need to distribute sources as well. In the first version the debugging of upp library should be only reserved to platform (TheIDE). I think initially there will not be many requests to add support for debug mode.

Providing CRT should be application responsibility. So, probably dynamic option is fine for us at least at the beginning. Mirek, do you see any objections difficulties here? The problem with static is that sometimes the CRT runtime is LGPL (Linux), so if we would like to go that path then we will need to make sure that we will do not link with libraries with that kind of license. I know that there is clang library that has BSD license. Anyway, this adds additional point of complexity, so initially I would focus only on dynamic version.

So, backing to math we could create only one variant instead of 12 and make whole solution simple :) This will also allow us to see how market will response and tune up in the future. KISS :)

It is actually 24. I forgot about x86 vs x64. And you cannot really avoid debug/release and dynamic/static, because Visual Studio will disallow linking in incompatible modes.

Quote:

#include <CtrlLib/CtrlLib.h>

I know that people might have objections for the second variant, so we should avoid it.

KISS. Let us try the simple path first...

Mirek

Subject: Re: theide/umk support for .lib/.a generation
Posted by Novo on Sun, 22 Nov 2020 20:50:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 22 November 2020 11:45
I hope somebody will step in here and start parceling U++ and creating the script or maybe just "CreatLibs" package in uppbox or something... In the end, maybe the only input information needed is which packages are to be converted with MAKE_LIB and which are not to be converted at all. Rest should be made with MAKE_MLIB....
I'm probably missing something, but isn't this stuff supposed to be a part of a Build Method?
You just need to auto-generate 24 different BM-files and everybody, who needs a library will just use an appropriate BM.
I'm doing this for sanitizers. This is just 6 more auto-generated BMs.

Subject: Re: theide/umk support for .lib/.a generation
Posted by Klugier on Sun, 22 Nov 2020 20:56:16 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Quote:It is actually 24. I forgot about x86 vs x64. And you cannot really avoid debug/release and dynamic/static, because Visual Studio will disallow linking in incompatible modes.

Make x86 pure optional. As far as I understand the libraries generation would be optional. As far as I understand on the instillation level we will give the user opportunity to generate standalone libraries depending on the needs. The question here is to specify reasonable defaults to shorten this process. In this case I agree with debug/release and static/dynamic CRT. So, in this case we should have 4 versions.

I see that on conan central there is similar matrix for libraries. So, for example for OpenSSL there is 24 variants for Windows and 52 for Linux.

Quote:KISS. Let us try the simple path first...
You are right, simpler approach is better. Many libraries do not use prefix for includes - we should be fine.

Klugier

Subject: Re: theide/umk support for .lib/.a generation
Posted by Novo on Sun, 22 Nov 2020 21:10:39 GMT
View Forum Message <> Reply to Message

24 variants for Windows include one compiler, but two versions of it (compiler.version=16 and 15).
Clang is missing. :roll:

Subject: Re: theide/umk support for .lib/.a generation
Posted by Novo on Sun, 22 Nov 2020 22:33:49 GMT
View Forum Message <> Reply to Message

BTW, there is an AddressSanitizer (ASan) for Windows with MSVC.
This brings number of configurations to 48. :roll:

Subject: Re: theide/umk support for .lib/.a generation
Posted by mirek on Sun, 22 Nov 2020 23:13:09 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 22 November 2020 21:50mirek wrote on Sun, 22 November 2020 11:45
I hope somebody will step in here and start parceling U++ and creating the script or maybe just "CreatLibs" package in uppbox or something... In the end, maybe the only input information needed is which packages are to be converted with MAKE_LIB and which are not to be converted at all. Rest should be made with MAKE_MLIB....
I'm probably missing something, but isn't this stuff supposed to be a part of a Build Method?
You just need to auto-generate 24 different BM-files and everybody, who needs a library will just use an appropriate BM.
I'm doing this for sanitizers. This is just 6 more auto-generated BMs.

Everybody who "needs" library will expect us to give him .lib files...

And it is not really 6 .bm, just a combination of umk flags and 2 .bms..

Anyway, all that is probably quite simple. The real issue is to decide which .lib files are needed.

Mirek

Subject: Re: theide/umk support for .lib/.a generation
Posted by Klugier on Mon, 23 Nov 2020 00:30:44 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Let's target packages that do not have external dependencies. The only require dependency I see is OpenSSL for core. So in the first version, we shouldn't support MySQL, PostgreSQL etc. sqlite should be sufficient.

Of course, we should leave all IDE and main packages of uppsrc. IconDes, HexView, CppBase, Common (why this package is in uppsrc anyway?), DocTypes (legacy) are also do not need. Some plugins also like plugin/FT_fontsys, plugin/DroidFonts, plugin/zstd_legacy, plugin/astyle (this should be replaced with clang format) or plugin/ndisam.

Not sure about VirtualGUI/SDL2GL seems that this requires SDL. Geom and it's packages are absolute.

art/bluebar (seems absolute even TheIDE doesn't use it anymore)

Klugier

Subject: Re: theide/umk support for .lib/.a generation
Posted by Novo on Mon, 23 Nov 2020 02:36:50 GMT
View Forum Message <> Reply to Message

Subject: Re: theide/umk support for .lib/.a generation
Posted by mirek on Mon, 23 Nov 2020 08:28:50 GMT
View Forum Message <> Reply to Message

Klugier wrote on Mon, 23 November 2020 01:30Hello Mirek,

Let's target packages that do not have external dependencies.

Thats only Core.

Quote:
 The only require dependency I see is OpenSSL for core. So in the first version, we shouldn't support MySQL, PostgreSQL etc. sqlite should be sufficient.

I disagree. Providing such packages is trivial. But it will need a bit of documentation... (you must add these dependencies manually).

Doing things without theide/umk is so complicated... :(

Mirek

Subject: Re: theide/umk support for .lib/.a generation
Posted by Klugier on Mon, 23 Nov 2020 22:10:03 GMT
View Forum Message <> Reply to Message

Hello Mirek,

World outside U++ platform is so complicated :) Glad we have TheIDE/umk ;) If MySQL and PostgreSQL are not problematic I ma fine with exporting them.

So seems that we have potential packages - would be nice to write them down and after that make the review.

Klugier
