Subject: Considering different approach to Win32 release
Posted by mirek on Sun, 20 Sep 2015 17:10:27 GMT
View Forum Message <> Reply to Message

Well, current installer is deeply "illegal" - using .exe to install things was OK 10 years ago, but today .msi is required.

Anyway, before doing that, I have got an idea that perhaps we could try something different. What about just creating something like "portable" U++, simple .zip which, when unpacked, would run TheIDE from any folder, autoconfiguring everything in the process.

mingw would probably now be part of release, otherwise autoconfig would try to setup MSC15 build method. Nothing would be written to the registry, to unistall, all you need to do is to delete the folder.

Is it a good idea?

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by dolik.rce on Sun, 20 Sep 2015 18:31:37 GMT
View Forum Message <> Reply to Message

Hi Mirek,

mirek wrote on Sun, 20 September 2015 19:10What about just creating something like "portable" U++, simple .zip which, when unpacked, would run TheIDE from any folder, autoconfiguring everything in the process.

Do you mean like this? :d That one actually ran on Linux too, but windows only would be much simpler.

mirek wrote on Sun, 20 September 2015 19:10Is it a good idea?
Definitely a good idea. I found it quite useful in past, to have a usb thumb drive that could've been used to run TheIDE on any computer. Also, it helps a lot when people are forced to use windows computers without administration rights, e.g. at work or at school :)

Best regards,
Honza

Subject: Re: Considering different approach to Win32 release
Posted by Didier on Sun, 20 Sep 2015 19:57:34 GMT
View Forum Message <> Reply to Message

YES !! portable is the way to go (or at least UPP must have this option).

If you only propose .msi install, this will not work when you don't have administrator wrights which happens more and more in work environnement.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Sun, 20 Sep 2015 21:36:18 GMT
View Forum Message <> Reply to Message

Ah, maybe I should clarify the question...

Is it OK not to have "installer" version? :) Would save a lot of time...

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by dolik.rce on Mon, 21 Sep 2015 05:27:55 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 20 September 2015 23:36Ah, maybe I should clarify the question...

Is it OK not to have "installer" version? :) Would save a lot of time...

Mirek

IMHO the only "added value" of msi installer is the uninstall function. If the new version keeps all thing in single directory, then uninstall is simply one delete operation. The only problem might be with desktop icons and start menu items - if you plan to have them at all. In that case, we can add simple batch file to install/uninstall those. Or we can just leave that to the user, after all U++'s target audience should be capable of handling few icons :)

So, yes, I think we can live without the installer.

Honza

Subject: Re: Considering different approach to Win32 release
Posted by timsky on Mon, 21 Sep 2015 13:58:13 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 20 September 2015 17:10Well, current installer is deeply "illegal" - using .exe to install things was OK 10 years ago, but today .msi is required.

Anyway, before doing that, I have got an idea that perhaps we could try something different. What about just creating something like "portable" U++, simple .zip which, when unpacked, would run TheIDE from any folder, autoconfiguring everything in the process.

mingw would probably now be part of release, otherwise autoconfig would try to setup MSC15 build method. Nothing would be written to the registry, to unistall, all you need to do is to delete the folder.

Is it a good idea?

Mirek

Current installer is great. Portable version would be even better.
U++ is a tool for developer, not for housewife :d

Subject: Re: Considering different approach to Win32 release
Posted by koldo on Mon, 21 Sep 2015 14:14:55 GMT
View Forum Message <> Reply to Message

Yes, it is a good idea to have a portable system that works without administrator permissions, with mingw inside but at the same time that tries to use existing compilers.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Wed, 21 Oct 2015 18:30:25 GMT
View Forum Message <> Reply to Message

Win32 nightly builds now changed according to this thread (will 'announce' again when download page is fixed accordingly).

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Mon, 26 Oct 2015 08:51:08 GMT
View Forum Message <> Reply to Message

Hi Mirek,

I'm having trouble getting started with this new portable version:

First, unpacking the 7z file takes quite a long time because of mingw is there. (How about a non-mingw variant of Upp? I think I will never use mingw anyway.)

Second, U++ can not find my installed MSC9 and MSC10 SDK compilers (32 nor 64 bit variants). Selecting Setup>Instant setup.. does not help.

Third, I tried compiling UWord with both MINGW32 and MINGW64. Both work in DEBUG mode, but when compiled in Optimal mode, both UWord executables crash on start.

Can you help with these? Edit: I mean especially with getting MSC9 and MSC10 back to work.

Dropping MINGW would be a nice option extra...

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Mon, 26 Oct 2015 11:23:12 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 26 October 2015 09:51Hi Mirek,

I'm having trouble getting started with this new portable version:

First, unpacking the 7z file takes quite a long time because of mingw is there. (How about a non-mingw variant of Upp? I think I will never use mingw anyway.)

Second, U++ can not find my installed MSC9 and MSC10 SDK compilers (32 nor 64 bit variants). Selecting Setup>Instant setup.. does not help.

Third, I tried compiling UWord with both MINGW32 and MINGW64. Both work in DEBUG mode, but when compiled in Optimal mode, both UWord executables crash on start.

Can you help with these? Edit: I mean especially with getting MSC9 and MSC10 back to work. Dropping MINGW would be a nice option extra...

Best regards,

Tom

Thanks for testing and comments.

Instant setup indeed ignores pre MSC15 compilers. The primary reason is simple: VS 2015 is the first real Win32 C++11 compiler. As we would eventually like to move to C++11, I have decided to force this upon poor users a bit :)

However, current U++ still supports MSC9 etc, including autosetup: It is hidden behind "Verbose" flag. Activate Verbose you will get legacy autosetup in Setup menu.

Thanks for archive size comment. I seemed OK to me, but this is perhaps the sign that more work is needed there... I guess I can significantly reduce the size, there is a lot of unused stuff there.

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Mon, 26 Oct 2015 14:01:59 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks for the tip! Helped a lot.

Even though the MSC9/MSC10 Build method does not have 'Use BLITZ' selected for release mode, I see that there seems to be a considerable amount of BLITZ used in release mode too. While radically boosting compilation times on MSC10, MSC10x64 and MSC9x64, it also results in an internal compiler error on 32 bit MSC9 somewhere in RichText/TxtOp.cpp. The code seems same as before, but disabling BLITZ for RichText in Package organizer allows successful compilation.

Shouldn't the 'Use BLITZ' selection for the build method be obeyed as the default?

Thanks,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Mon, 26 Oct 2015 14:26:44 GMT
View Forum Message <> Reply to Message

Hi,

One more thing: It appears the custom build steps (mine are of post-link type) are running twice (and seemingly simultaneously) -- causing failure in this case.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Mon, 26 Oct 2015 14:34:11 GMT
View Forum Message <> Reply to Message

Hi,

mirek wrote on Mon, 26 October 2015 13:23
Instant setup indeed ignores pre MSC15 compilers. The primary reason is simple: VS 2015 is the first real Win32 C++11 compiler. As we would eventually like to move to C++11, I have decided to force this upon poor users a bit :)

However, current U++ still supports MSC9 etc, including autosetup: It is hidden behind "Verbose" flag. Activate Verbose you will get legacy autosetup in Setup menu.

Thanks for archive size comment. I seemed OK to me, but this is perhaps the sign that more work is needed there... I guess I can significantly reduce the size, there is a lot of unused stuff there.

Mirek

While on the subject of 'force this upon poor users a bit' could you take a look at Bazaar/Protect from Max? I really need this type of solution and MSC9 is the last compiler to support the current implementation. (Right?) If there is an alternative solution for this task that works on MSC15 compilers, I'd be just happy to upgrade, but meanwhile I will need to stick with the old MSC9.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Mon, 26 Oct 2015 17:58:09 GMT

View Forum Message <> Reply to Message

Tom1 wrote on Mon, 26 October 2015 15:34Hi,

mirek wrote on Mon, 26 October 2015 13:23
Instant setup indeed ignores pre MSC15 compilers. The primary reason is simple: VS 2015 is the first real Win32 C++11 compiler. As we would eventually like to move to C++11, I have decided to force this upon poor users a bit :)

However, current U++ still supports MSC9 etc, including autosetup: It is hidden behind "Verbose" flag. Activate Verbose you will get legacy autosetup in Setup menu.

Thanks for archive size comment. I seemed OK to me, but this is perhaps the sign that more work is needed there... I guess I can significantly reduce the size, there is a lot of unused stuff there.

Mirek

While on the subject of 'force this upon poor users a bit' could you take a look at Bazaar/Protect from Max? I really need this type of solution and MSC9 is the last compiler to support the current implementation. (Right?) If there is an alternative solution for this task that works on MSC15 compilers, I'd be just happy to upgrade, but meanwhile I will need to stick with the old MSC9.

Best regards,

Tom

What are symptoms? (I have compiled packages fine, after some minor C++11 fixing).

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Mon, 26 Oct 2015 17:59:40 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Mon, 26 October 2015 15:01Hi Mirek,

Thanks for the tip! Helped a lot.

Even though the MSC9/MSC10 Build method does not have 'Use BLITZ' selected for release mode, I see that there seems to be a considerable amount of BLITZ used in release mode too. While radically boosting compilation times on MSC10, MSC10x64 and MSC9x64, it also results in an internal compiler error on 32 bit MSC9 somewhere in RichText/TxtOp.cpp. The code seems same as before, but disabling BLITZ for RichText in Package organizer allows successful compilation.

Shouldn't the 'Use BLITZ' selection for the build method be obeyed as the default?

Thanks,

Tom

This is really strange. Can you post a copy of console log?

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Tue, 27 Oct 2015 08:01:35 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 26 October 2015 19:58Tom1 wrote on Mon, 26 October 2015 15:34Hi,

mirek wrote on Mon, 26 October 2015 13:23
Instant setup indeed ignores pre MSC15 compilers. The primary reason is simple: VS 2015 is the first real Win32 C++11 compiler. As we would eventually like to move to C++11, I have decided to force this upon poor users a bit :)

However, current U++ still supports MSC9 etc, including autosetup: It is hidden behind "Verbose" flag. Activate Verbose you will get legacy autosetup in Setup menu.

Thanks for archive size comment. I seemed OK to me, but this is perhaps the sign that more work is needed there... I guess I can significantly reduce the size, there is a lot of unused stuff there.

Mirek

While on the subject of 'force this upon poor users a bit' could you take a look at Bazaar/Protect from Max? I really need this type of solution and MSC9 is the last compiler to support the current implementation. (Right?) If there is an alternative solution for this task that works on MSC15 compilers, I'd be just happy to upgrade, but meanwhile I will need to stick with the old MSC9.

Best regards,

Tom

What are symptoms? (I have compiled packages fine, after some minor C++11 fixing).

Mirek

Here's the way to go:

1. In examples-bazaar nest build Bazaar/ProtectEncrypt (no flags here, just build mode MSCx Optimal)
2. In examples-bazaar open main package Bazaar/ProtectTest and edit its custom build step for WIN32 to use ProtectEncrypt.exe you just built instead of the default.
3. Then build Bazaar/ProtectTest (flags GUI PROTECT, build mode MSCx Optimal).

When I did this using MSC9, it worked all the way and the resulting ProtectTest.exe worked fine. The same using MSC10 resulted in crashing ProtectTest.exe right on start.

Compiling for x64 sadly fails with:

C:\Users\tom\Desktop\upp-win32-9081\upp\bazaar\ProtectTest\main.cpp(19) : error C4235: nonstandard extension used : '__asm' keyword not supported on this architecture

I just downloaded and installed Visual studio community 2015, but TheIDE does not seem to be able to detect the compiler. I must have missed something here... Should I have the Professional version of VS? Anyway, I can't try if this works any better with Protect.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Tue, 27 Oct 2015 08:20:21 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 26 October 2015 19:59Tom1 wrote on Mon, 26 October 2015 15:01Hi Mirek,

Thanks for the tip! Helped a lot.

Even though the MSC9/MSC10 Build method does not have 'Use BLITZ' selected for release mode, I see that there seems to be a considerable amount of BLITZ used in release mode too. While radically boosting compilation times on MSC10, MSC10x64 and MSC9x64, it also results in an internal compiler error on 32 bit MSC9 somewhere in RichText/TxtOp.cpp. The code seems same as before, but disabling BLITZ for RichText in Package organizer allows successful compilation.

Shouldn't the 'Use BLITZ' selection for the build method be obeyed as the default?

Thanks,

Tom

This is really strange. Can you post a copy of console log?

Mirek

No need for log, I found the cause: While in 'Build methods' Use BLITZ is not active by default for MSC9 or MSC10, in 'Output mode' dialog it is active by default for release mode. By switching it of in 'Output mode' cures the symptoms.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Tue, 27 Oct 2015 09:29:58 GMT
View Forum Message <> Reply to Message

So I tried updating to the new scheme.

Installation took forever from the 7z file and eats up 1 GB.

Automatic setup took again a while and did not find any of my MSC versions.

Running any GUI program and closing it still leaves TheIDE thinking that the application is running and I need to force close it.

This new system does not look production ready at the moment.

Subject: Re: Considering different approach to Win32 release

Posted by cbpporter on Tue, 27 Oct 2015 09:39:48 GMT
View Forum Message <> Reply to Message

Additionally, things like nullptr and

VectorMap<String, Vector<String>> classpath;
are not recognized out of the box when using MinGW.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Tue, 27 Oct 2015 09:45:33 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 27 October 2015 10:29So I tried updating to the new scheme.

Thanks!

Quote:
Installation took forever from the 7z file and eats up 1 GB.

Well, mingw toolchains are THAT big in original form. Frankly, I have not expected that this to be a problem... BUT OK, I think I can half it.

Quote:
Automatic setup took again a while and did not find any of my MSC versions.

As explained above, it only picks VS2015, for a reason...

Quote:
Running any GUI program and closing it still leaves TheIDE thinking that the application is running and I need to force close it.

You mean in mingw debugger? That was a bug, should be now fixed.

Quote:
This new system does not look production ready at the moment.

Working on it...

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Tue, 27 Oct 2015 09:55:38 GMT

View Forum Message <> Reply to Message

Thank for the answers Mirek!

I tried with upp-win32-9087.7z.

After this step is done, maybe it is time to consider a more modern update mechanic? Big downloads and installs take away the old advantage of U++ of getting it up and running and updated using nightly builds.

Maybe TheIDE, after initial install, could check once in while to see if new mingw versions are available? Or new upp version? I doubt MINGW changes that often, so an option to only update uppsrc would be great.

Since you said that using exe is not that modern, neither is .msc. Modern things update themselves.

I missed the VS2015 explanation. Does that mean that U++ now needs C++11? Our official compiler in 2010 :).

And what about MINGW not recognizing nullptr and other things? I think that the switch to MINGW will result in a lot of compatibility patches in user's code since MSC does take its fair share of liberties.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Tue, 27 Oct 2015 09:59:01 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 27 October 2015 10:39Additionally, things like nullptr and

VectorMap<String, Vector<String>> classpath;
are not recognized out of the box when using MinGW.

Thanks, fixed.

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Tue, 27 Oct 2015 10:00:31 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 27 October 2015 09:20
No need for log, I found the cause: While in 'Build methods' Use BLITZ is not active by default for MSC9 or MSC10, in 'Output mode' dialog it is active by default for release mode. By switching it of in 'Output mode' cures the symptoms.

Best regards,

Tom

Well, but that is weird too, it should not be active... (unless you have activated it accidentally).

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Tue, 27 Oct 2015 10:26:38 GMT
View Forum Message <> Reply to Message

Last version in official use is 8230.

I'm working on merging things so we can switch to a more mainstream version.

Need to figure out what to do with the CodeEditor fork...

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Tue, 27 Oct 2015 11:25:11 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 27 October 2015 12:26Last version in official use is 8230.

I'm working on merging things so we can switch to a more mainstream version.

Need to figure out what to do with the CodeEditor fork...
Actually, it is 8627. That was our last official version which worked just fine with MSC.

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Tue, 27 Oct 2015 11:48:50 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 27 October 2015 12:00Tom1 wrote on Tue, 27 October 2015 09:20
No need for log, I found the cause: While in 'Build methods' Use BLITZ is not active by default for MSC9 or MSC10, in 'Output mode' dialog it is active by default for release mode. By switching it of in 'Output mode' cures the symptoms.

Best regards,

Tom

Well, but that is weird too, it should not be active... (unless you have activated it accidentally).

I reinstalled 9081 to be sure and in 'Output mode' dialog BLITZ is enabled for both debug and release modes out of the box. (Of course I had to create build methods for my MSC9 and MSC10 by using the Be verbose + Automatic setup, but that's really all.)

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Wed, 28 Oct 2015 06:57:27 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Tue, 27 October 2015 10:55Thank for the answers Mirek!

I tried with upp-win32-9087.7z.

After this step is done, maybe it is time to consider a more modern update mechanic? Big downloads and installs take away the old advantage of U++ of getting it up and running and updated using nightly builds.

Agreed, but perhaps not for this release. (It will take significant time to develop).

Quote:
I missed the VS2015 explanation. Does that mean that U++ now needs C++11? Our official compiler in 2010 :).

No, not yet. That said, we would like like to move there eventually.

Anyway, it is true that with C++11, transfer semantics issues were changed, so some minimal changes to the code, backward compatible with C++98, are needed. It took about 5 minutes to fix theide, so no big deal.

Quote:
And what about MINGW not recognizing nullptr and other things? I think that the switch to MINGW will result in a lot of compatibility patches in user's code since MSC does take its fair share of liberties.

(fixed)

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Wed, 28 Oct 2015 06:58:47 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 27 October 2015 12:48mirek wrote on Tue, 27 October 2015 12:00Tom1

wrote on Tue, 27 October 2015 09:20
No need for log, I found the cause: While in 'Build methods' Use BLITZ is not active by default for MSC9 or MSC10, in 'Output mode' dialog it is active by default for release mode. By switching it of in 'Output mode' cures the symptoms.

Best regards,

Tom

Well, but that is weird too, it should not be active... (unless you have activated it accidentally).

I reinstalled 9081 to be sure and in 'Output mode' dialog BLITZ is enabled for both debug and release modes out of the box. (Of course I had to create build methods for my MSC9 and MSC10 by using the Be verbose + Automatic setup, but that's really all.)

Best regards,

Tom

Thanks, I believe I have figured it out (mingw had it active for release, and so it got there on first run).

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Wed, 28 Oct 2015 07:06:24 GMT
View Forum Message <> Reply to Message

Hi,

Thanks Mirek. Now BLITZ is not active any more by default. After using the new menu for legacy compilers, all MSC9, MSC9x64, MSC10 and MSC10x64 variants work without BLITZ right from the start.

A side note for MINGW: When compiling the UWord example, both 32 and 64 bit variants produce executables that crash on start. (Running on Windows 8.1 Professional x64.) Don't know what's the actual problem but it seems to be cured after switching all optimizations off (flag O0).

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release

Posted by cbpporter on Wed, 28 Oct 2015 09:53:07 GMT
View Forum Message <> Reply to Message

OK, the C++11x issues are gone and now I'm left with 144 other errors. Most of them all in my code, but there is also an AssertMoveable issue:

```
C:\dev\upp/uppsrc/Core/Topt.h:212:70: error: specialization of 'template<class T> void Upp::AssertMoveable(T*)' in different namespace [-fpermissive]
 #define NTL_MOVEABLE(T) template<> inline void AssertMoveable<T>(T *) {}
```

My offending code is:

```
class Class;

class Source {
public:
 String Path;
 String data;
 ZParser parser;
 int index;
 bool IsScaned;

 Vector<Class*> Classes;
};

NTL_MOVEABLE(Source);
```

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Wed, 28 Oct 2015 10:10:31 GMT
View Forum Message <> Reply to Message

Yeah, I am left with 56 C++11x related errors (move, copy constructors and implicitly deleted constructors for whatever reasons) that I have never encountered before.

This is the price you pay for using MSC 2010 :).

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Wed, 28 Oct 2015 10:32:00 GMT
View Forum Message <> Reply to Message

When using something like:

AST ast = AST(ass);

I get an error related to an implicitly deleted copy constructor AST::AST(const AST&). It is probably deleted since I have a constructor that takes parameters and no default one. Up to this moment I have lived with the impression that such a statement, since it is a variable declaration, both officially and unofficially optimizes away and does not involve the use of a copy constructor.

Subject: Re: Considering different approach to Win32 release
Posted by Mindtraveller on Wed, 28 Oct 2015 15:39:06 GMT
View Forum Message <> Reply to Message

Actually there's simple way to correct it. You just need to 'say' what you want explicitly:
AST ast = pick (ass);

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Wed, 28 Oct 2015 15:48:49 GMT
View Forum Message <> Reply to Message

Mindtraveller wrote on Wed, 28 October 2015 17:39Actually there's simple way to correct it. You just need to 'say' what you want explicitly:
AST ast = pick (ass);
It is a constructor, not a copy constructor. ast and ass do not have the same types.

The solution is:
AST ast(ass);

I write it as:

AST ast = AST(ass);

for uniformity.

I am expecting both to generate the same code and not involve a copy constructor, ass in AST(ass) being built and the the compiler complaining that there is no copy constructor.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Wed, 28 Oct 2015 18:20:47 GMT
View Forum Message <> Reply to Message

use

class Source : Moveable<Source>

That said, sometimes I am thinking that perhaps that "Moveable" thing is unnecessary... (It only serves to remind the programmer that there are rules about things put into Vector).

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Thu, 29 Oct 2015 08:30:38 GMT

View Forum Message <> Reply to Message

mirek wrote on Wed, 28 October 2015 20:20use

```
class Source : Moveable<Source>
```

That said, sometimes I am thinking that perhaps that "Moveable" thing is unnecessary... (It only serves to remind the programmer that there are rules about things put into Vector).

Thanks! So NTL_MOVEABLE no longer works under this setup?

It is weird tracking things down with all the difference s between compilers and version.

I have one final issue which I know how to solve:

```
def.Blocks.Pop();
```

Now, first of all, I shouldn't be using Pop, since I do not care about the value. I should be using Drop.

```
class Block: Moveable<Block> {
public:
 VectorMap<String, Variable> Vars;
 int Temps;
};
```

I get the error "use of deleted Block::Bock(const block&)" / "is implicitly deleted because the default definition would be ill-formatted".

Supposing I wanted to use Pop or any method with copy construction, which is the proper way to fix this to maximize compatibility with MSC2010 with its 1% C++11x compatibility, MSC15 and MINGW? Should I just write a copy constructor?

Anyway, main project now compiles under MINGW and all the changes to Core (only TimeStop precision issues) are now moved to my code base. I have no real interest in MINGW, but for the foreseeable future I shall be using MINGW just for testing purposes so I can report issues.

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Thu, 29 Oct 2015 08:42:34 GMT

View Forum Message <> Reply to Message

Hi Mirek,

I have now tested 9105 on Windows 8.1 Professional x64 with the following results:

1. From my point of view MSC9 and MSC10 both work perfectly OK now.

2. It still can't find the installed Visual studio 2015 community edition. Is this expected?

3. MINGW Debug compiles UWord correctly and the resulting executable works.

4. MINGW Size compiles UWord correctly and the resulting executable works.

5. MINGW Optimal compiles UWord but the resulting executable crashes. It requires dropping the speed optimization flag from O3 to O2 in order to make it work.

6. MINGW Speed compiles UWord but the resulting executable crashes. It requires dropping the speed optimization flag from O3 to O2 in order to make it work.

7. MINGWx64 Debug compiles UWord correctly and the resulting executable works.

8. MINGWx64 Size compiles UWord but the resulting executable crashes. (Switching from Os to e.g. O0 in size optimization flags fixes the executable.)

9. MINGWx64 Optimal compiles UWord but the resulting executable crashes. (Switching from Os to O0, O1, O2 or O3 in size optimization flags fixes the executable.)

10. MINGWx64 Speed compiles UWord correctly and the resulting executable works.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 12:13:25 GMT

View Forum Message <> Reply to Message

cbpporter wrote on Thu, 29 October 2015 09:30mirek wrote on Wed, 28 October 2015 20:20use class Source : Moveable<Source>

That said, sometimes I am thinking that perhaps that "Moveable" thing is unnecessary... (It only serves to remind the programmer that there are rules about things put into Vector).

Thanks! So NTL_MOVEABLE no longer works under this setup?

It is weird tracking things down with all the difference s between compilers and version.

I have one final issue which I know how to solve:

def.Blocks.Pop();

Now, first of all, I shouldn't be using Pop, since I do not care about the value. I should be using Drop.

```
class Block: Moveable<Block> {
public:
 VectorMap<String, Variable> Vars;
 int Temps;
};
```

I get the error "use of deleted Block::Bock(const block&)" / "is implicitly deleted because the default definition would be ill-formatted".

This another small difference. See http://www.ultimatepp.org/srcdoc$Core$pick_$en-us.html, Composition.

You need to add

```
  rval_default(Block);
```

to the class.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 12:22:24 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 29 October 2015 09:42Hi Mirek, I have now tested 9105 on Windows 8.1 Professional x64 with the following results:

Thanks for testing! Definitely helpful.

Quote:
2. It still can't find the installed Visual studio 2015 community edition. Is this expected?

That is definitely unexpected and "wrong".

Well, instead of long and detailed request for investigation, could you please just look into ide/InstantSetup.cpp and put on line 180

```
  vc = df.ScanForDir("/vc", "", "bin/link.exe;bin/cl.exe;bin/mspdb140.dll", "bin/1033");
  bin = df.ScanForDir(x64 ? "bin/x64" : "bin/x86", "/windows kits/", "makecat.exe;accevent.exe", "");
  inc = df.ScanForDir("", "/windows kits/", "um/adhoc.h", "um;ucrt;shared");
  lib = df.ScanForDir("", "/windows kits/", "um/x86/kernel32.lib", "um;ucrt");

DUMP(vc); DUMP(bin); DUMP(inc); DUMP(lib); // <<==== THIS

  if(vc.GetCount() * bin.GetCount() * inc.GetCount() * lib.GetCount()) {
   bins.At(0) = vc + (x64 ? "/bin/amd64" : "/bin");
   bins.At(1) = bin;
   String& sslbin = bins.At(2);
```

and then run "Instant setup.." and look into .log?

(Explanation: Instead looking into registry (which proved error-prone with MSC12), theide now scans Program files directories and attempts to find appropriate directories.)

Quote:
3. MINGW Debug compiles UWord correctly and the resulting executable works.

4. MINGW Size compiles UWord correctly and the resulting executable works.

5. MINGW Optimal compiles UWord but the resulting executable crashes. It requires dropping the speed optimization flag from O3 to O2 in order to make it work.

6. MINGW Speed compiles UWord but the resulting executable crashes. It requires dropping the speed optimization flag from O3 to O2 in order to make it work.

7. MINGWx64 Debug compiles UWord correctly and the resulting executable works.

8. MINGWx64 Size compiles UWord but the resulting executable crashes. (Switching from Os to e.g. O0 in size optimization flags fixes the executable.)

9. MINGWx64 Optimal compiles UWord but the resulting executable crashes. (Switching from Os to O0, O1, O2 or O3 in size optimization flags fixes the executable.)

10. MINGWx64 Speed compiles UWord correctly and the resulting executable works.

Best regards,

Tom

Unfortunately, investigation revealed apparent bug in GCC optimizer (you can compile in release mode with full debug info and run in debug - bug in assembly is quite apparent).

I will try to downgrade gcc version. If that does not help (it should, in Linux it works fine), I will change -O levels...

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Thu, 29 Oct 2015 12:43:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 29 October 2015 14:13cbpporter wrote on Thu, 29 October 2015 09:30mirek wrote on Wed, 28 October 2015 20:20use

class Source : Moveable<Source>

That said, sometimes I am thinking that perhaps that "Moveable" thing is unnecessary... (It only serves to remind the programmer that there are rules about things put into Vector).

Thanks! So NTL_MOVEABLE no longer works under this setup?

It is weird tracking things down with all the difference s between compilers and version.

I have one final issue which I know how to solve:

def.Blocks.Pop();

Now, first of all, I shouldn't be using Pop, since I do not care about the value. I should be using Drop.

```
class Block: Moveable<Block> {
public:
 VectorMap<String, Variable> Vars;
 int Temps;
};
```

I get the error "use of deleted Block::Bock(const block&)" / "is implicitly deleted because the default definition would be ill-formatted".

This another small difference. See http://www.ultimatepp.org/srcdoc$Core$pick_$en-us.html, Composition.

You need to add

```
   rval_default(Block);
```

to the class.

Thank you!

I also needed to add a default constructor, but it works now.

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Thu, 29 Oct 2015 13:17:14 GMT

View Forum Message <> Reply to Message

Hi,

Here's a very tiny portion of the log (mostly filled with "&area ="):

```
&area = 0x025dc4e4
&area = 0x025dc418
&area = 0x025dc36c
&area = 0x025dc2a0
&area = 0x025dc534
vc = c:/program files (x86)/microsoft visual studio 14.0/vc
bin = c:/program files (x86)/windows kits/8.1/bin/x86
inc =
lib =
vc = c:/program files (x86)/microsoft visual studio 14.0/vc
bin = c:/program files (x86)/windows kits/8.1/bin/x64
inc =
```

lib =
&area = 0x025dc838
&area = 0x025dc800
&area = 0x025dc7f0
&area = 0x025dc7b4
&area = 0x025dd0a4

Hope this helps...

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 13:22:55 GMT
View Forum Message <> Reply to Message

Do you have

C:\Program Files (x86)\Windows Kits/10

folder?

(If not, then it looks like you do not have current SDK installed).

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Thu, 29 Oct 2015 13:33:41 GMT
View Forum Message <> Reply to Message

I have both C:\Program Files (x86)\Windows Kits\10 (about 490 MB) and C:\Program Files (x86)\Windows Kits\8.1 (about 532 MB).

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 14:04:12 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 29 October 2015 14:33I have both C:\Program Files (x86)\Windows Kits\10 (about 490 MB) and C:\Program Files (x86)\Windows Kits\8.1 (about 532 MB).

Tom

Hm, would it be somehow possible to list all dirs/files in ..Kits\10, recursively with subdirs, and post here as attachment? (If not, I will ask about specific files, but that will take longer...)

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Thu, 29 Oct 2015 14:26:58 GMT
View Forum Message <> Reply to Message

This code helped me to correctly find the SDK, but I confess that last time I tried it was on the pre-release VS. Then I saw the single .lib file and decided not to support it :).

```
void TestLib() {
  if (create) {
   if (FindFile(sdk + "\\lib\\*.lib"))
    sdklib = "\\lib";
   if (FindFile(sdk + "\\lib\\x86\\*.lib"))
    sdklib = "\\lib\\x86";
   else if (FindFile(sdk + "\\lib\\win8\\um\\x86\\*.lib"))
    sdklib = "\\lib\\win8\\um\\x86";
   else if (FindFile(sdk + "\\lib\\winv6.3\\um\\x86"))
    sdklib = "\\lib\\winv6.3\\um\\x86";
  }
  if (create64) {
   if (FindFile(sdk + "\\lib\\*.lib"))
    sdklib64 = "\\lib";
   else if (FindFile(sdk + "\\lib\\x64\\*.lib"))
    sdklib64 = "\\lib\\x64";
   else if (FindFile(sdk + "\\lib\\win8\\um\\x64\\*.lib"))
    sdklib64 = "\\lib\\win8\\um\\x64";
   else if (FindFile(sdk + "\\lib\\winv6.3\\um\\x64"))
    sdklib64 = "\\lib\\winv6.3\\um\\x64";
  }
 }
```

If the lib paths are empty even though apparently the SDK has been found, I ignore it.

Subject: Re: Considering different approach to Win32 release

Posted by Tom1 on Thu, 29 Oct 2015 14:48:48 GMT

Hi,

I think dir /s covers it quite well. Please find attached the listings for both Windows Kits/8.1 and Windows Kits/10.

Best regards,

Tom

File Attachments
1) listings.zip, downloaded 376 times

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 15:45:09 GMT

Tom1 wrote on Thu, 29 October 2015 15:48Hi,

I think dir /s covers it quite well. Please find attached the listings for both Windows Kits/8.1 and Windows Kits/10.

Best regards,

Tom

Thanks. I have fixed 'instant setup' for your tree, however, I have a very strong feeling that you really do not have SDK 10 installed: Notice that there e.g. are no win32 system includes anywhere. Only part I can see is standard C library.

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 15:46:44 GMT

Tom1 wrote on Thu, 29 October 2015 14:33I have both C:\Program Files (x86)\Windows Kits\10 (about 490 MB) and C:\Program Files (x86)\Windows Kits\8.1 (about 532 MB).

Tom

P.S.: Mine ...Kits/10 has 1.8G...

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Thu, 29 Oct 2015 17:18:28 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Thanks, I will try it tomorrow when back at the office.

I have not explicitly installed SDK 10 for sure. What I have, came with Visual studio community 2015. I can't check it here, but I have a strong feeling that I just installed the "Programming Languages > Visual C++ > Common Tools for Visual C++ 2015", which gives a tip: "Tools for creating Windows applications using the Visual Studio 2015 Visual C++ compiler toolset (v140). Also includes the Visual C++ 2015 libraries and project templates for Windows Desktop development."

Should I have selected something additional in order to make it work? Could you list the minimum set of items required for U++. Or is SDK 10 a separate download somewhere? If so, could you post a link to that?

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Thu, 29 Oct 2015 19:16:56 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 29 October 2015 18:18Hi Mirek,

Thanks, I will try it tomorrow when back at the office.

I have not explicitly installed SDK 10 for sure. What I have, came with Visual studio community 2015. I can't check it here, but I have a strong feeling that I just installed the "Programming Languages > Visual C++ > Common Tools for Visual C++ 2015", which gives a tip: "Tools for creating Windows applications using the Visual Studio 2015 Visual C++ compiler toolset (v140). Also includes the Visual C++ 2015 libraries and project templates for Windows Desktop development."

Should I have selected something additional in order to make it work? Could you list the minimum set of items required for U++. Or is SDK 10 a separate download somewhere? If so, could you post a link to that?

Best regards,

Tom

I wish I knew :)

There seems to be separate SDK here:

https://dev.windows.com/en-us/downloads/windows-10-sdk

Frankly, it is some time since I have installed 2015, not sure whether I had installed this separately or not :(

Subject: Re: Considering different approach to Win32 release
Posted by cbpporter on Fri, 30 Oct 2015 09:13:03 GMT

View Forum Message <> Reply to Message

Great idea with separating the nightly MINGW from the nightly UPP. Now at least one can update in a reasonable time.

So the "porting" to MINGW is done for both my console and GUI apps. The GUI optimal mode one do crash on startup on Windows 7, but otherwise I am fine and running out of the box UPP. Once MINGW is fixed for GUI, I'll use MINGW. But in some early testing, it is a bit slower.

Out of he box except for CodeEditor. But I found a "hack" for it. I copied it over from uppsrc to MyApps and merged my custom changes into it and I'm linking with it. CodeEditor is really great and the bread and butter for one of my GUI apps, the IDE, but it is not exactly customizable, so I was not able to provide general patches for it.

But I do have plans for it. I want to refactor it and instead of having languages and hard coded behavior for each, it will have about 6-7 new flags. A language will set keyword and a combination of flags, allowing CodeEditor to be decoupled from knowing about languages.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Fri, 30 Oct 2015 11:17:18 GMT

View Forum Message <> Reply to Message

cbpporter wrote on Fri, 30 October 2015 10:13Great idea with separating the nightly MINGW from the nightly UPP. Now at least one can update in a reasonable time.

So the "porting" to MINGW is done for both my console and GUI apps. The GUI optimal mode one do crash on startup on Windows 7, but otherwise I am fine and running out of the box UPP. Once MINGW is fixed for GUI, I'll use MINGW. But in some early testing, it is a bit slower.

For now, the fix is just replacing -O3 and -Os with -O1...

I consider MINGW just to serve two roles: Ability to test U++ without installing anything else and testing against GCC compiler.

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Fri, 30 Oct 2015 13:39:41 GMT
View Forum Message <> Reply to Message

Hi,

I tried with 9114 with not much luck. It did not detect MSC15.

I installed step-by-step more features from Visual Studio Community 2015 until after installing "Windows and Web Development > Universal Windows App Development Tools > Tools (1.1.1) and Windows SDK (10.0.10240)" TheIDE finally found MSC15 and MSC15x64.

Compiling ProtectTest on MSC15 (32bit) succeeds, but running results in message "Bad key !!&Do you want to continue anyways ?", which is pretty strange since the key is correct and intact. Does this work correctly on your machine?

EDIT: In fact many times the result is 'crash on start'. Seems a bit unstable behavior -- how is that possible?

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Fri, 30 Oct 2015 16:05:44 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Fri, 30 October 2015 14:39Hi,

I tried with 9114 with not much luck. It did not detect MSC15.

I installed step-by-step more features from Visual Studio Community 2015 until after installing "Windows and Web Development > Universal Windows App Development Tools > Tools (1.1.1) and Windows SDK (10.0.10240)" TheIDE finally found MSC15 and MSC15x64.

Well, after the hypothesis that you do NOT have in fact SDK installed, I have reverted changes for your tree (with missing SDK) - I think it is better that it is not installed than installed and not working (because of missing SDK).

Quote:
Compiling ProtectTest on MSC15 (32bit) succeeds, but running results in message "Bad key !!&Do you want to continue anyways ?", which is pretty strange since the key is correct and intact.

Does this work correctly on your machine?

EDIT: In fact many times the result is 'crash on start'. Seems a bit unstable behavior -- how is that possible?

EDIT belongs to ProtectTest?

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Fri, 30 Oct 2015 16:51:20 GMT

View Forum Message <> Reply to Message

Hi,

Yes, absolutely. Of course it should reject a partial/non-functional SDK. And now that I actually

The edit belongs to ProtectTest. This is the last big issue. Wonder if clang could have anything to offer here on the Windows side yet.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Mon, 02 Nov 2015 16:17:51 GMT

View Forum Message <> Reply to Message

OK, so I have digged deep into the protect and found the problem...

...well, the problem is that it cannot work the way it is designed...

```
double CryptedTest(double d, double e)
{
 PROTECT_START_FUNC(Decrypt);

 double f;
 f = d * e;

 PromptOK("CryptedTest DECRYPTED SUCCESFULLY!!!");
 return 2 * f + e;
 PROTECT_END_FUNC;
}
```

The problem here is that ProtectEncrypt absolutely ignores any linker fixups. This means that literal string constant in PromptOK gets encrypted 'raw'. Then, when program is run, .exe loader relocates *encrypted data*, and they get decrypted only after relocation, resulting in wrong address (that is why it crashes) and possibly broken decryption.

So, for now, if you need to use Protect, the only code it might be able to handle is the one that does not reference static data or other functions.

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Mon, 02 Nov 2015 16:34:23 GMT
View Forum Message <> Reply to Message

P.S.: Example and docs updated...

Subject: Re: Considering different approach to Win32 release
Posted by mdelfede on Mon, 02 Nov 2015 17:10:23 GMT
View Forum Message <> Reply to Message

mhhhhh... I don't remember all the work done, but IIRC at least up to MSC9 the linker fixups were not an issue.... or I never stumbled on them, and I have quite big functions encrypted in my app, which of course access other functions too.
Probably up to MSC9 the compiler/linker insert relative addressing and do not need fixups, ant that changed from version 10 upwards, I don't know.

Anyways, fixing such an issue is not trivial at all, if even possible.
The encrypter now do its job on executable on disk and the decrypter does it in memory when the fixups are (wrongly) applied, so there's no simple solution other than rethink the whole stuff.
Or, better said, a possible solution would be to leave fixups unencrypted, but then we'd need a table somewhere to tell decrypter to leave them alone. Not an easy task either, and would entail the complete PE header analysis.

Anyways, the package doesn't work either on 64 bit M$ compilers, which do not support online assembly.

So, I guess for now the package will stay as it is.... I'm using it on my app, and I'll stay with MSC9 too, for now, or for linux compilers which up to now behaves good with it.
For a more "professional" solution I'd need monthes to code it, and have no time now ;)

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Tue, 03 Nov 2015 08:27:32 GMT
View Forum Message <> Reply to Message

Hi,

Thanks for your effort on Protect. I can confirm that the last working MS compiler environment is MSC9 32 bit.

Mirek: I just tested 9142 with MSC15 and the updated ProtectTest still crashes on CryptedTest() function. Commenting out the call to CryptedTest() enables the rest of the demo to run correctly. So obfuscation actually works properly even with MSC15 while keyed protection crashes. My wild guess is that obtaining the key is the unwanted static/global reference.

Is it possible to make a decision to keep U++ source code compatible with MSC9 until there is a working replacement for the current Protect package?

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Tue, 03 Nov 2015 08:48:11 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 03 November 2015 09:27Hi,

Thanks for your effort on Protect. I can confirm that the last working MS compiler environment is MSC9 32 bit.

Mirek: I just tested 9142 with MSC15 and the updated ProtectTest still crashes on CryptedTest() function. Commenting out the call to CryptedTest() enables the rest of the demo to run correctly. So obfuscation actually works properly even with MSC15 while keyed protection crashes. My wild guess is that obtaining the key is the unwanted static/global reference.

Well, that '2' constant actually seems to be a problem... It gets stored in memory and is loaded.

Quote:
Is it possible to make a decision to keep U++ source code compatible with MSC9 until there is a working replacement for the current Protect package?

I have not changed anything in Protect package(s). Just added a bit to docs.

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Tue, 03 Nov 2015 08:52:17 GMT

View Forum Message <> Reply to Message

mdelfede wrote on Mon, 02 November 2015 18:10
Or, better said, a possible solution would be to leave fixups unencrypted, but then we'd need a table somewhere to tell decrypter to leave them alone. Not an easy task either, and would entail the complete PE header analysis.

Maybe you could limit crypting only to specific opcodes... I believe that only "absolute address" opcodes are 'dangerous'.

You could use ndisasm for testing (although it is a bit disgusting to use opensource to close the source :)

Mirek

Subject: Re: Considering different approach to Win32 release
Posted by Tom1 on Tue, 03 Nov 2015 10:33:01 GMT

View Forum Message <> Reply to Message

mirek wrote on Tue, 03 November 2015 10:48Tom1 wrote on Tue, 03 November 2015 09:27Hi,

Thanks for your effort on Protect. I can confirm that the last working MS compiler environment is MSC9 32 bit.

Mirek: I just tested 9142 with MSC15 and the updated ProtectTest still crashes on CryptedTest() function. Commenting out the call to CryptedTest() enables the rest of the demo to run correctly. So obfuscation actually works properly even with MSC15 while keyed protection crashes. My wild guess is that obtaining the key is the unwanted static/global reference.

Well, that '2' constant actually seems to be a problem... It gets stored in memory and is loaded.

Quote:
Is it possible to make a decision to keep U++ source code compatible with MSC9 until there is a working replacement for the current Protect package?

I have not changed anything in Protect package(s). Just added a bit to docs.

Mirek

I mean, I wish the entire U++ to be kept MSC9 compatible until Protect can be made to work with current compiler(s) e.g. MSC15. I just want to point out this need for backward compatibility when we are simultaneously talking about eventual switch to C++11, and therefore, MSC15.

Best regards,

Tom

Subject: Re: Considering different approach to Win32 release
Posted by mirek on Wed, 25 Nov 2015 12:14:16 GMT

View Forum Message <> Reply to Message

Well, as much as I am in fact sort of uncomfortable about this feature (a little by the mere fact that we have have such copy protection tool, but much more by how it is fragile by definition), I nevertheless continued thinking and digging....

Now here seems to be the function to determine the length of x86 instruction:

http://stackoverflow.com/questions/23788236/get-size-of-asse
mbly-instructions/23843450#23843450

Using this, I suggest to encrypt only the FIRST BYTE of each instruction. Should solve the problem...

Subject: Re: Considering different approach to Win32 release
Posted by mdelfede on Thu, 26 Nov 2015 08:35:54 GMT

View Forum Message <> Reply to Message

mirek wrote on Tue, 03 November 2015 09:52
.... (although it is a bit disgusting to use opensource to close the source :)
Mirek

Well, I find it funny, indeed :p
Anyways, I normally don't like closed source apps, but sometimes is a must....
I have an app that costed me tons of ours to develop and to make it user friendly.
Knowing my country's customers (in building design field) having it opensource would mean zero money earned for it....

BTW, your way of encrypting only the first byte is not bad. I'll look at it when I've a bit spare time.
I hope it will not require too much code.

Ciao

Massimo