Subject: U++/Win32 is now using CLANG toolchain
Posted by mirek on Sun, 01 Mar 2020 23:38:36 GMT
View Forum Message <> Reply to Message

I have found this wonderful little project:

https://github.com/mstorsjo/llvm-mingw

which provides CLANG toolchain in windows. This is great match for several reasons:

- it is multichain, single compiler for both 32 and 64 bits, which results in nice small footprint - in fact, I have dropped "without toolchain" windows version, it is now single 140MB .7z file...
- it has good optimal TLS support, unlike gcc in win32
- it is actually able to produce .pdb compatible debug info. That means we can use the same (superior) debugger as for Microsoft C++, including "smart" parts that now display U++ types (!) There are some issues that I plan to gradually fix with workarounds, but overal this seems to work prette well.

Currently it has some downsides too, namely

- targets Windows universal CRT (instead of MSVCRT.dll), which means that for previous versions you need to ship some .dlls with your application. This might or might not be fixed in future...
- being there, it was not possible to have single static linked OpenSSL library for both clang and msvc, so I have gave up and OpenSSL is once again linked dynamically (just as SDL2, MYSQL, PGSQL are)
- there are some issues with weak references, which affected global new/delete overloading)https://github.com/mstorsjo/llvm-mingw/issues/91), but we have workaround for now
- there is also some issue with std::chrono timers, not able to reproduce, but I needed to revert 'msecs' function to older code to fix some weird issues in theide.

Overal, this is a great step forward, next release will be great!

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Xemuth on Mon, 02 Mar 2020 07:55:50 GMT
View Forum Message <> Reply to Message

Thanks for this implementation, I plan to compile several of my personal project with it.
In case of problem, do we use this post as "bug/ dysfunction reporting" ? or shall we create a new thread ?

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by koldo on Mon, 02 Mar 2020 09:33:54 GMT
View Forum Message <> Reply to Message

Thank you Mirek.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Oblivion on Mon, 02 Mar 2020 11:19:37 GMT
View Forum Message <> Reply to Message

Thank you very much, Mirek!

Best regards,
Oblivion

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Novo on Tue, 03 Mar 2020 19:57:27 GMT
View Forum Message <> Reply to Message

Are you planning to add cross-compilation from Linux to Windows?
It looks like LLVM-MinGW supports that.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 04 Mar 2020 08:37:21 GMT
View Forum Message <> Reply to Message

Did not came to my mind...

Currently I am using wine to build win32 stuff in Linux. TheIDE works just fine in wine..

Anyway, if you would like to investigate this area, that would be nice...

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by koldo on Wed, 04 Mar 2020 13:09:34 GMT
View Forum Message <> Reply to Message

TheIDE compiled with CLANG in Windows 10. It works well

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 04 Mar 2020 13:16:08 GMT
View Forum Message <> Reply to Message

koldo wrote on Wed, 04 March 2020 14:09TheIDE compiled with CLANG in Windows 10. It works

well

Yeah, I am actually working in clang compiled ide for about 14 days now. After some initial fixes (e.g. chrono does not seem to work, so I have to revert to olde msecs implementation) everything seems to be rock stable.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Tue, 24 Mar 2020 19:05:29 GMT
View Forum Message <> Reply to Message

Hi,

Is it possible to take a look at the bazaar/Protect with CLANG? I already noticed that removing file xed/VERSION fixes the compilation issue of Xed package that Protect uses.

However, something goes wrong when running ProtectEncrypt for an exe file created with CLANG.

-

Also, I use some external static libraries and they in turn have some Windows library dependencies that are not being satisfied:

Linking...
lld-link: error: could not open 'libLIBCMT.a': No such file or directory
lld-link: error: could not open 'libOLDNAMES.a': No such file or directory
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (0:00.50)

In reality these libraries are called 'libcmt.lib' and 'oldnames.lib' and are included with e.g. MSBT. Any suggestions how to proceed here?

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Tue, 24 Mar 2020 20:30:50 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Tue, 24 March 2020 20:05

Linking...
lld-link: error: could not open 'libLIBCMT.a': No such file or directory
lld-link: error: could not open 'libOLDNAMES.a': No such file or directory
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (0:00.50)

In reality these libraries are called 'libcmt.lib' and 'oldnames.lib' and are included with e.g. MSBT.
Any suggestions how to proceed here?

Best regards,

Tom

In organizer, specify them with ".lib" extension. The build will then take it "literally".

Mirek

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Wed, 25 Mar 2020 07:57:20 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Adding the libs manually
when: WIN32
Libraries: ../lib/libcmt.lib ../lib/liboldnames.lib

introduces a different issue:
Linking...
lld-link: error: duplicate symbol: atexit
>>> defined at ../crt\crtexe.c:438
>>> C:\upp-svn\bin\clang\x86_64-w64-mingw32\lib\crt2.o
>>> defined at libcmt.lib(utility.obj)

lld-link: error: duplicate symbol: __dyn_tls_init
>>> defined at d:\agent_work\4\s\src\vctools\crt\vcstartup\src\tls\tlsdyn.cpp:87
>>> libcmt.lib(tlsdyn.obj)
>>> defined at ../crt\tlssup.c:77
>>> libmingw32.a(lib64_libmingw32_a-tlssup.o)

lld-link: error: duplicate symbol: __dyn_tls_init_callback
>>> defined at libcmt.lib(tlsdyn.obj)
>>> defined at ../crt\tlssup.c:103

```
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)

lld-link: error: duplicate symbol: _onexit
>>> defined at d:\agent\_work\4\s\src\vctools\crt\vcstartup\src\utility\utility.cpp:256
>>>             libcmt.lib(utility.obj)
>>> defined at ../crt\ucrtbase_compat.c:86
>>>             libmsvcrt.a(lib64_libucrt_extra_a-ucrtbase_compat.o)

lld-link: error: duplicate symbol: _tls_index
>>> defined at ../crt\tlssup.c:35
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: duplicate symbol: _tls_start
>>> defined at ../crt\tlssup.c:41
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: duplicate symbol: _tls_end
>>> defined at ../crt\tlssup.c:42
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: duplicate symbol: __xl_a
>>> defined at ../crt\tlssup.c:44
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: duplicate symbol: __xl_z
>>> defined at ../crt\tlssup.c:45
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: duplicate symbol: _tls_used
>>> defined at ../crt\tlssup.c:47
>>>             libmingw32.a(lib64_libmingw32_a-tlssup.o)
>>> defined at libcmt.lib(tlssup.obj)

lld-link: error: could not open 'liblibvcruntime.a': No such file or directory
lld-link: error: could not open 'liblibucrt.a': No such file or directory
lld-link: error: could not open 'libLIBCMT.a': No such file or directory
lld-link: error: could not open 'libOLDNAMES.a': No such file or directory
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (0:00.50)
```

So adding the libraries yields duplicate symbols while adding yet more unsatisfied dependencies. Unfortunately, they still do not satisfy the requirement for the strange 'libLIBCMT.a' and

'libOLDNAMES.a'. (I also tried to copy the original files to these names, but that did not work either.)

I wonder if I should create a couple of dummy libraries with no public symbols and give them names 'libLIBCMT.a' and 'libOLDNAMES.a'...

Any suggestions to try?

Best regards,

Tom

[EDIT] Well, those dummy empty libraries with names 'libLIBCMT.a' and 'libOLDNAMES.a' did not help either.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 25 Mar 2020 11:41:59 GMT
View Forum Message <> Reply to Message

Why do you need those libs in the first place?

libcmt is AFAIK old C library. No wonder there are duplicates....

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Wed, 25 Mar 2020 11:52:10 GMT
View Forum Message <> Reply to Message

Hi,

I use some special devices that are accompanied by static libraries to be used to access those devices. When I add those specific libraries (lib files) into the project, the requirement for these two (old) Windows libraries appear.

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 25 Mar 2020 12:00:45 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 25 March 2020 12:52Hi,

I use some special devices that are accompanied by static libraries to be used to access those

devices. When I add those specific libraries (lib files) into the project, the requirement for these two (old) Windows libraries appear.

Best regards,

Tom

Understood. Does it work with original U++ shipped mingw? (I am asking because mingw AFAIK is using msvcrt and the clang toolchain we are using should be possible to be recompiled to use msvcrt as well, something I was considering anyway but postponed because of other issues to solve first).

Mirek

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Wed, 25 Mar 2020 12:24:55 GMT
View Forum Message <> Reply to Message

OK, I need to check that separately.

As a matter of fact I never really used MINGW version as ... well I cannot remember exactly, but it did not work for me.

Can you point out where to download the last MINGW version?

Best regards,

Tom

[EDIT] Nevermind... I found the 2019.2 MINGW on sourceforge.net...

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 25 Mar 2020 13:28:29 GMT
View Forum Message <> Reply to Message

sf.net?

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Wed, 25 Mar 2020 14:37:17 GMT
View Forum Message <> Reply to Message

OK, upp-mingw-13664 tested. The same result as with CLANG:

Linking...

lld-link: error: could not open libLIBCMT.a: no such file or directory
lld-link: error: could not open libOLDNAMES.a: no such file or directory
collect2.exe: error: ld returned 1 exit status

Also, if I explicitly give the libraries, the behavior follows that of CLANG.

--

BTW: It seems that MINGW is not able to deal with GDAL/OGR source properly... and some other packages too. I guess this was one of my reasons to drop it.

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 25 Mar 2020 16:50:32 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 25 March 2020 15:37OK, upp-mingw-13664 tested. The same result as with CLANG:

Linking...
lld-link: error: could not open libLIBCMT.a: no such file or directory
lld-link: error: could not open libOLDNAMES.a: no such file or directory
collect2.exe: error: ld returned 1 exit status

Also, if I explicitly give the libraries, the behavior follows that of CLANG.

--

BTW: It seems that MINGW is not able to deal with GDAL/OGR source properly... and some other packages too. I guess this was one of my reasons to drop it.

Best regards,

Tom

Too bad...

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Thu, 26 Mar 2020 10:40:55 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 25 March 2020 08:57Hi Mirek,
Libraries: ../lib/libcmt.lib ../lib/liboldnames.lib

I was thinking about it; if you do not link those libs, what symbols are missing?

Maybe it would be possible to provide the implementation just for missing symbols as workaround? Perhaps even check wine sources to get a clue what they are supposed to do... or MSC library sources - I think those are available.

Mirek

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Thu, 26 Mar 2020 11:00:42 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Actually, I think it is not about missing symbols at all...

I opened the binary .lib file and noticed that it contains the following text string:

/DEFAULTLIB:"LIBCMT" /DEFAULTLIB:"OLDNAMES"

IMO, these are the problem. I just can't figure out how to remove these dependencies cleanly. All my attempts with binary editing resulted in invalid library file.

Is there some LIB or AR command that could be used to remove these? Then I could see missing symbols and try to fix them.

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Thu, 26 Mar 2020 15:55:07 GMT
View Forum Message <> Reply to Message

Hi,

One step forward...

when: WIN32 CLANG
Link options: -Xlinker /NODEFAULTLIB:OLDNAMES -Xlinker /NODEFAULTLIB:LIBCMT

This prevents those default libs to be linked.

However, this comes with a price tag:

```
Linking...
lld-link: error: undefined symbol: __GSHandlerCheck
lld-link: error: undefined symbol: __security_check_cookie
lld-link: error: undefined symbol: __report_rangecheckfailure
lld-link: error: undefined symbol: __chkstk
lld-link: error: undefined symbol: _fltused
lld-link: error: undefined symbol: EnumPrintersA
lld-link: error: undefined symbol: OpenPrinterA
lld-link: error: undefined symbol: SetPrinterA
lld-link: error: undefined symbol: ClosePrinter
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)

There were errors. (0:00.61)
```

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Thu, 26 Mar 2020 16:26:41 GMT
View Forum Message <> Reply to Message

And here's finally the solution to this game. Had to add fake functions to satisfy the dependencies. I know these functions do not do anything, but I decided not to really need those anyway...

```
#ifdef __MINGW32__
extern "C"{
 void __cdecl __GSHandlerCheck() { }
 void __cdecl __security_check_cookie(uintptr_t i) { }
 void __cdecl __chkstk() { }
 void __cdecl __report_rangecheckfailure() { }
 void __cdecl _fltused() { }
 WINBOOL WINAPI EnumPrintersA(DWORD Flags,LPSTR Name,DWORD Level,LPBYTE
pPrinterEnum,DWORD cbBuf,LPDWORD pcbNeeded,LPDWORD pcReturned) { return false; }
 WINBOOL WINAPI OpenPrinterA(LPSTR pPrinterName,LPHANDLE
phPrinter,LPPRINTER_DEFAULTSA pDefault) { return false; }
 WINBOOL WINAPI SetPrinterA(HANDLE hPrinter,DWORD Level,LPBYTE pPrinter,DWORD
Command) { return false; }
 WINBOOL WINAPI ClosePrinter(HANDLE hPrinter) { return false; }
}
```

#endif //__MINGW32__

One more library to go... with a different challenge.

Best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Thu, 26 Mar 2020 17:39:28 GMT
View Forum Message <> Reply to Message

What about create empty, add 'manually'? It looks like in the step where you tried to add empty libs, you did not combine that with this.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Tom1 on Thu, 26 Mar 2020 17:57:29 GMT
View Forum Message <> Reply to Message

The above solution is actually easier for me as I do not need to maintain separate dummy empty libs. The lines of code required can be included in the module using the library so it is quite straight forward in this special case.

Thanks and best regards,

Tom

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Thu, 26 Mar 2020 21:24:27 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Thu, 26 March 2020 18:57The above solution is actually easier for me as I do not need to maintain separate dummy empty libs. The lines of code required can be included in the module using the library so it is quite straight forward in this special case.

Thanks and best regards,

Tom

Cool!

Mirek

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Oblivion on Thu, 13 Aug 2020 10:13:19 GMT
View Forum Message <> Reply to Message

Hello Mirek,

One problem I've noticed about CLANG on windows: Windows 10 api doesn't seem to be supported by Upp. They cant't be compiled because some of them are conditionally included (ver >= 0x0A00, ver >= 0x0700, etc.)

(I've noticed this while implementing the windows 10 pseudoconsole API support for terminal ctrl.)

The headers, and prototypes are already included in CLANG bundle but It requires manual intervention in Core/config.h to change _WIN32_WINNT from 0x0501 to 0x0A00. (redefining it elsewhere doesn't seem to work either; or at least I couldn't find a simple way)

If I set the version number to 0x0A00 everthing compiles fine on CLANG.
(Also, there is no such problem with MSVC 19. It does not require any intervention.)

Is there a workaround for this problem? (may be a flag in TheIDE's main configuration, etc. ?)

Best regards,
Oblivion

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Thu, 13 Aug 2020 12:20:43 GMT
View Forum Message <> Reply to Message

Oblivion wrote on Thu, 13 August 2020 12:13Hello Mirek,

One problem I've noticed about CLANG on windows: Windows 10 api doesn't seem to be supported by Upp. They cant't be compiled because some of them are conditionally included (ver >= 0x0A00, ver >= 0x0700, etc.)

(I've noticed this while implementing the windows 10 pseudoconsole API support for terminal ctrl.)

The headers, and prototypes are already included in CLANG bundle but It requires manual intervention in Core/config.h to change _WIN32_WINNT from 0x0501 to 0x0A00. (redefining it elsewhere doesn't seem to work either; or at least I couldn't find a simple way)

If I set the version number to 0x0A00 everthing compiles fine on CLANG.
(Also, there is no such problem with MSVC 19. It does not require any intervention.)

Is there a workaround for this problem? (may be a flag in TheIDE's main configuration, etc. ?)

Best regards,
Oblivion

I guess adding a flag is the most appropriate at this moment. flagWIN10?

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Oblivion on Thu, 13 Aug 2020 14:50:30 GMT
View Forum Message <> Reply to Message

Quote:I guess adding a flag is the most appropriate at this moment. flagWIN10?

I think that would be fine.

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by Oblivion on Tue, 18 Aug 2020 18:31:59 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Can we set the definitions in config.h to:

```
#ifdef flagWIN10
 #define _WIN32_WINNT _WIN32_WINNT_WIN10
 #undef NTDDI_VERSION
 #define NTDDI_VERSION NTDDI_WIN10_19H1
#else
 #define _WIN32_WINNT _WIN32_WINNT_WINXP
#endif
```

Apparently CLANG Windows 10 api requires setting both the _WIN32_WINNT amd NTDDI_VERSION, to properly -fully- work.
It seems that both versioning constants have to match, and the default NTDDI is set to WINXP.

The WIN10_19H1 is the latest windows 10 upgrade that our CLANG bundle supports. (Gives better Win10 api access to driver kits, which pseudoconsole requires. It doesn't seem to break anything anyway.).

Best regards,
Oblivion

Subject: Re: U++/Win32 is now using CLANG toolchain
Posted by mirek on Wed, 19 Aug 2020 07:16:26 GMT
View Forum Message <> Reply to Message

[quote title=Oblivion wrote on Tue, 18 August 2020 20:31]Hello Mirek,

Can we set the definitions in config.h to:

```
#ifdef flagWIN10
 #define _WIN32_WINNT _WIN32_WINNT_WIN10
 #undef NTDDI_VERSION
 #define NTDDI_VERSION NTDDI_WIN10_19H1
#else
 #define _WIN32_WINNT _WIN32_WINNT_WINXP
#endif
```

Can it be

```
#ifdef flagWIN10
 #define _WIN32_WINNT _WIN32_WINNT_WIN10
 #undef NTDDI_VERSION
 #define NTDDI_VERSION NTDDI_WIN10_19H1
#else
 #define _WIN32_WINNT 0x0501
#endif
```

? (more conservative w.r.t. _WIN32_WINNT_WIN10 availability)

Mirek