Subject: 2020.1 officially released
Posted by mirek on Fri, 08 May 2020 14:26:56 GMT
View Forum Message <> Reply to Message

https://sourceforge.net/p/upp/news/2020/05/u-20201-released/

Subject: Re: 2020.1 officially released
Posted by Tom1 on Fri, 08 May 2020 17:08:41 GMT
View Forum Message <> Reply to Message

Hi!

Thanks for your hard work!

Best regards,

Tom

Subject: Re: 2020.1 officially released
Posted by forlano on Fri, 08 May 2020 20:09:02 GMT
View Forum Message <> Reply to Message

Thanks a lot!

Luigi

Subject: Re: 2020.1 officially released
Posted by deep on Sat, 09 May 2020 14:27:27 GMT
View Forum Message <> Reply to Message

Great

Thanks a lot.

Subject: Re: 2020.1 officially released
Posted by BioBytes on Sat, 09 May 2020 19:41:32 GMT
View Forum Message <> Reply to Message

Great work

Thanks a lot to all contributors to U++

Biobytes

Subject: Re: 2020.1 officially released
Posted by h3l1 on Wed, 13 May 2020 09:33:06 GMT
View Forum Message <> Reply to Message

Hi Mirek,

great work, really looks good with the dark theme and new GTK3 on my 4K 15" notebook running Ubuntu 18.04.

Found a small problem: was just testing the Bombs application from examples and the window size does not adapt when selecting a different size.
Maybe also adapt the UNIT size for hidpi...

Greetings
Heli

Subject: Re: 2020.1 officially released
Posted by mirek on Wed, 13 May 2020 12:02:31 GMT
View Forum Message <> Reply to Message

h3l1 wrote on Wed, 13 May 2020 11:33Hi Mirek,

great work, really looks good with the dark theme and new GTK3 on my 4K 15" notebook running Ubuntu 18.04.

Found a small problem: was just testing the Bombs application from examples and the window size does not adapt when selecting a different size.
Maybe also adapt the UNIT size for hidpi...

Greetings
Heli

Thanks, good catch, fixed in trunk.

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 15 May 2020 14:17:05 GMT
View Forum Message <> Reply to Message

Thanks for the new release.

It seems to have a couple of issues with Mac.

It is impossible to compile umk using makefile via "make -f uMakefile".
ld: warning: option -s is obsolete and being ignored
ld: unknown option: -O
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Even if you have a prebuilt version of umk, an attempt to build ide via "umk uppsrc ide CLANG +brus" leads to
ld: warning: directory not found for option '-L/opt/local/lib'
ld: library not found for -lcrt0.o
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Interestingly, CLANG.bm, automatically created by umk, contains options below.
INCLUDE = "/opt/local/include;/usr/include";
LIB = "/opt/local/lib;/usr/lib";

Although they are not needed.

Subject: Re: 2020.1 officially released
Posted by mirek on Sat, 16 May 2020 07:33:40 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 15 May 2020 16:17Thanks for the new release.

It seems to have a couple of issues with Mac.
It is impossible to compile umk using makefile via "make -f uMakefile".
ld: warning: option -s is obsolete and being ignored
ld: unknown option: -O
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Even if you have a prebuilt version of umk, an attempt to build ide via "umk uppsrc ide CLANG +brus" leads to
ld: warning: directory not found for option '-L/opt/local/lib'
ld: library not found for -lcrt0.o
clang: error: linker command failed with exit code 1 (use -v to see invocation)
Interestingly, CLANG.bm, automatically created by umk, contains options below.
INCLUDE = "/opt/local/include;/usr/include";
LIB = "/opt/local/lib;/usr/lib";

Although they are not needed.

Are you trying with upp-posix.tar.xz?

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 16 May 2020 11:32:57 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 16 May 2020 03:33Are you trying with upp-posix.tar.xz?
All this is based on src from git.

Basically, "git clone" and "make -f uMakefile".

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 16 May 2020 11:38:08 GMT
View Forum Message <> Reply to Message

I'm not the first one who tried to do that. There is a related ticket.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 16 May 2020 11:54:30 GMT
View Forum Message <> Reply to Message

This is happening on OSX 10.13.
clang is installed via brew.
$which clang++
/usr/bin/clang++

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 04:31:49 GMT
View Forum Message <> Reply to Message

Another interesting observation, this time about timing with Clang on Windows.
----- GLDrawDemo (GUI MAIN CLANG SHARED BLITZ WIN32) (14 / 14)
main.cpp
plugin/tess2: 8 file(s) built in (0:00.90), 113 msecs / file
GLCtrl: 3 file(s) built in (0:00.58), 194 msecs / file
GLDrawDemo: 1 file(s) built in (0:00.79), 794 msecs / file
GLDraw: 20 file(s) built in (0:01.22), 61 msecs / file
plugin/glew: 1 file(s) built in (1:19.54), 79540 msecs / file <--- !!!

----- ScatterDraw_Demo (MAIN CLANG DEBUG SHARED DEBUG_FULL BLITZ WIN32) (8 / 8)
ScatterDraw_Demo.cpp
ScatterDraw_Demo: 1 file(s) built in (0:02.22), 2225 msecs / file
ScatterDraw: 8 file(s) built in (1:03.92), 7991 msecs / file <--- !!!

Actually, I was using wine on Linux.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 05:13:29 GMT
View Forum Message <> Reply to Message

Interesting message when compiling on Windows with CLANGx64:

#0 0x0000000142abaa9c (clang-10+0x2abaa9c)
#1 0x0000000142a52a8c (clang-10+0x2a52a8c)
#2 0x0000000142a6b016 (clang-10+0x2a6b016)
#3 0x0000000142a60146 (clang-10+0x2a60146)
clang-10: error: clang frontend command failed due to signal (use -v to see invocation)
clang version 10.0.0 (https://github.com/llvm/llvm-project.git
d32170dbd5b0d54436537b6b75beaf44324e0c28)
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: c:\local\apps\upp\bin\clang\bin
clang-10: note: diagnostic msg: PLEASE submit a bug report to https://bugs.llvm.org/ and include the crash backtrace, preprocessed source, and associated run script.
heapdbg.cpp
CharSet.cpp
t.cpp
z.cpp
lz4.c
xxhash.c
clang-10: note: diagnostic msg:
****
PLEASE ATTACH THE FOLLOWING FILES TO THE BUG REPORT:
Preprocessed source(s) and associated run script(s) are located at:
clang-10: note: diagnostic msg: C:\users\ssg\Temp\wkparser$blitz-44ef80.cpp
clang-10: note: diagnostic msg: C:\users\ssg\Temp\wkparser$blitz-44ef80.sh
clang-10: note: diagnostic msg:
****
plugin/bz2: 2 file(s) built in (0:00.37), 186 msecs / file
plugin/bmp: 4 file(s) built in (0:00.30), 76 msecs / file
dvlp/ctrl/about: 1 file(s) built in (0:00.33), 332 msecs / file
Painter: 28 file(s) built in (0:01.72), 61 msecs / file
dvlp/wiki/wkparser: 6 file(s) built in (0:00.47), 79 msecs / file
Draw: 35 file(s) built in (0:00.47), 13 msecs / file
RichText: 22 file(s) built in (0:00.50), 23 msecs / file
CtrlCore: 62 file(s) built in (0:00.87), 14 msecs / file
CtrlLib: 58 file(s) built in (0:02.91), 50 msecs / file
Core: 68 file(s) built in (0:05.87), 86 msecs / file
There were errors. (0:17.72)
program finished with exit code 1
elapsedTime=18.027474

And this is primarily C++98 code with a lot of templates.
I mean my own app.

Subject: Re: 2020.1 officially released
Posted by mirek on Sat, 30 May 2020 06:55:24 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 30 May 2020 06:31Another interesting observation, this time about timing with Clang on Windows.
----- GLDrawDemo (GUI MAIN CLANG SHARED BLITZ WIN32) (14 / 14)
main.cpp
plugin/tess2: 8 file(s) built in (0:00.90), 113 msecs / file
GLCtrl: 3 file(s) built in (0:00.58), 194 msecs / file
GLDrawDemo: 1 file(s) built in (0:00.79), 794 msecs / file
GLDraw: 20 file(s) built in (0:01.22), 61 msecs / file
plugin/glew: 1 file(s) built in (1:19.54), 79540 msecs / file <--- !!!

----- ScatterDraw_Demo (MAIN CLANG DEBUG SHARED DEBUG_FULL BLITZ WIN32) (8 / 8)
ScatterDraw_Demo.cpp
ScatterDraw_Demo: 1 file(s) built in (0:02.22), 2225 msecs / file
ScatterDraw: 8 file(s) built in (1:03.92), 7991 msecs / file <--- !!!

Actually, I was using wine on Linux.

Those numbers are real? I mean, is it a problem with clang, or problem with builder printing wrong number?

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Sat, 30 May 2020 06:57:48 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 30 May 2020 07:13Interesting message when compiling on Windows with CLANGx64:
#0 0x0000000142abaa9c (clang-10+0x2abaa9c)
#1 0x0000000142a52a8c (clang-10+0x2a52a8c)
#2 0x0000000142a6b016 (clang-10+0x2a6b016)
#3 0x0000000142a60146 (clang-10+0x2a60146)
clang-10: error: clang frontend command failed due to signal (use -v to see invocation)
clang version 10.0.0 (https://github.com/llvm/llvm-project.git
d32170dbd5b0d54436537b6b75beaf44324e0c28)
Target: x86_64-w64-windows-gnu
Thread model: posix
InstalledDir: c:\local\apps\upp\bin\clang\bin
clang-10: note: diagnostic msg: PLEASE submit a bug report to https://bugs.llvm.org/ and include the crash backtrace, preprocessed source, and associated run script.
heapdbg.cpp
CharSet.cpp
t.cpp
z.cpp
lz4.c
xxhash.c
clang-10: note: diagnostic msg:

****
PLEASE ATTACH THE FOLLOWING FILES TO THE BUG REPORT:
Preprocessed source(s) and associated run script(s) are located at:
clang-10: note: diagnostic msg: C:\users\ssg\Temp\wkparser$blitz-44ef80.cpp
clang-10: note: diagnostic msg: C:\users\ssg\Temp\wkparser$blitz-44ef80.sh
clang-10: note: diagnostic msg:
****
plugin/bz2: 2 file(s) built in (0:00.37), 186 msecs / file
plugin/bmp: 4 file(s) built in (0:00.30), 76 msecs / file
dvlp/ctrl/about: 1 file(s) built in (0:00.33), 332 msecs / file
Painter: 28 file(s) built in (0:01.72), 61 msecs / file
dvlp/wiki/wkparser: 6 file(s) built in (0:00.47), 79 msecs / file
Draw: 35 file(s) built in (0:00.47), 13 msecs / file
RichText: 22 file(s) built in (0:00.50), 23 msecs / file
CtrlCore: 62 file(s) built in (0:00.87), 14 msecs / file
CtrlLib: 58 file(s) built in (0:02.91), 50 msecs / file
Core: 68 file(s) built in (0:05.87), 86 msecs / file
There were errors. (0:17.72)
program finished with exit code 1
elapsedTime=18.027474

And this is primarily C++98 code with a lot of templates.
I mean my own app.

It would be good to narrow that down to single file / construct and send them report.

For what is worth, for me CLANG was rock stable so far, except maybe some issues with debugger info (but those unfortunately exist with msc as well).

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 15:10:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 30 May 2020 02:55Those numbers are real? I mean, is it a problem with clang, or problem with builder printing wrong number?

Mirek
Yes, numbers are real. It is Clang.
Another case with CLANGx64.
----- AddressBookWeb (MT MAIN CLANG SHARED BLITZ WIN32) (10 / 10)
Main.cpp
Pages.icpp
2 warnings generated.
Skylark/Iml: 1 file(s) built in (0:00.16), 160 msecs / file
plugin/jpg: 61 file(s) built in (0:04.61), 75 msecs / file

plugin/png: 3 file(s) built in (0:01.07), 358 msecs / file
AddressBookWeb: 2 file(s) built in (0:01.04), 523 msecs / file
Draw: 35 file(s) built in (0:05.58), 159 msecs / file
plugin/sqlite3: 2 file(s) built in (1:46.72), 53363 msecs / file
Linking...
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\.cache\upp.out\CLANGx64.Blitz.Mt.Shared\AddressBookWeb.exe (6354432 B) linked in (0:00.72)

As you can see, linking is blazingly fast.
But almost two minutes to compile sqlite3 is unacceptable.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 15:56:59 GMT
View Forum Message <> Reply to Message

Another sqlite3 case:
----- SQL_Sqlite3 (MAIN CLANG SHARED BLITZ WIN32) (5 / 5)
simple.cpp
In file included from
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\uppsrc\plugin\sqlite3\lib.c:8:
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\uppsrc\plugin\sqlite3/lib/sqleet.c:113138:38: warning: implicit conversion from 'long long' to 'double' changes value from 9223372036854775806 to 9223372036854775808 [-Wimplicit-int-float-conversion]
  if(n==0 && r>=0 && r<LARGEST_INT64-1){
                      ~~~~~~~~~~~~~~~^~
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\uppsrc\plugin\sqlite3/lib/sqleet.c:113140:46: warning: implicit conversion from 'long long' to 'double' changes value from 9223372036854775806 to 9223372036854775808 [-Wimplicit-int-float-conversion]
  }else if(n==0 && r<0 && (-r)<LARGEST_INT64-1){
                           ~~~~~~~~~~~~~~~^~
2 warnings generated.
SQL_Sqlite3: 1 file(s) built in (0:00.93), 938 msecs / file
plugin/sqlite3: 2 file(s) built in (2:00.78), 60390 msecs / file
Linking...
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\.cache\upp.out\CLANGx64.Blitz.Shared\SQL_Sqlite3.exe (5206016 B) linked in (0:00.71)

This time it ran for exactly two minutes. I watched compilation process in top.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 15:57:59 GMT
View Forum Message <> Reply to Message

I need to double-check my BM-files. It is possible that all this is my fault.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 20:16:06 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 30 May 2020 02:57
It would be good to narrow that down to single file / construct and send them report.

For what is worth, for me CLANG was rock stable so far, except maybe some issues with debugger info (but those unfortunately exist with msc as well).

Mirek
I've updated bm-files. Nothing has really changed.
This compiler crash is still there.
Interestingly, all apps in Upp itself can be built without any problems (x64).
This is just my app causing clang to crash, but the compiler is crashing when compiling Core. This is weird.
I'll double-check situation with i686. Everything was fine with it.

Subject: Re: 2020.1 officially released
Posted by Novo on Sat, 30 May 2020 20:52:26 GMT
View Forum Message <> Reply to Message

Problem is solved. I was able to reproduce this crash on Linux.
Clang is crashing in a third-party library, which requires C++17, but the code was compiled with -std=c++14.
Instead of reporting an error Clang just crashes.

Life is going on.

Subject: Re: 2020.1 officially released
Posted by mirek on Sun, 31 May 2020 21:13:50 GMT
View Forum Message <> Reply to Message

Novo wrote on Sat, 30 May 2020 22:52Problem is solved. I was able to reproduce this crash on Linux.
Clang is crashing in a third-party library, which requires C++17, but the code was compiled with -std=c++14.
Instead of reporting an error Clang just crashes.

Life is going on.

What about those compile times?

Subject: Re: 2020.1 officially released

Posted by Novo on Sun, 31 May 2020 22:36:46 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 31 May 2020 17:13
What about those compile times?
Nothing has changed.
The worst case I've seen is reference/Eigen_demo. I never saw end of compilation in Debug conf.
Release compiles in 1.5 minutes.
I cannot check compilation speed at the moment because your last commit broke compilation of reference/EscApp with Clang on Windows.

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 08:19:00 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 00:36mirek wrote on Sun, 31 May 2020 17:13
What about those compile times?
Nothing has changed.
The worst case I've seen is reference/Eigen_demo. I never saw end of compilation in Debug conf.
Release compiles in 1.5 minutes.
I cannot check compilation speed at the moment because your last commit broke compilation of reference/EscApp with Clang on Windows.

Fixed...

Well, in Win32 it works more or less fine:

----- Core (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (1 / 4)
----- plugin/Eigen (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (2 / 4)
----- plugin/z (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (3 / 4)
----- Eigen_demo (MAIN CLANG DEBUG DEBUG_FULL BLITZ WIN32) (4 / 4)
eigen_demo.cpp
non-linear.cpp
fft.cpp
Eigen_demo: 3 file(s) built in (0:26.75), 8916 msecs / file
Linking...
C:\upp\out\reference\CLANGx64.Debug.Debug_Full\Eigen_demo.ex e (6641664 B) linked in (0:00.85)

OK. (0:28.09)

It however seems to take a bit longer to compile fft.cpp...

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 08:23:42 GMT
View Forum Message <> Reply to Message

OK, eigen_demo.cpp alone (Alt+F7) takes 24s to compile with CLANGx64. That's a lot.

However, if I use Visual C++, it takes to compile.... 24s. So perhaps it is simply the complexity of eigen...

Mirek

Subject: Re: 2020.1 officially released
Posted by koldo on Mon, 01 Jun 2020 12:17:34 GMT
View Forum Message <> Reply to Message

Yes, Eigen uses template metaprogramming to optimize speed handling SSE2 and Altivec. The cost of it is longer compiling time.
Just a simple vector sum like u = v + w, has some black magic inside.
This is explined here.

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 19:35:10 GMT
View Forum Message <> Reply to Message

There is another unexpected problem with Clang on Windows.
If I add "-std=c++17" to cpp options, then in case of GUI app (example: tutorial/Gui01) in Debug configuration I get this (Release is fine):
Linking...
lld-link: error: duplicate symbol: std::__throw_bad_alloc()
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:315
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator new(unsigned long long)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:301
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator new(unsigned long long, std::nothrow_t const&)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:307
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator new[](unsigned long long)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:304
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o

>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator new[](unsigned long long, std::nothrow_t const&)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:310
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator delete(void*)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:302
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator delete(void*, std::nothrow_t const&)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:308
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator delete[](void*)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:305
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator delete[](void*, std::nothrow_t const&)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:311
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debug_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)
clang-10: error: linker command failed with exit code 1 (use -v to see invocation)

Because STD_NEWDELETE is defined, and it gets defined in config.h here:
#ifdef flagCLR
#define flagUSEMALLOC
#define STD_NEWDELETE
#endif

I have no idea where flagCLR is coming from ...
wine umk tutorial Gui01 CLANGx64cpp17 -bu

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 19:41:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 01 June 2020 04:23OK, eigen_demo.cpp alone (Alt+F7) takes 24s to compile with CLANGx64. That's a lot.

However, if I use Visual C++, it takes to compile.... 24s. So perhaps it is simply the complexity of eigen...

Mirek
In my case it takes 1.5 minutes to compile plugins/sqlite3. IMHO, this is unacceptable ...
P.S. I compile with BLITZ enabled even in Release.

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 19:46:44 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 21:41mirek wrote on Mon, 01 June 2020 04:23OK, eigen_demo.cpp alone (Alt+F7) takes 24s to compile with CLANGx64. That's a lot.

However, if I use Visual C++, it takes to compile.... 24s. So perhaps it is simply the complexity of eigen...

Mirek
In my case it takes 1.5 minutes to compile plugins/sqlite3. IMHO, this is unacceptable ...
P.S. I compile with BLITZ enabled even in Release.

OK, indeed release with blitz, it is 1 minute. Interesting... That said, I do not see what we can do here...

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 19:53:42 GMT
View Forum Message <> Reply to Message

The problematic file is plugin/sqlite3/lib.c. It definitely depends on O level

O1 15s
O2 42s
O3 55s

It is 200K lines file. Maybe the issue is that clang is trying to perform some global optimisations that go exponential with the file size?

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 20:16:03 GMT
View Forum Message <> Reply to Message

I have just checked linux clang and gcc (different machine).

Linux clang compiles sqlite3 lib.c in 50s, gcc in 30s.

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 20:40:01 GMT
View Forum Message <> Reply to Message

I managed to compile Eigen_demo in Debug ...

$ wine umk reference Eigen_demo CLANGx64 -bu
----- Core (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (1 / 4)
----- plugin/Eigen (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (2 / 4)
----- plugin/z (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (3 / 4)
----- Eigen_demo (MAIN CLANG DEBUG DEBUG_FULL BLITZ WIN32) (4 / 4)
BLITZ: eigen_demo.cpp non-linear.cpp fft.cpp
Eigen_demo: 3 file(s) built in (37:40.15), 753384 msecs / file
Linking...
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\.cache\upp.out\CLANGx64.Debug.Debug_Full\Eigen_demo.exe (6637568 B) linked in (0:01.46)

OK. (37:42.11)

Less than 40 minutes

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 21:07:58 GMT
View Forum Message <> Reply to Message

It takes only 23 sec. to compile the same code in Linux using same version of Clang.
----- Eigen_demo (MAIN GCC DEBUG SHARED DEBUG_FULL BLITZ POSIX LINUX) (3 / 3)
BLITZ: eigen_demo.cpp non-linear.cpp fft.cpp
Core: 68 file(s) built in (0:02.42), 35 msecs / file
plugin/Eigen: 1 file(s) built in (0:03.34), 3346 msecs / file
Eigen_demo: 3 file(s) built in (0:22.95), 7651 msecs / file
Linking...
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/CLANG.Debug.Debug_Full.Shared/Eigen_demo (78994008 B) linked in (0:02.03)

OK. (0:31.83)

$ ./umk reference Eigen_demo CLANG -busa

IMHO, something isn't right with the Clang bundled with Upp.
It, probably, has been compiled differently ...

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 21:10:20 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 16:40I managed to compile Eigen_demo in Debug ...

$ wine umk reference Eigen_demo CLANGx64 -bu
----- Core (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (1 / 4)
----- plugin/Eigen (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (2 / 4)
----- plugin/z (CLANG DEBUG DEBUG_FULL BLITZ WIN32) (3 / 4)
----- Eigen_demo (MAIN CLANG DEBUG DEBUG_FULL BLITZ WIN32) (4 / 4)
BLITZ: eigen_demo.cpp non-linear.cpp fft.cpp
Eigen_demo: 3 file(s) built in (37:40.15), 753384 msecs / file
Linking...
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\.cache\upp.out\CLANGx64.Debug.Debug_Full\Eigen_demo.exe (6637568 B) linked in (0:01.46)

OK. (37:42.11)

Less than 40 minutes
Without BLITZ it takes only 20 minutes
----- Eigen_demo (MAIN CLANG DEBUG DEBUG_FULL WIN32) (4 / 4)
eigen_demo.cpp
non-linear.cpp
fft.cpp
Eigen_demo: 3 file(s) built in (20:14.75), 404918 msecs / file
Linking...
Z:\home\ssg\.local\soft\bb-worker\worker\wine-upp\build\.cache\upp.out\CLANGx64.Debug.Debug_Full.Noblitz\Eigen_demo.exe (6253568 B) linked in (0:01.40)

OK. (21:47.23)

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 21:27:33 GMT
View Forum Message <> Reply to Message

I'm afraid it is wine to blame in my case.
Clang-10 spends ~75% of its time calling one function in ntdll.dll.so

---

Subject: Re: 2020.1 officially released
Posted by Novo on Mon, 01 Jun 2020 21:31:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 15:35I have no idea where flagCLR is coming from ...
wine umk tutorial Gui01 CLANGx64cpp17 -bu

It would be great to get this fixed ...

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 21:32:42 GMT
View Forum Message <> Reply to Message

Maybe you can try cross-compilation.

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

If you do not have ubuntu 18.04, building llvm-mingw for you linux distro should be relatively simple....

It could be interesting experiment. Actually, we can even add a simple support in ide / umake to prepend binary with "wine" for execution...

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 01 Jun 2020 21:34:51 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 23:31Novo wrote on Mon, 01 June 2020 15:35I have no idea where flagCLR is coming from ...
wine umk tutorial Gui01 CLANGx64cpp17 -bu

It would be great to get this fixed ...

I will ASAP, but not today. For now in RM. If it is not fixed by the end of week, please ping me.

Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 02 Jun 2020 08:31:05 GMT
View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 23:27I'm afraid it is wine to blame in my case.
Clang-10 spends ~75% of its time calling one function in ntdll.dll.so

BTW, which function is that? Maybe there could be done something about it...

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 14:24:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 02 June 2020 04:31Novo wrote on Mon, 01 June 2020 23:27I'm afraid it is wine to blame in my case.
Clang-10 spends ~75% of its time calling one function in ntdll.dll.so

BTW, which function is that? Maybe there could be done something about it...

Mirek
I cannot tell that. There is no debug information in wine.
What I can see is just an address.

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 14:35:11 GMT
View Forum Message <> Reply to Message

mirek wrote on Mon, 01 June 2020 17:32Maybe you can try cross-compilation.

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

If you do not have ubuntu 18.04, building llvm-mingw for you linux distro should be relatively simple....

It could be interesting experiment. Actually, we can even add a simple support in ide / umake to prepend binary with "wine" for execution...

Mirek
First attempt is unsuccessful.
umk completely ignores all my attempts to set path to a different clang version.
I changed PATH and even put absolute path to clang into COMPILER.
Everything was ignored.

BTW, you put in PATH folders with dll-s ... What is that for?

Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 02 Jun 2020 15:40:12 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 02 June 2020 16:35mirek wrote on Mon, 01 June 2020 17:32Maybe you can try cross-compilation.

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

If you do not have ubuntu 18.04, building llvm-mingw for you linux distro should be relatively simple....

It could be interesting experiment. Actually, we can even add a simple support in ide / umake to prepend binary with "wine" for execution...

Mirek
First attempt is unsuccessful.
umk completely ignores all my attempts to set path to a different clang version.
I changed PATH and even put absolute path to clang into COMPILER.
Everything was ignored.

Can you post .bm?

Quote:BTW, you put in PATH folders with dll-s ... What is that for?

Well, thats in case you somehow link with shared (.dll) standard library. PATH is used to running .exe from the ide as well. Also mysql, sdl and pgsql dlls.

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 16:56:03 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 02 June 2020 11:40Can you post .bm?

I fixed problem with clang++ by using x86_64-w64-mingw32-clang++ instead of it. Before that I was trying to use /home/ssg/.local/soft/llvm-mingw/llvm-mingw-20200325-ubuntu-18.04/bin/clang++ as a compiler ... Plain clang++ didn't work by definition.

A new problem is that umk is passing POSIX and LINUX flags to x86_64-w64-mingw32-clang++ because it runs on Linux ...
Basically, this makes cross-compilation impossible with the current umk ...

BTW, Upp is currently broken ...
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/Ops.h:165:29: error: use of undeclared identifier 'HASH64_CONST3'
return (dword)SwapEndian64(HASH64_CONST3 * h);

File Attachments

1) CLANG.bm, downloaded 347 times

Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 02 Jun 2020 19:07:11 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 02 June 2020 18:56mirek wrote on Tue, 02 June 2020 11:40Can you post .bm?

I fixed problem with clang++ by using x86_64-w64-mingw32-clang++ instead of it. Before that I was trying to use /home/ssg/.local/soft/llvm-mingw/llvm-mingw-20200325-ubuntu-18.04/bin/clang++ as a compiler ... Plain clang++ didn't work by definition.

A new problem is that umk is passing POSIX and LINUX flags to x86_64-w64-mingw32-clang++ because it runs on Linux ...
Basically, this makes cross-compilation impossible with the current umk ...

Hm, we probably need to add flags override to .bm, right?

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 19:20:17 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 02 June 2020 15:07Novo wrote on Tue, 02 June 2020 18:56mirek wrote on Tue, 02 June 2020 11:40Can you post .bm?

I fixed problem with clang++ by using x86_64-w64-mingw32-clang++ instead of it. Before that I was trying to use /home/ssg/.local/soft/llvm-mingw/llvm-mingw-20200325-ubuntu-18.04/bin/clang++ as a compiler ... Plain clang++ didn't work by definition.

A new problem is that umk is passing POSIX and LINUX flags to x86_64-w64-mingw32-clang++ because it runs on Linux ...
Basically, this makes cross-compilation impossible with the current umk ...

Hm, we probably need to add flags override to .bm, right?

IMHO, we need an option for umk. Something like "target platform", "-t", I guess. Possible values should be "windows", "linux", "mac", "native", e.t.c.

Another useful option for umk on Windows is "use POSIX-like configuration files". I'm already doing this, but I manually change source code for that.

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 19:26:01 GMT

View Forum Message <> Reply to Message

This is how a command line looks at the moment ...
x86_64-w64-mingw32-clang++ -c `pkg-config --cflags freetype2` `pkg-config --cflags x11` `pkg-config --cflags xinerama` `pkg-config --cflags xrender` `pkg-config --cflags xft` `pkg-config --cflags xdmcp` `pkg-config --cflags fontconfig` `pkg-config --cflags xcb` `pkg-config --cflags xext` `pkg-config --cflags gtk+-3.0` `pkg-config --cflags libnotify` `pkg-config --cflags expat` `pkg-config --cflags libpng` -I"/home/ssg/.local/soft/bb-worker/worker/mingw-sdb/build" -I"/home/ssg/dvlp/cpp/code/upp/git/bazaar" -I"/home/ssg/dvlp/cpp/code/upp/git/uppsrc" -I"/home/ssg/.wine/drive_c/local/apps/upp/bin/SDL2/include" -I"/home/ssg/.wine/drive_c/local/apps/upp/bin/pgsql/x64/include" -I"/home/ssg/.wine/drive_c/local/apps/upp/bin/mysql/include" -I"/home/ssg/.local/soft/bb-worker/worker/mingw-sdb/build/dvlp/plugin/sol3" -I"/home/ssg/.local/soft/bb-worker/worker/mingw-sdb/build/dvlp/plugin/lua/lib" -I"/home/ssg/.local/soft/bb-worker/worker/mingw-sdb/build/.cache/upp.out/Painter/CLANGcpp17.Debug.Debug_Full.Gui.Shared" -DflagGUI -DflagCLANG -DflagDEBUG -DflagSHARED -DflagDEBUG_FULL -DflagBLITZ -DflagPOSIX -DflagLINUX -ggdb -g2 -fexceptions -D_DEBUG -O0 -x c++ -Wall -Wno-logical-op-parentheses -std=c++17 "/home/ssg/dvlp/cpp/code/upp/git/uppsrc/Painter/SvgUtil.cpp" -o "/home/ssg/.local/soft/bb-worker/worker/mingw-sdb/build/.cache/upp.out/Painter/CLANGcpp17.Debug.Debug_Full.Gui.Shared/SvgUtil.o"

Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 02 Jun 2020 19:27:03 GMT
View Forum Message <> Reply to Message

Novo wrote on Tue, 02 June 2020 21:20mirek wrote on Tue, 02 June 2020 15:07Novo wrote on Tue, 02 June 2020 18:56mirek wrote on Tue, 02 June 2020 11:40Can you post .bm?

I fixed problem with clang++ by using x86_64-w64-mingw32-clang++ instead of it. Before that I was trying to use /home/ssg/.local/soft/llvm-mingw/llvm-mingw-20200325-ubuntu-18.04/bin/clang++ as a compiler ... Plain clang++ didn't work by definition.

A new problem is that umk is passing POSIX and LINUX flags to x86_64-w64-mingw32-clang++ because it runs on Linux ...
Basically, this makes cross-compilation impossible with the current umk ...

Hm, we probably need to add flags override to .bm, right?

IMHO, we need an option for umk. Something like "target platform", "-t", I guess. Possible values should be "windows", "linux", "mac", "native", e.t.c.

Another useful option for umk on Windows is "use POSIX-like configuration files". I'm already doing this, but I manually change source code for that.

Well, we already can add flags via "Common fixed flags". I think all we need is "override" option that would avoid adding platform flags before adding these fixed. And then of replace any PLATFORM_ #ifdefs in GCC builder with if(HasFlag(...

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 19:33:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 02 June 2020 15:27
Well, we already can add flags via "Common fixed flags". I think all we need is "override" option that would avoid adding platform flags before adding these fixed. And then of replace any PLATFORM_ #ifdefs in GCC builder with if(HasFlag(...
IMHO, that won't affect stuff like `pkg-config --cflags x11`
This is done by umk...

Edited: Or I'm missing something.

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 19:39:44 GMT
View Forum Message <> Reply to Message

Another thing. umk is ignoring options from bm-files.
Even if I enable BLITZ in a bm-file, I still have to pass "-b" to umk...

Edited: I need to double-check that. Probably, I've got lost among ide, theide, and umk configuration folders ...
Edited again: Yes, umk ignores settings from bm-files.

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 02 Jun 2020 23:30:08 GMT
View Forum Message <> Reply to Message

reference/GLDrawDemo is still broken.
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/GLDraw/GLShaders.cpp:30:3: error: expected expression
DLOG(error);
^

Subject: Re: 2020.1 officially released
Posted by Sender Ghost on Wed, 03 Jun 2020 05:37:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 02 June 2020 19:27
Well, we already can add flags via "Common fixed flags". I think all we need is "override" option that would avoid adding platform flags before adding these fixed.
As far as I know, there is a possibility to override platform related flags. This feature is implemented in LocalHost::AddFlags and LocalHost::HasPlatformFlag methods [1, 2, 3].
For example, possible to generate Makefile(s) for WIN32 platform, e.g.:
./umk uppsrc ide CLANG -r -M=Makefile +WIN32,GUI
or FreeBSD:
./umk uppsrc ide CLANG -rs -M=BSDMakefile +POSIX,BSD,FREEBSD,GUI
or Linux:
./umk uppsrc ide CLANG -rs -M=GNUMakefile +POSIX,LINUX,GUI
etc.

There is just a need to use correct build method for them. But I didn't test this for cross-platform/remote build(s), just for Makefile generation.

May suggest to add "OSX" flag to platformFlags also (based on order of appearance in LocalHost::AddFlags method). Patch attached.

File Attachments

1) uppsrc_ide_Core_Host_r14539.diff, downloaded 342 times

Subject: Re: 2020.1 officially released
Posted by Tom1 on Wed, 03 Jun 2020 07:28:26 GMT
View Forum Message <> Reply to Message

Quote:
IMHO, we need an option for umk. Something like "target platform", "-t", I guess. Possible values should be "windows", "linux", "mac", "native", e.t.c.
Hi,

This idea sounds very interesting. Especially if this could be possible to extend to TheIDE on Windows. Then it would be possible to build with CLANG on Windows directly for Raspberry Pi. This would make developing for RPi so much more convenient!

Best regards,

Tom

Subject: Re: 2020.1 officially released
Posted by mirek on Wed, 03 Jun 2020 10:46:19 GMT
View Forum Message <> Reply to Message

Sender Ghost wrote on Wed, 03 June 2020 07:37mirek wrote on Tue, 02 June 2020 19:27
Well, we already can add flags via "Common fixed flags". I think all we need is "override" option

that would avoid adding platform flags before adding these fixed.
As far as I know, there is a possibility to override platform related flags. This feature is implemented in LocalHost::AddFlags and LocalHost::HasPlatformFlag methods [1, 2, 3].
For example, possible to generate Makefile(s) for WIN32 platform, e.g.:
./umk uppsrc ide CLANG -r -M=Makefile +WIN32,GUI
or FreeBSD:
./umk uppsrc ide CLANG -rs -M=BSDMakefile +POSIX,BSD,FREEBSD,GUI
or Linux:
./umk uppsrc ide CLANG -rs -M=GNUMakefile +POSIX,LINUX,GUI
etc.

There is just a need to use correct build method for them. But I didn't test this for cross-platform/remote build(s), just for Makefile generation.

May suggest to add "OSX" flag to platformFlags also (based on order of appearance in LocalHost::AddFlags method). Patch attached.

Yes, but that ADDs flags. The problem is that there are default, platform defined flags too. We need a method to clear those...

Well, radical solution would be to remove them completely and move them strictly to build method. Less radical is to add an option "No default flags" to build method.

Second problem is that there are some #ifdef PLATFORM_ conditional compilation flags in GCC builder. Those need to be replaced to actually react to current flag configuration.

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Wed, 03 Jun 2020 11:51:57 GMT
View Forum Message <> Reply to Message

Novo wrote on Wed, 03 June 2020 01:30reference/GLDrawDemo is still broken.
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/GLDraw/GLShaders.cpp:30:3: error: expected expression
DLOG(error);
^

Thanks, fixed.

Subject: Re: 2020.1 officially released
Posted by Novo on Wed, 03 Jun 2020 14:16:16 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 03 June 2020 07:51Novo wrote on Wed, 03 June 2020

01:30reference/GLDrawDemo is still broken.
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/GLDraw/GLShaders.cpp:30:3: error: expected expression
DLOG(error);
^

Thanks, fixed.
This fix didn't get propagated into git ...

Subject: Re: 2020.1 officially released
Posted by Sender Ghost on Thu, 04 Jun 2020 03:16:43 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 03 June 2020 10:46
Yes, but that ADDs flags. The problem is that there are default, platform defined flags too. We need a method to clear those...

Actually, this is what it does before adding default platform flags, defined by #if(def)(s), if some platform flag was found for override:

```
if(HasPlatformFlag(cfg))
 return;
```

Novo wrote on Tue, 02 June 2020 19:20
IMHO, we need an option for umk. Something like "target platform", "-t", I guess.

Thanks for suggestion. I proposed this feature on Redmine #2041.

Subject: Re: 2020.1 officially released
Posted by mirek on Thu, 04 Jun 2020 08:31:37 GMT
View Forum Message <> Reply to Message

I have removed all #ifdef PLATFORM_* from GCC builder (except one case where this was not possible, but that one should not affect POSIX -> WIN32 compilation).

As for target option, I do not think it is necessarry (as corretly pointed out by SenderGhost). As long as you specify any platform flag in commandline, the build process will avoid adding these, so you should be OK.

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Thu, 04 Jun 2020 08:37:09 GMT

View Forum Message <> Reply to Message

Novo wrote on Mon, 01 June 2020 21:35There is another unexpected problem with Clang on Windows.
If I add "-std=c++17" to cpp options, then in case of GUI app (example: tutorial/Gui01) in Debug configuration I get this (Release is fine):
[code]Linking...
lld-link: error: duplicate symbol: std::__throw_bad_alloc()
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:315
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debu g_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

lld-link: error: duplicate symbol: operator new(unsigned long long)
>>> defined at c:\local\apps\upp\uppsrc\Core\heap.cpp:301
>>> c:/local/apps/upp/out/tutorial/Core/CLANGx64cpp17.Debug.Debu g_Full.Gui\heap.o
>>> defined at libc++.a(new.cpp.obj)

I have checked this one and unfortunately it seems like another instance of weak symbol bug

https://github.com/mstorsjo/llvm-mingw/issues/91

so the options are:

- USEMALLOC flag (this will switch off U++ allocator completely)
- STD_NEWDELETE flag (this will keep U++ allocator and use it where possible but will use standard new/delete - this is default on MacOS BTW)
- Try with GIT head, as Martin claims this is now fixed. This would be most interesting option for me (Being there, you could check msvcrt version too).

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Thu, 04 Jun 2020 11:52:07 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 04 June 2020 04:31I have removed all #ifdef PLATFORM_* from GCC builder (except one case where this was not possible, but that one should not affect POSIX -> WIN32 compilation).

Thanks.
Unfortunately, your last commit broke linking of Linux console apps.
Linking...
/usr/bin/ld:
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/ide/Builders/CLANG.Debug.Debug_Full.Noblitz.Shared/Builders.a(Cocoa.o): in function `GccBuilder::CocoaAppBundle()':
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:18: undefined

reference to `Upp::StreamRaster::LoadFileAny(char const*, Upp::Function<bool (int, int)>)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:26: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:26: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:48: undefined reference to `Upp::Image::Image(Upp::Image const&)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::PNGEncoder::PNGEncoder(int, Upp::ImageKind, bool)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::Rescale(Upp::Image const&, int, int, Upp::Function<bool (int, int)>)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::StreamRasterEncoder::SaveFile(char const*, Upp::Image const&)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::PNGEncoder::~PNGEncoder()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:52: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:51: undefined reference to `Upp::PNGEncoder::~PNGEncoder()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/ide/Builders/Cocoa.cpp:52: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/ide/Builders/CLANG.Debug.Debug_Full.Noblitz.Shared/Builders.a(Cocoa.o): in function `PNGRaster__initialize_struct':
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/plugin/png/png.h:8: undefined reference to `Upp::PNGRaster__initializer()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/ide/Builders/CLANG.Debug.Debug_Full.Noblitz.Shared/Builders.a(Cocoa.o): in function `Upp::SortedVectorMap<int, Upp::Image, Upp::StdLess<int> >::Add(int const&, Upp::Image const&)':
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/InVector.h:650: undefined reference to `Upp::Image::operator=(Upp::Image const&)'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/InVector.h:650: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/InVector.h:650: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/ide/Builders/CLANG.Debug.Debug_Full.Noblitz.Shared/Builders.a(Cocoa.o): in function `void Upp::Destroy<Upp::Image>(Upp::Image*, Upp::Image const*)':
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/Vcont.h:28: undefined reference to `Upp::Image::~Image()'
/usr/bin/ld: /home/ssg/.local/soft/bb-worker/worker/l-upp/build/.cache/upp.out/ide/Builders/CLANG.Debug.Deb

ug_Full.Noblitz.Shared/Builders.a(Cocoa.o): in function `Upp::Image Upp::clone<Upp::Image>(Upp::Image const&)':
/home/ssg/.local/soft/bb-worker/worker/l-upp/build/uppsrc/Core/Defs.h:398: undefined reference to `Upp::Image::Image(Upp::Image const&)'

Basically, umk cannot be built either using a Makefile or umk itself ...

Subject: Re: 2020.1 officially released
Posted by mirek on Thu, 04 Jun 2020 12:24:25 GMT
View Forum Message <> Reply to Message

Sorry, fixed at the price of another #ifdef, which will mean you cannot build OSX GUI appliaction in LINUX (which is impossible anyway).

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Thu, 04 Jun 2020 14:00:58 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 04 June 2020 08:24Sorry, fixed at the price of another #ifdef, which will mean you cannot build OSX GUI appliaction in LINUX (which is impossible anyway).

Mirek
Thanks!

Subject: Re: 2020.1 officially released
Posted by Novo on Thu, 04 Jun 2020 15:35:27 GMT
View Forum Message <> Reply to Message

A minor issue with mingw on Linux.
umk calls x86_64-w64-mingw32-windres.exe, which is called x86_64-w64-mingw32-windres (no extension) on Linux, otherwise +WIN32 seems to work ...

Subject: Re: 2020.1 officially released
Posted by Novo on Thu, 04 Jun 2020 19:14:00 GMT
View Forum Message <> Reply to Message

Another interesting observation. Size of executable.

mingw:

:~/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/CLANG.Blitz.Gui.Win32$ ll AK.exe

-rwx------ 1 ssg ssg 2541056 Jun 4 14:56 AK.exe*

wine:
:~/.local/soft/bb-worker/worker/wine-upp/build/.cache/upp.out/CLANGx64.Blitz.Gui$ ll AK.exe
-rwxrwxr-x 1 ssg ssg 5104128 Jun 4 14:28 AK.exe*

Two times bigger.
Both are 64-bit executables.

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 02:59:22 GMT
View Forum Message <> Reply to Message

Problem with mingw & linking when compiling reference/GuiWebCrawler.
Command line contains "-lcrypto -lssl".
Linker is looking for libcrypto.a & libssl.a, which are named crypto.lib & ssl.lib.
Even after creating of links with correct names linker is still looking for libLIBCMT.a & libOLDNAMES.a, which are not available ...

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 03:03:04 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 04 June 2020 11:35A minor issue with mingw on Linux.
umk calls x86_64-w64-mingw32-windres.exe, which is called x86_64-w64-mingw32-windres (no extension) on Linux, otherwise +WIN32 seems to work ...
Could you please fix that?
You just need to replace
String windres = "windres.exe";
with
String windres = "windres";

TIA

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 03:44:17 GMT
View Forum Message <> Reply to Message

Compilation speed of mingw is good.
Eigen_demo compiles in 35 sec.

This is Debug:
$ ./umk reference Eigen_demo CLANG -bu
...
----- Eigen_demo (WIN32 MAIN CLANG DEBUG DEBUG_FULL BLITZ) (4 / 4)
BLITZ: eigen_demo.cpp non-linear.cpp fft.cpp
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/Eigen_demo/CLANG.Debug.Debug_Full.Main.Win32/Eigen_demo$blitz.cpp:3:
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/reference/Eigen_demo/eigen_demo.cpp:1:
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/uppsrc/Core/Core.h:281:
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/uppsrc/Core/Mem.h:342:8: warning: unused variable 'Cmp128' [-Wunused-variable]
auto Cmp128 = [&](size_t at) { return _mm_cmpeq_epi32(_mm_loadu_si128((__m128i *)(s + at)), _mm_loadu_si128((__m128i *)(t + at))); };
^
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/Eigen_demo/CLANG.Debug.Debug_Full.Main.Win32/Eigen_demo$blitz.cpp:3:
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/reference/Eigen_demo/eigen_demo.cpp:1:
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/uppsrc/Core/Core.h:285:
In file included from
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/uppsrc/Core/String.h:956:
/home/ssg/.local/soft/bb-worker/worker/mingw-upp/build/uppsrc/Core/AString.hpp:270:14: warning: unused variable 't' [-Wunused-variable]
const char *t = begin();
^
2 warnings generated.
Eigen_demo: 3 file(s) built in (0:34.91), 11637 msecs / file

Release takes 45 sec.
Mission is acomplished

Subject: Re: 2020.1 officially released
Posted by mirek on Fri, 05 Jun 2020 07:34:18 GMT
View Forum Message <> Reply to Message

Novo wrote on Thu, 04 June 2020 21:14Another interesting observation. Size of executable.

mingw:
:~/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/CLANG.Blitz.Gui.Win32$ ll AK.exe

-rwx------ 1 ssg ssg 2541056 Jun 4 14:56 AK.exe*

Not sure what is "mingw" here... Do you mean llvm-mingw toolchain in crosscompile mode?

If yes, could you activate Verbose mode and compare all compiler/linker options?

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Fri, 05 Jun 2020 07:37:30 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 05 June 2020 05:03Novo wrote on Thu, 04 June 2020 11:35A minor issue with mingw on Linux.
umk calls x86_64-w64-mingw32-windres.exe, which is called x86_64-w64-mingw32-windres (no extension) on Linux, otherwise +WIN32 seems to work ...
Could you please fix that?
You just need to replace
String windres = "windres.exe";
with
String windres = "windres";

TIA

I will, but it is not that easy - if I do above, it will stop working in Win32. I can add #ifdef PLATFORM_ perhaps, or I can try to test which variant exists...

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Fri, 05 Jun 2020 07:44:47 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 05 June 2020 04:59Problem with mingw & linking when compiling reference/GuiWebCrawler.
Command line contains "-lcrypto -lssl".
Linker is looking for libcrypto.a & libssl.a, which are named crypto.lib & ssl.lib.

Can you try to add .lib in package organizer? Just to know it that helps.

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 15:16:39 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 05 June 2020 03:34Novo wrote on Thu, 04 June 2020 21:14Another interesting observation. Size of executable.

mingw:
:~/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/CLANG.Blitz.Gui.Win32$ ll AK.exe

-rwx------ 1 ssg ssg 2541056 Jun 4 14:56 AK.exe*

Not sure what is "mingw" here... Do you mean llvm-mingw toolchain in crosscompile mode?

If yes, could you activate Verbose mode and compare all compiler/linker options?

Mirek

Yes, "mingw" is llvm-mingw toolchain in cross-compile mode.
Please check attached file.
txt-files is what you asked for.
log-files - complete build logs.

File Attachments
1) 01.zip, downloaded 285 times

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 18:09:30 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 05 June 2020 03:44Novo wrote on Fri, 05 June 2020 04:59Problem with mingw & linking when compiling reference/GuiWebCrawler.
Command line contains "-lcrypto -lssl".
Linker is looking for libcrypto.a & libssl.a, which are named crypto.lib & ssl.lib.

Can you try to add .lib in package organizer? Just to know it that helps.

Adding .lib to crypto & ssl doesn't affect anything.

BTW, precompiled ssl is already included into Clang distribution bundled with Upp.
And, I guess, Clang is using its own ssl ...
And one shipped with Upp seems to be compiled with msvc ...

Another thing. llvm-mingw-20200325-ubuntu-18.04 doesn't include a prebuilt version of ssl ...
This is weird ...

Subject: Re: 2020.1 officially released
Posted by luoganda on Fri, 05 Jun 2020 20:24:10 GMT
View Forum Message <> Reply to Message

reference/GLDrawDemo doesn't work properly..
Tested in kubuntu 18.04, without changing anything, that is: GUI.NOGTK, GCC Debug.

App compiles ok, but when it's run - window is ok - but where GLCtrl is - nothing is drawn.
And when app is closed, there comes a dialog with 'Fatal error:' and message 'Heap leaks detected!'

Subject: Re: 2020.1 officially released
Posted by Novo on Fri, 05 Jun 2020 20:24:41 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 04 June 2020 04:37
- Try with GIT head, as Martin claims this is now fixed. This would be most interesting option for me (Being there, you could check msvcrt version too).

Mirek
Checked against GIT head. The problem is still there.

What do you mean by msvcrt version?

Subject: Re: 2020.1 officially released
Posted by mirek on Sat, 06 Jun 2020 08:26:11 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 05 June 2020 20:09mirek wrote on Fri, 05 June 2020 03:44Novo wrote on Fri, 05 June 2020 04:59Problem with mingw & linking when compiling reference/GuiWebCrawler.
Command line contains "-lcrypto -lssl".
Linker is looking for libcrypto.a & libssl.a, which are named crypto.lib & ssl.lib.

Can you try to add .lib in package organizer? Just to know it that helps.

Adding .lib to crypto & ssl doesn't affect anything.

BTW, precompiled ssl is already included into Clang distribution bundled with Upp.

Yes, but that one does not work with MSVC...

Frankly, it is dancing between blades.... I was really happy to find working solution for genuine windows for all small issues involved. It is really stupid that it has wine problem.

Mirek

Subject: Re: 2020.1 officially released
Posted by mirek on Sat, 06 Jun 2020 08:30:51 GMT
View Forum Message <> Reply to Message

Novo wrote on Fri, 05 June 2020 22:24mirek wrote on Thu, 04 June 2020 04:37
- Try with GIT head, as Martin claims this is now fixed. This would be most interesting option for me (Being there, you could check msvcrt version too).

Mirek
Checked against GIT head. The problem is still there.

Just to be sure: I mean llvm-mingw head (not U++)....

Quote:
What do you mean by msvcrt version?

By default, llvm-mingw builds for universal CRT, which is the new Windows OS C library in Win10 (and via system updates, since Win7). There is older one, "msvcrt.dll" which is present sice WinNT. However, situation is more complicated:

https://github.com/mstorsjo/llvm-mingw/issues/104

Still, it would probably be better to ship with msvcrt version... maybe next release.

Mirek

Subject: Re: 2020.1 officially released
Posted by Novo on Sun, 07 Jun 2020 13:22:41 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 06 June 2020 04:30Novo wrote on Fri, 05 June 2020 22:24mirek wrote on Thu, 04 June 2020 04:37
- Try with GIT head, as Martin claims this is now fixed. This would be most interesting option for me (Being there, you could check msvcrt version too).

Mirek

Checked against GIT head. The problem is still there.

Just to be sure: I mean llvm-mingw head (not U++)....

Yes, I've built llvm-mingw from git. This is easy. Just one simple command.
I'm still getting the same "duplicated symbols" error messages.

Subject: Re: 2020.1 officially released
Posted by Novo on Sun, 07 Jun 2020 13:45:44 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 06 June 2020 04:30
By default, llvm-mingw builds for universal CRT, which is the new Windows OS C library in Win10 (and via system updates, since Win7). There is older one, "msvcrt.dll" which is present sice WinNT. However, situation is more complicated:

https://github.com/mstorsjo/llvm-mingw/issues/104

Still, it would probably be better to ship with msvcrt version... maybe next release.

Mirek
I'm a little bit confused. I thought that x86_64-w64-mingw32-clang++ builds for msvcrt, and x86_64-w64-mingw32uwp-clang++ builds for ucrt.

Anyway, I'll try to recompile llvm-mingw with --with-default-msvcrt=msvcrt on the build-mingw-w64.sh line.

Subject: Re: 2020.1 officially released
Posted by Novo on Sun, 07 Jun 2020 14:08:22 GMT
View Forum Message <> Reply to Message

Novo wrote on Sun, 07 June 2020 09:45
Anyway, I'll try to recompile llvm-mingw with --with-default-msvcrt=msvcrt on the build-mingw-w64.sh line.
That didn't help. I'm still getting same errors.
The only solution so far is to use the USEMALLOC flag.

Subject: Re: 2020.1 officially released
Posted by Novo on Sun, 07 Jun 2020 14:52:56 GMT
View Forum Message <> Reply to Message

A minor issue with umk.
It doesn't check for duplicate flags.

I have USEMALLOC defined in a project file and in a bm-file. In the end I get this:
CLANGx64cpp17.Debug.Debug_Full.Gui.Usemalloc.Usemalloc

Subject: Re: 2020.1 officially released
Posted by Novo on Sun, 07 Jun 2020 15:41:03 GMT
View Forum Message <> Reply to Message

mirek wrote on Sat, 06 June 2020 04:26
Frankly, it is dancing between blades.... I was really happy to find working solution for genuine windows for all small issues involved. It is really stupid that it has wine problem.

Mirek
IMHO, the "wine problem" is not a problem anymore.
Wine is not needed anymore because we have cross-compilation now. (It would be interesting to check Upp against a a standard gcc-based mingw)

IMHO, a real problem is a problem with new/delete and cpp17. But even this problem has a workaround ...

Life is good

Subject: Re: 2020.1 officially released
Posted by Novo on Tue, 16 Jun 2020 17:07:10 GMT
View Forum Message <> Reply to Message

mirek wrote on Fri, 05 June 2020 03:34Novo wrote on Thu, 04 June 2020 21:14Another interesting observation. Size of executable.

mingw:
:~/.local/soft/bb-worker/worker/mingw-upp/build/.cache/upp.out/CLANG.Blitz.Gui.Win32$ ll AK.exe

-rwx------ 1 ssg ssg 2541056 Jun 4 14:56 AK.exe*

Not sure what is "mingw" here... Do you mean llvm-mingw toolchain in crosscompile mode?

If yes, could you activate Verbose mode and compare all compiler/linker options?

Mirek

In my "wine" configuration I was using settings taken from a bm-file shipped with Upp.
They do not look optimal to me now ...

Subject: Re: 2020.1 officially released: minor String bug?
Posted by luoganda on Thu, 18 Jun 2020 12:54:21 GMT
View Forum Message <> Reply to Message

It seems that there is one thing missing in String - one of Insert flavour, that is:
void Insert(int pos, const char *s, int count).
Hmm - has no one ever used this func . Maybe i am wrong about this - maybe this was fixed, although when i used it - it didn't compiled correctly - i think count and s were reversed. Just to check..

This compiled on one not so old revision, meaning one wrote code that was meant in one way but compiled ok in wrong way.

WString has this: void Insert(int pos, const wchar *s, int count), it seems String does not.

Subject: Re: 2020.1 officially released: minor String bug?
Posted by mirek on Thu, 18 Jun 2020 13:00:56 GMT
View Forum Message <> Reply to Message

luoganda wrote on Thu, 18 June 2020 14:54It seems that there is one thing missing in String - one of Insert flavour, that is:
void Insert(int pos, const char *s, int count).
Hmm - has no one ever used this func . Maybe i am wrong about this - maybe this was fixed, although when i used it - it didn't compiled correctly - i think count and s were reversed. Just to check..

This compiled on one not so old revision, meaning one wrote code that was meant in one way but compiled ok in wrong way.

WString has this: void Insert(int pos, const wchar *s, int count), it seems String does not.

This works:

```
CONSOLE_APP_MAIN
{
 String x = "1234";
 x.Insert(1, "ABCD", 2);
 DDUMP(x);
}
```

are you sure?

Subject: Re: 2020.1 officially released

Posted by Giorgio on Tue, 07 Jul 2020 07:43:48 GMT
View Forum Message <> Reply to Message

Hi there,
thanks for this new release.

I'm trying to compile an application of mine with the latest version of Visual Studio that compiles fine with 2019.2. I tried with MSVS19x64, MSVS19, MSVS15x64 (debug and release mode)on Windows 10and I got the following error: "c2872: 'Ref': ambiguous symbol". It appears in the .sch file, each time the keyword "TABLE_" appears.

Anyone experienced the same problem?

Tahnks,
gio

Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 07 Jul 2020 09:08:22 GMT
View Forum Message <> Reply to Message

No and I am doing a lot of .sch.

I guess this is some corner case, could you post some testcase? I guess your current .sch file and c++ that you are using to include it would be fine.

Subject: Re: 2020.1 officially released
Posted by Giorgio on Wed, 08 Jul 2020 09:50:35 GMT
View Forum Message <> Reply to Message

Hi,
the application I developed it's quite structured: there are about 15 different packages (each package access some tables of a PostgreSQL DB) and one main "wrapping" packages that contains the main menu of the application. The app is composed by the "wrapping" package (always): other packages are included or not in the application during compilation (depending on the user I'm compiling for). If I create a test case stripping down the application to the bare minimum (only the "wrapping" package with the .sch file) it compiles without problems.

Each package includes a .h file as follows:

#include <PostgreSQL/PostgreSQL.h>
#define SCHEMADIALECT <PostgreSQL/PostgreSQLSchema.h>
#define MODEL "path/to/file.sch"

If I do not include it I got compilation errors (basically the compiler does not recognize the name of

the SqlIds).

So basically, I think that I use the wrong method to include the schema file in the packages: what should be the right way to do that?

Subject: Re: 2020.1 officially released
Posted by mirek on Wed, 08 Jul 2020 15:45:44 GMT
View Forum Message <> Reply to Message

Well, thinking about it, I would begin with what it says: Ref being ambiguous.

Do you define anything named "Ref" in your application? Do you have any namespaces and/or using namespace ?

Mirek

Subject: Re: 2020.1 officially released
Posted by Giorgio on Thu, 09 Jul 2020 12:54:41 GMT
View Forum Message <> Reply to Message

Hi Mirek,
I do use namespace in my application, but I do not use something called 'Ref'.

The error of the compiler actually says that the ambiguity is between 'Upp::Ref' or 'Eigen::Ref'.

By the way, I think I have fixed the problem. In my application the include of the SCHEMA file and related headers (ref. https://www.ultimatepp.org/srcdocSqlBasicUse_en-us.html) was very messy: there was an include file who collected some other headers, but in some case this was included in a third header. I didn't care very much till today, 'cause everything was working. So I cleaned up the mess and it came up that the problem was the following line:

#include <ScatterCtrl/ScatterCtrl.h>

If I include the above header before the header related to the schema file, I got the error, if I put the schema before and then the ScatterDraw it works fine.

Bye,
gio

Subject: Re: 2020.1 officially released
Posted by mirek on Mon, 13 Jul 2020 08:28:15 GMT

Giorgio wrote on Thu, 09 July 2020 14:54Hi Mirek,
I do use namespace in my application, but I do not use something called 'Ref'.

The error of the compiler actually says that the ambiguity is between 'Upp::Ref' or 'Eigen::Ref'.

By the way, I think I have fixed the problem. In my application the include of the SCHEMA file and related headers (ref. https://www.ultimatepp.org/srcdocSqlBasicUse_en-us.html) was very messy: there was an include file who collected some other headers, but in some case this was included in a third header. I didn't care very much till today, 'cause everything was working. So I cleaned up the mess and it came up that the problem was the following line:

#include <ScatterCtrl/ScatterCtrl.h>

If I include the above header before the header related to the schema file, I got the error, if I put the schema before and then the ScatterDraw it works fine.

Bye,
gio

Well, it looks like schema sources would be better with full qualification of ids (Upp::Ref). Will fix that for the next release.

Subject: Re: 2020.1 officially released
Posted by pvictor on Mon, 13 Jul 2020 16:51:10 GMT

Hello.

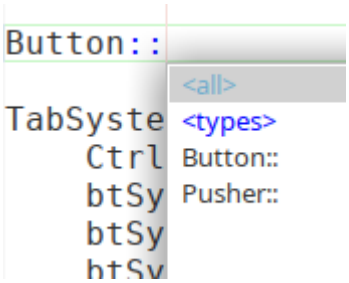
I noticed one strange thing: Assist doesn't show methods of Ctrl.
For example, when I type Ctrl:: - nothing appears,
and when I type Button:: - only methods of Button and Pusher appear:

Version: 14656 (64 bit) (GCC) (C++11) (C++14) (Gtk)

Best regards,
Victor

File Attachments

1) aaa.png, downloaded 700 times



Subject: Re: 2020.1 officially released
Posted by mirek on Tue, 14 Jul 2020 08:23:47 GMT
View Forum Message <> Reply to Message

pvictor wrote on Mon, 13 July 2020 18:51Hello.

I noticed one strange thing: Assist doesn't show methods of Ctrl.
For example, when I type Ctrl:: - nothing appears,
and when I type Button:: - only methods of Button and Pusher appear:

Version: 14656 (64 bit) (GCC) (C++11) (C++14) (Gtk)

Best regards,
Victor

This does not really belong into this thread, 14656 is not 2020.1...

This bug was already fixed in 14661.

Mirek
