Subject: U++ 2017 beta
Posted by mirek on Thu, 22 Dec 2016 08:25:46 GMT
View Forum Message <> Reply to Message

It is that time again... Took longer than expected as this ended as the most radical change to U++ in years.

http://www.ultimatepp.org/www$uppweb$Roadmap$en-us.html

Consider the current nightly build a beta. Only serious bugs will be fixed until the release (planned Jan 2).

Mirek

Subject: Re: U++ 2017 beta
Posted by MrSarup on Thu, 22 Dec 2016 19:25:36 GMT
View Forum Message <> Reply to Message

Hello Mirek,
mirek wrote on Thu, 22 December 2016 09:25Consider the current nightly build a beta. Only serious bugs will be fixed until the release (planned Jan 2).
Mirek
Thanks for your wonderful hard work, as well as contribution by others.

Somehow I do not see any point to distribute buggy codes in examples or Bazaar.

While you want to declare the nightly as beta, I would suggest to remove or deactivate all example source codes that does not get build. If they do not work, why have them in there? It can appear again if they are fixed by the authors.

As a new comer, I found it disappointing to see on one side that some of the examples provided in the nightly build did not get far enough to compile and on the other side so many complex examples are just missing. (I know that such expectations on an open source unpaid plattform are wrong...)

One of the brilliant aspect of U++ is to have a super fast and comfortable start. However, when a new comer like myself tries to work with the entire different basics of a language put "upside down", then it becomes difficult to catch up in the absence of some complex examples.

After learning to play a bit with Callbacks, I found that that just got depreciated! Grrrr... And then some examples did not get build. Grrrr.

And not many participates in the forum postings. Grrrr....

Subject: Re: U++ 2017 beta

Posted by mirek on Thu, 22 Dec 2016 22:00:53 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Thu, 22 December 2016 20:25Hello Mirek,
mirek wrote on Thu, 22 December 2016 09:25Consider the current nightly build a beta. Only serious bugs will be fixed until the release (planned Jan 2).
Mirek
Thanks for your wonderful hard work, as well as contribution by others.

Somehow I do not see any point to distribute buggy codes in examples or Bazaar.

While you want to declare the nightly as beta, I would suggest to remove or deactivate all example source codes that does not get build. If they do not work, why have them in there? It can appear again if they are fixed by the authors.

All examples are test-build each night (I mean stuff from 'example' and 'reference' folders/nests) and current test-suite does not show any problems. If there are any of them that do not build, please let me know, it must be some interesting quirk or platform incompatibility. (Please report your platform).

Bazaar is community submitted content, bar is lower. I am unhappy about status of many things there too, but not brave enought to start deleting things...

Quote:
After learning to play a bit with Callbacks, I found that that just got depreciated! Grrrr... And then some examples did not get build. Grrrr.

Yeah, but replacement is not that different in use. It was more C++11 compatibility issue than concept change. You came at the point when many things had to change, sorry about that.

Mirek

Subject: Re: U++ 2017 beta
Posted by mr_ped on Fri, 23 Dec 2016 03:02:07 GMT
View Forum Message <> Reply to Message

Sounds great. All of it. :d

Subject: Re: U++ 2017 beta
Posted by koldo on Fri, 23 Dec 2016 09:28:14 GMT
View Forum Message <> Reply to Message

Thank you Mirek.

Subject: Re: U++ 2017 beta
Posted by MrSarup on Sun, 25 Dec 2016 07:02:44 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Thanks for your answer.
mirek wrote on Thu, 22 December 2016 23:00
All examples are test-build each night (I mean stuff from 'example' and 'reference' folders/nests) and current test-suite does not show any problems. If there are any of them that do not build, please let me know, it must be some interesting quirk or platform incompatibility. (Please report your platform).
Mirek
I am working with U++ on Windows 7 Ultimate 64bits and Centos 7 64bits. Until now, I have build many on my windows workstation and found that some did not complete the build. Most likely they were not under 'example' but one or more under 'reference', however I cannot exactly remember. If this helps, I would be more than happy to execute these one again on my platform and report it here.

mirek wrote on Thu, 22 December 2016 23:00
Bazaar is community submitted content, bar is lower. I am unhappy about status of many things there too, but not brave enough to start deleting things...
Mirek
That is true. How about creating "beta-xxx" in the assembly, or any other connotation in the event if one package fails to be built. If the report is confirmed, then one could - instead of deleting it completely - move it under a beta assembly. For e.g. there is under the assembly:

Bazaar-Stable
Bazaar-Beta-Windows
Bazaar-Beta-Linux

Upon confirmation, that package gets moved under the beta assembly. Upon enhancement, one could bring it back again. Here, one knows exactly what is under beta under a platform and the author needs to work on it.

mirek wrote on Thu, 22 December 2016 23:00
It was more C++11 compatibility issue than concept change.
Mirek
In my other thread, I had problems to build theIDE under Centos 7. This I managed to build somehow.

After built, I found that certain dir and files did not get created under the upp dir but got spread out. For e.g. /root/MyApp/, /root/upp.out/, /root/theide, /root/umk got built/copied/created under /root whereas all other dir/files under /root/upp.

Then, I needed to bring them back all under ---> /root/upp. Thereafter, I have changed the path in all *var files from /root/.upp/theide/*.var for e.g.:

OUTPUT = "/root/upp.out";

to
OUTPUT = "/root/upp/upp.out";

Is this a bug somewhere that it creates some files and dirs outside of /upp installation?

The second thing:

In the GCC.bm and CLANG.bm created, the parameter shows the following:

COMMON_CPP_OPTIONS = "-std=c++0x";

Should it not be focusing on c++11?

Subject: Re: U++ 2017 beta
Posted by MrSarup on Sun, 25 Dec 2016 08:23:54 GMT
View Forum Message <> Reply to Message

Hello Mirek,

In my post above, you notice description of errors during compilation. This resulted becaused I made a mistake from the following link here:

http://www.ultimatepp.org/www$uppweb$download$en-us.html

Here, I downloaded 5431 because there is rpm in there! My mistake was not to see the date. Anyway, the correct URL for Stable release should be the following, which the redirection needs to take in there:

https://sourceforge.net/projects/upp/files/upp/2015.2/

As I use Cento 7, I obviously downloaded from the link given there under "Fedora" and "Stable releases"! Consequently, I have installed "successfully" a version build 5431 prepared in middle ages. After starting to work with gcc, I immediately found this out!

Now I downloaded the nightly build 10577M.

Inside the upp-x11-src-10577M.tar.gz, the upp.spec has a wrong rpmbuild command. This should be as follows:

rpmbuild -tb --define 'version 10577M' --define "date $(LC_TIME=En date '+%a %b %d %Y')" upp-x11-src-10577M.tar.gz
...
#define version 10577M

Do correct this. I do find a lot of relative use of path, which is just a headache. The best is to use absolute path that could be substituted. Define once and generate with absolute parameters, instead of relative path.

After this, i.e. executing the above command for rpmbuild in bold, I could not build or compile theIDE with the above command. This is obviously the equivalent of a sequence of entering in the untarred dir and executing make from there.

On Centos 7 64bits, the version 10577M may not get build (or on any linux for that matter) because it is not ready yet. Or there is something that I am still missing or not doing the right thing. Following are the errors:

```
mkdir -p
/root/rpmbuild/BUILD/upp-x11-src-10577M/out/plugin/bmp//home/cxl/Scripts/GCCMK.bm-Gcc-Gui
-Linux-Mt-Posix-Shared/
mkdir -p
/root/rpmbuild/BUILD/upp-x11-src-10577M/out/RichText//home/cxl/Scripts/GCCMK.bm-Gcc-Gui-Li
nux-Mt-Posix-Shared/
mkdir -p
/root/rpmbuild/BUILD/upp-x11-src-10577M/out/plugin/png//home/cxl/Scripts/GCCMK.bm-Gcc-Gui-
Linux-Mt-Posix-Shared/
c++ -c -x c++ -O3 -ffunction-sections -fdata-sections  -std=c++0x -I. -pthread
-I/usr/include/freetype2 -I/usr/include/gtk-2.0 -I/usr/lib64/gtk-2.0/include -I/usr/include/atk-1.0
-I/usr/include/cairo -I/usr/include/gdk-pixbuf-2.0 -I/usr/include/pango-1.0 -I/usr/include/glib-2.0
-I/usr/lib64/glib-2.0/include -I/usr/include/pixman-1 -I/usr/include/libpng15 -I/usr/include/libdrm
-I/usr/include/harfbuzz   -DflagGUI -DflagMT -DflagGCC -DflagSHARED -DflagLINUX
-DflagPOSIX -DflagMAIN  ide/BaseDlg.cpp -o
/root/rpmbuild/BUILD/upp-x11-src-10577M/out/ide//home/cxl/Scripts/GCCMK.bm-Gcc-Gui-Linux-
Main-Mt-Posix-Shared/BaseDlg.o
In file included from ./Core/Core.h:329:0,
                 from ./Esc/Esc.h:4,
                 from ./ide/Core/Core.h:4,
                 from ./ide/Common/Common.h:4,
                 from ide/ide.h:4,
                 from ide/BaseDlg.cpp:1:
./Core/Map.hpp: In member function 'void Upp::Index<T>::Sweep()':
./Core/Map.hpp:268:8: error: call of overloaded 'Sort(Upp::Vector<int>&)' is ambiguous
  Sort(b);
       ^
./Core/Map.hpp:268:8: note: candidates are:
In file included from ./Core/Core.h:269:0,
                 from ./Esc/Esc.h:4,
                 from ./ide/Core/Core.h:4,
                 from ./ide/Common/Common.h:4,
                 from ide/ide.h:4,
                 from ide/BaseDlg.cpp:1:
./Core/Sort.h:119:6: note: void Upp::Sort(Range&) [with Range = Upp::Vector<int>]
 void Sort(Range& c)
      ^
./Core/Sort.h:125:6: note: void Upp::Sort(Range&&) [with Range = Upp::Vector<int>&]
```

```
 void Sort(Range&& c) { Sort(c); }
      ^
In file included from ./Core/Core.h:329:0,
                 from ./Esc/Esc.h:4,
                 from ./ide/Core/Core.h:4,
                 from ./ide/Common/Common.h:4,
                 from ide/ide.h:4,
                 from ide/BaseDlg.cpp:1:
./Core/Map.hpp: In member function 'void Upp::AMap<K, T, V>::Sweep()':
./Core/Map.hpp:590:8: error: call of overloaded 'Sort(Upp::Vector<int>&)' is ambiguous
  Sort(b);
       ^
./Core/Map.hpp:590:8: note: candidates are:
In file included from ./Core/Core.h:269:0,
                 from ./Esc/Esc.h:4,
                 from ./ide/Core/Core.h:4,
                 from ./ide/Common/Common.h:4,
                 from ide/ide.h:4,
                 from ide/BaseDlg.cpp:1:
./Core/Sort.h:119:6: note: void Upp::Sort(Range&) [with Range = Upp::Vector<int>]
 void Sort(Range& c)
      ^
./Core/Sort.h:125:6: note: void Upp::Sort(Range&&) [with Range = Upp::Vector<int>&]
 void Sort(Range&& c) { Sort(c); }
      ^
make: ***
[/root/rpmbuild/BUILD/upp-x11-src-10577M/out/ide//home/cxl/Scripts/GCCMK.bm-Gcc-Gui-Linux-
Main-Mt-Posix-Shared/BaseDlg.o] Error 1
make: Leaving directory `/root/rpmbuild/BUILD/upp-x11-src-10577M/uppsrc'
error: Bad exit status from /root/rpmbuild/tmp/rpm-tmp.Bwwh4d (%build)


RPM build errors:
    Bad exit status from /root/rpmbuild/tmp/rpm-tmp.Bwwh4d (%build)
```

Subject: Re: U++ 2017 beta
Posted by MrSarup on Sun, 25 Dec 2016 08:52:45 GMT
View Forum Message <> Reply to Message

Hello Mirek,

Regarding the above errors and message, I found following post here:

http://www.ultimatepp.org/forums/index.php?t=msg&goto=44 946&#msg_44946

Under Centos 7 64bits, I do not have X11 (except dev package) or KDE installed. It is a simple

CLI Linux server that has some basic LAMP functions.

From the post above, it does not look like that theIDE is ready to be build on a basic Centos server, or?

It would be better to provide installers for Linux versions in shell scripts that does the job. Until this is done, one could get into hundreds of problems. The install shell scripts could be either http://xxx.../linux-upp-stable.sh or http://xxx.../linux-upp-nightly.sh. Then, it will download the right tar ball, check the prequisite dependencies based on the plattform/architecture and install it.

Subject: Re: U++ 2017 beta
Posted by mirek on Sun, 25 Dec 2016 08:52:50 GMT
View Forum Message <> Reply to Message

I guess this might be related to this:

http://www.ultimatepp.org/forums/index.php?t=msg&th=9818 &start=0&

Personally, I would prefer removing .spec file altogether.

.rpm based distros are Spanish village for me. IMO, our responsibility for nightly builds / releases should stop at producing tarball with makefile which builds (perhaps after some package installs) on any most targets. Platform specific packages should be out of ultimatepp.org responsibility... There are too many target platforms to maintain directly...

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Sun, 25 Dec 2016 09:02:22 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Sun, 25 December 2016 09:52Hello Mirek,

Regarding the above errors and message, I found following post here:

http://www.ultimatepp.org/forums/index.php?t=msg&goto=44 946&#msg_44946

Looks like we have posted at the same moment :)

Quote:
From the post above, it does not look like that theIDE is ready to be build on a basic Centos server, or?

Well, theide is GUI app - needs X11/GTK.

Anyway, maybe there is a bit of misunderstanding about what 'building U++' means. It really just compiles theide and umk. U++ does not create any 'permantent' libraries.

Mirek

Subject: Re: U++ 2017 beta
Posted by MrSarup on Sun, 25 Dec 2016 10:24:13 GMT
View Forum Message <> Reply to Message

Hi Mirek,
mirek wrote on Sun, 25 December 2016 09:52
Personally, I would prefer removing .spec file altogether.

.rpm based distros are Spanish village for me. IMO, our responsibility for nightly builds / releases should stop at producing tarball with makefile which builds (perhaps after some package installs) on any most targets. Platform specific packages should be out of ultimatepp.org responsibility... There are too many target platforms to maintain directly...
Mirek

May be you did not see different possibility of using cross-platform FPM packages for distribution of source code.

Thus, I - as I disagree with you - have opened a seperate thread here and created a poll:

http://www.ultimatepp.org/forums/index.php?t=msg&goto=47 174&#msg_47174

I am now curious if you give a vote to my idea!

Subject: Re: U++ 2017 beta
Posted by cbpporter on Sun, 25 Dec 2016 11:18:21 GMT
View Forum Message <> Reply to Message

mirek wrote on Thu, 22 December 2016 10:25It is that time again... Took longer than expected as this ended as the most radical change to U++ in years.

http://www.ultimatepp.org/www$uppweb$Roadmap$en-us.html

Consider the current nightly build a beta. Only serious bugs will be fixed until the release (planned Jan 2).

Mirek
Hi!

And Merry Christmas to you all!

I'm almost back from my hiatus with U++ projects (other urgent tasks came up) and I'm ready for another year of using U++.

But I swear, I won't touch TheIDE ever again if that blasted scroll wheel/loose focus/gain focus bug is not fixed! :lol: :lol: :lol:

Been complaining about if for at least 5 years. I know, I know, shame on me for not fixing it myself.

Other that that, I gave the beta a whirl on a fresh system. First, I tested without MSC to see if the toolchain package is any good. There is still the issue of 5-10 minute wait for TheIDE to finish search for compilers, only to return with nothing but GCC. I don't know why the registry approach was abandoned. I know it can be finicky and it never worked for some users, but 5 minute wait is too much. But I'm having no problems using registry in my own IDE. I even have the problem that it picked up the ancient MSC8 and naturally nothing won't compile on that.

UWord compiled just fine, but GCC debugged crashed as soon as I hit F5. It works well though for command line apps.

Output mode/target file override is still stored locally in the upp folder and applied to new packages. This makes creating a new package a guaranteed error because it will overwrite a binary from another package. Plus it makes it fiddly to move packages between machines because the path must be set up on each one. This option should be stored in the package and kept isolated form other packages.

Other than that, and the fact that at least 6 classes are forked by me, 2 with bugfixes, the entire project suite compiled and works without a hitch.

Our plan is to use 2017.1 and nighlys exclusively for 2017, so no more backwards compatibility and C++1x issues.

Subject: Re: U++ 2017 beta
Posted by Klugier on Sun, 25 Dec 2016 20:29:44 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Sun, 25 December 2016 08:02Hello Mirek,
The second thing:

In the GCC.bm and CLANG.bm created, the parameter shows the following:

COMMON_CPP_OPTIONS = "-std=c++0x";

Should it not be focusing on c++11?

Hello MrSarup,

I pushed patch for this issue - can you check tomorrow with nightly builds (It should be package overnight).

Sincerely and thanks for reporting,
Klugier

Subject: Re: U++ 2017 beta
Posted by omari on Mon, 26 Dec 2016 12:36:21 GMT
View Forum Message <> Reply to Message

Hi,

Quote:
GCC debugged crashed as soon as I hit F5

I think the debugger crash is a gdb bug.
I replaced gdb.exe with version 7.12 downloaded from
http://www.equation.com/servlet/equation.cmd?fa=gdb, and it works without crash.

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 28 Dec 2016 12:01:46 GMT
View Forum Message <> Reply to Message

My review of the beta continues...

So one of the test cases (we use UT a lot) crashes with MINGW. MINGW is more clunky than MSC, but tends to be correct, so I have faith that it is something on our side that can be fixed.

But in the meantime, I installed MSC to see how it fares, since while MINGW compiles well, there is case of GDB crashing and one of a testcase crashing.

I have no idea what MSC you use for the "new" Core, the site is weirdly scarce on info on this (no new comer shall ever find it), but I have used Visual Studio 2015 since I first tried the new C++1x port and it works well. The official build method setup also picks it up (after 5 minutes) and mislabels it MSC15 (it is 14, reported this in the past). Visual Studio 2016/MSC15 has been moved back to 2017 and even if it is released in 2017, with bureaucracy an inertia, you can't expect a lot of people to upgrade to it before 2020. So MSC14 should be properly detected. And it is not, because it misses some include paths.

After I fix it by hand, everything works, including the testcases.

After playing around with it for a day or so, it looks good and stable.

On a sidenote, not related to the beta, Xmlize can be ridiculously slow. Here is are some outputs:

Compilation finished in 0.256 seconds. 0.072 seconds (28.162%) spent on update.
Compilation finished in 0.093 seconds. 0.077 seconds (81.781%) spent on update.

That is 0.160 used on up a single line:
```
for (int i = 0; i < sources.GetCount(); i++) {
  ZPackage& pak = *sources[i].Package;
  StoreAsXMLFile(pak, "cache", pak.OutPath + "\\cache.xml");
 }
```

The StoreAsXMLFile saves a whooping 36.2 KiB of XML on disk in 0.160 seconds. Anything above 0.010 I find unacceptable, so I think it is time to say goodbye to Xmlize.

So In conclusion:
 - The quality of the code in the packages is at its usual high standards and U++ is a great library
 - installing and ease of getting started is at the same all time low it has been for a year now
 - MINGW will never be good until Mirek switches over to it fully for at least 2 months, not touching MSC in this period.

Subject: Re: U++ 2017 beta
Posted by MrSarup on Wed, 28 Dec 2016 12:31:21 GMT
View Forum Message <> Reply to Message

Hello cbpporter,
cbpporter wrote on Wed, 28 December 2016 13:01
My review of the beta continues...
...
So In conclusion:
 - The quality of the code in the packages is at its usual high standards and U++ is a great library
 - installing and ease of getting started is at the same all time low it has been for a year now
 - MINGW will never be good until Mirek switches over to it fully for at least 2 months, not touching MSC in this period.
My review of the community of U++ continues...

I concluded the U++ community is in beta. Thus, I received little support from the community.

Whatever the little support I have received, I would like to draw your attention to my postings here because it related to the compatibility issue you have addressed in your above post:

http://www.ultimatepp.org/forums/index.php?t=msg&th=9820 &start=0&

Of course, it relates to Linux with Centos and Fedora flavor. So, our conclusion - as a new comer -

about the U++ is:

- DO NOT USE U++ UNTIL IT IS FULLY COMPATIBLE WITH GCC ON LINUX!

Hello Mirek,

I have failed to grasp the intention of developers on launching and maintaining this beautiful project while leaving a very fundamental issue of installation and ease of working with it aside.

It appears to me that developers have really not paid much attention on other platform other than windows + android.

While it works like charm on windows, it does not work as easy on other platform. The Era of Bill's Monopoly is gone. We live in a cross-platform world today. My suggestion to developers for further development:

- Maintain the beta status further until the cross-platform compatibility is achieved!

- Freeze everything of what is there TODAY!

- Work on CROSS PLATFORM CONCEPT FIRST. By making compatible on every platform, provide a much better installer for other platforms, line Centos, Fedora and assure that users could very easily install, compile and re-use examples.

- Assure that a user has - from install to compile - least headache, at least for new comers.

- Learn on how to make packages in an efficient matter and provide binary distribution with technologies like FPM.

Have nice holidays ....

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 28 Dec 2016 15:53:20 GMT
View Forum Message <> Reply to Message

Quote:
- Work on CROSS PLATFORM CONCEPT FIRST. By making compatible on every platform, provide a much better installer for other platforms, line Centos, Fedora and assure that users could very easily install, compile and re-use examples.

What do you mean by 'every platform'?

There are thousands of distros, with small differences in packaging everywhere.

I believe that the original status where we provide compilable tarball was OK. It should not be our 'release' responsibility to provide binary packages for 'every platform' for the release.

I guess that if you would have originally downloaded nightly tarball, ignored .spec file, just did make and fixed dependencies (install missing packages), your experience would have been much better.

Mirek

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 28 Dec 2016 18:40:02 GMT
View Forum Message <> Reply to Message

Here are numbers on a very stable release we have with a very stable reference test case:
Compilation finished in 0.034 seconds. 0.029 seconds (85.010%) spent on update.

This is without any serialization.

Compilation finished in 0.043 seconds. 0.030 seconds (66.361%) spent on update.

This is with serialization.

Compilation finished in 0.052 seconds. 0.031 seconds (59.647%) spent on update.
This is with Xmlize.

Even Serialize adds 10ms for writing out 3.7 KiB of data. I need to do a very comprehensive benchmark on all this.

mirek wrote on Wed, 28 December 2016 17:53Quote:
- Work on CROSS PLATFORM CONCEPT FIRST. By making compatible on every platform, provide a much better installer for other platforms, line Centos, Fedora and assure that users could very easily install, compile and re-use examples.

What do you mean by 'every platform'?

There are thousands of distros, with small differences in packaging everywhere.

I believe that the original status where we provide compilable tarball was OK. It should not be our 'release' responsibility to provide binary packages for 'every platform' for the release.

I guess that if you would have originally downloaded nightly tarball, ignored .spec file, just did make and fixed dependencies (install missing packages), your experience would have been much better.

Mirek

I think MrSarup is a bit harsh.

I tried to release software under Linux recently and I think it is near impossible. The Linux world has taken every single sound software distribution principle (except for OSS, which I like and agree with) and has thrown it out of the window. You literally can't develop for Linux. Period. You need to OSS it and pass on the responsibility to platform specific packagers.

And even then the solution is sub par. I have a 64bit Linux. You have a 64bit Linux. I compiled for a 64bit Linux a simple unambitious program 5 years ago. It does not work today. Somebody needs to fix Linux. Part of the far future of our project is fixing Linux, at it involves inventing a new .so format with JIT compilation and version control.

But how about Windows? I compiled 10 years ago some programs. They still run fine.

U++ used to work out of the box.

Now it is very hard to get to work. This needs to be fixed ASAP. You can obviously make it run, so share with us your secret.

MSC14 + some hand picked MINGW should work out of the box.

MSC14 is the only non beta platform that can compile U++. It should be detected instantly and without fault. GDB should not crash on UWord.

There should be clear installation instruction on the site. With a section for common GOTCHAS!

U++ has always had excellent quality in its main components but it was always a hard sell. A bit small. Now widely used. A bit weird. Suffering from a lot of NotInventedHere TM. But the U++ fans like me and all the people on the forum stuck around because of the quality, despite some of the cons. But it was easy to get into. No hassle, no installers. Just drop a ZIP somewhere and it worked. Configurable and hackable. You had the security that you can get it to work.

The new package just doesn't work. When the C++1x rewrite first came, for like 3 months I couldn't use it. I had to use the old versions. Not that I mind, but still. Back then the main site pointed you to a download for a MSC that installed version 6.0 of the libraries. Fully not C++1x compatible of course, since I've been using 8.0 for years now.

Like I detailed in my two posts, all you get is a clunky MINGW version with crashes and a "good luck, but it won't work unless you know exactly which version of MSC installs what where and you create your custom build methods" version.

Nobody will wait around 5 minutes for a setup more than once if the setup quits with no result or it detects something and it doesn't work. Like the beta today. The 5 minute search needs to be optional. Registry instant detection needs to come back. Wizards must be added. "Oh, it looks like you have MSC14 installed. Here are the paths. Do you agree?"

Back in the day, with all of its small problems, you had the security that U++ will be around and will work fine for years to come. Now I have a 2 year plan that can be used to phase out U++ if needed if it will ever feel like it is dyeing. Which it certainly felt like when the C++11x period began.

But I want U++ to live and prosper. I want a bigger user base and for it to receive the acclaim it deserves.

And as a first step, little Timmy who can't code hello world, but has a working MSC, should be able to take a 50 MiB package from u++.org, unpack, double click and have access to the entire package, working perfectly. Like it did form 2007? to 2015.

Subject: Re: U++ 2017 beta
Posted by MrSarup on Wed, 28 Dec 2016 19:53:37 GMT
View Forum Message <> Reply to Message

Hello Mirek and cbpporter,

Both of you have NOT REALLY CAPTURED the crux of the story, although you understand what I have been talking about. I have used both methods, rpm and make. The question of using either of these methods only relates to building theIDE. But if I do not want to build it, I could simply compile a console application.

No one have told me how to compile a simple console application and which commands I need to use it to compile it UNTIL TODAY! In the other thread of FPM, I requested to the poster to let me know what modules i.e. additional cpp files I need to compile with GCC to make a simple console application of SocketServer from reference.

There are no docs. There is no mention anywhere on the website that gives me this information.

What is then this U++ ideal for, to mislead users like me and let them stand alone? I could not work and cannot work until today. So I give up and move forward. This is a classic example of a new comer, who is either not getting help from docs or community or things does not work fundamentally.

I DO NOT WANT TO USE theIDE on Centos 7 but want to code only console applications. Nothing more. So I do not even need to compile with make or rpm, right? Had I known, that this does not work on Centos 7 at all, then I would have saved 20 hours. I needed to read the website, docs, forums, setups, installs, etc.

Where does it say on the website that the U++ is not working with Centos 7 and Fedora?
Was it tested before at all by someone?

I wasted hours and hours to find out this as a new comer trying to compile a little ServerSocket.cpp on Centos! On windows it worked perfect. Lets see what in ServerSocket.cpp console application: Does U++ have many core modules or functions that A MINI CONSOLE APPLICATION FAILS TO COMPILE or run on Centos/Fedora?

Most likely this IS NOT THE CASE! I suspect that there are functions in the core that may not be

needed in there to run a mini console application.

Developers should make some testing on platforms and declare on which platform/version it has been tested. Did the developers know that it is not possible to make a tiny console application on Centos or Fedora?

Currently, I am coding on Laravel PHP Framework. U++ is in principle in the same direction of Laravel Framework on php. They both have similar philosophy. In Laravel, there is composer which works on Mac, windows and Linux. Then, after installing, it does all the work beautifully, like make, make install for php and configure the application or project. It does not have the usual coding techniques and uses its own convention, just like U++ does. This is the reason why Laravel became the best framework. It installs, configures, coding like charm.

However, here I appreciate U++ very much, as I immediately could recognize its quality, and find that U++ will take a long time to increase its user base or grow its community. This will never work, or at least it will take many years of delay.

To solve this, one - as developers - HAS to make sure to release applications that are stable and easy to handle. If this does not work, the sorry is over before it began.

And that's the sad future of U++ I see. I am sorry to take a distance with U++ because the docs are terrible, there are not many examples to handle some complex situations, there is less cross-platform compatibility and the community is tiny.

I remember of making an exe in 2010 on windows. I had intended to make it on Mac too. That did not work. After release, I had heard tons of complains from Mac users.

Cross-platform means that most users should be able to use it on variety of platform. Why would I restrict myself with U++, when I can achieve a users group three times bigger by coding on C++. Here, the story ends if U++ restricts me by imposing Handcuffs of coding techniques.

Then I need to say goodbye to U++. And that's the crux of this story.

So, it is certainly not the question of being harsh. It is the question of usability of a software. If one cannot use it or if the use is difficult, then one cannot use it. As simple.

Apart from that, you will not find many honest ones, who are criticizing so openly like myself, take their time and offer a feedback. I merely take the opportunity to discuss here as I do think that people have not understood certain things and live in their little world surrounded by a tiny community.

I also think that the philosophy of U++ coding is more powerful to have a huge community that what it has achieved until today. The fact that it is extremely powerful and has not progressed so much shows that there needs to be a change somewhere, strategically speaking.

Subject: Re: U++ 2017 beta

Posted by Mindtraveller on Wed, 28 Dec 2016 22:12:44 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Wed, 28 December 2016 22:53The fact that it is extremely powerful and has not progressed so much shows that there needs to be a change somewhere, strategically speaking.
What exactly do you suggest?

Subject: Re: U++ 2017 beta
Posted by Klugier on Wed, 28 Dec 2016 22:15:27 GMT
View Forum Message <> Reply to Message

Hello,

Dear MrSarup, I would like to thank you for your comments. They mean a lot for us.

To build console application outside TheIDE you need to use utilities called UMK (Ultimate make) that builds ultimate package from terminal. More information about that tool can be found on the following documentation http://www.ultimatepp.org/appideumk$en-us.html. The you can code everything from your vi or emacs. And you cannot build upp packages with plain g++ command right now.

Backing to Fedora and Cent OS - we can have there some problems. Personally, I didn't use that kinds of distribution so bugs can be presented there. Probably, we need there good maintainer, but as you said community is tiny. So, thus ensuring support is hard.

Someone says that finding visual studio from registry was good idea. I agree with that and think that this should be recovered fast. I don't know why we drooped it - it works nice on my configuration. I remembered that Mirek tall something that it doesn't work in some cases. But, as cpborder said we can where user accepts the paths found from registry. However this decision belongs to Mirek.

The next problem I found here is that many persons post that U++ has got bugs with MinGW and gdb. I am aware that at the moment we cannot fixed everything. We do not have many resources - the main developer is Mirek. Sometimes I commit new code for TheIDE - but this is rather hobby than my regular work. So, the code is done when i want to. So, if somebody want to help with development of u++ - we can create more svn account to make thinks faster and give more accounts on redmine. In a word, we need more word class developers to make thinks works better.

The last, but not least is of course money. We cannot work full time on that project. We need to find better ways of financing this project. Maybe I don't know 48 hours e-mail support for fee, advertisements in TheIDE, more aggressive donation campaigning etc. I know that some users may don't like it, but money is important for the further development. We can make discussion about this in another topic.

I relay believe ++ has strong potential - that we should better use. In my opinion the two most

important factor is lack of new active developers and money to make current developers more productive. However, even with that problems it is fantastic framework - that is used by many happy users. Mirek dose fantastic job here - at some point of my life he was a relay good mentor for me. At the end I would like to wish you dear U++ all the best in 2017.

Sincerely,
Klugier

Subject: Re: U++ 2017 beta
Posted by amrein on Thu, 29 Dec 2016 07:44:28 GMT
View Forum Message <> Reply to Message

Hi MrSarup

We are trying to help you as you can see. Please keep posting into your first thread only. It becomes difficult to follow your issue if you keep posting in every thread.

http:// www.ultimatepp.org/forums/index.php?t=msg&goto=47167&amp ;#msg_47167

As I said elsewhere, you can compile stable upp source on Centos 7 without error. It's only the last snapshots that won't compile.

Subject: Re: U++ 2017 beta
Posted by mirek on Sun, 01 Jan 2017 20:51:00 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Sun, 25 December 2016 12:18mirek wrote on Thu, 22 December 2016 10:25It is that time again... Took longer than expected as this ended as the most radical change to U++ in years.

http://www.ultimatepp.org/www$uppweb$Roadmap$en-us.html

Consider the current nightly build a beta. Only serious bugs will be fixed until the release (planned Jan 2).

Mirek
Hi!

And Merry Christmas to you all!

I'm almost back from my hiatus with U++ projects (other urgent tasks came up) and I'm ready for another year of using U++.

But I swear, I won't touch TheIDE ever again if that blasted scroll wheel/loose focus/gain focus

bug is not fixed! :lol: :lol: :lol:

Please, file here or into RM detailed description of the problem (esp. how to reproduce it).

Subject: Re: U++ 2017 beta
Posted by mirek on Sun, 01 Jan 2017 20:54:47 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 28 December 2016 13:012017, with bureaucracy an inertia, you can't expect a lot of people to upgrade to it before 2020. So MSC14 should be properly detected. And it is not, because it misses some include paths.

Please, can you list them so that I can fix it? (It did not miss these last time I have checked).

Quote:
On a sidenote, not related to the beta, Xmlize can be ridiculously slow. Here is are some outputs:

Compilation finished in 0.256 seconds. 0.072 seconds (28.162%) spent on update.
Compilation finished in 0.093 seconds. 0.077 seconds (81.781%) spent on update.

Will gladly look into this after current release... RM issue with perhaps some testing code would help.

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Sun, 01 Jan 2017 21:03:05 GMT
View Forum Message <> Reply to Message

OK, so here is the plan for 2017:

Windows:

- rename MSC15 to MSC14

- return to registry check of MSC. If not found, suggest full directory scan (with prompt first, to avoid long scans).

- update MINGW to the point that debugger works.

- fix include paths issue (if cbpporter will be kind enough to provide info)

Tarballs:

- provide separate 'umk only' make option

- info on using umk to compile

- ? remove .spec ?

- if possible, option to use clang instead of gcc (for distros using buggy 4.8.* GCC)

Subject: Re: U++ 2017 beta
Posted by MrSarup on Mon, 02 Jan 2017 05:09:20 GMT
View Forum Message <> Reply to Message

Hello Mirek,

HAPPY NEW YEAR TO MIREK AND THE COMMUNITY OF U++.

mirek wrote on Sun, 01 January 2017 22:03OK, so here is the plan for 2017:
Windows:

Here is my wishlist for 2017:

1) INSTALL AND COMPILE SCRIPTS ON CONSOLE

From Linux, we know that there are shell script-installers available. It guides the whole process by choosing numbers. It also detects if certain dependencies are installed and, if not, does the needful. Release an installer that does all the work on console with bash scripts stable.

For e.g. ./umk.sh could offer a menu to choose, if one wants to install or compile. If one chooses compile, it could offer assemblies as choices after reading the local directories. Choosing one of them, it could offer packages to be compiled.

2) DEVELOPING U++ FOR OTHER PLATFORMS

Change the status of U++ to beta UNTIL IT WORKS ON ALL PLATFORM (and do not laugh on this...). Make distribution available that works on all platforms!

3) MAKE COMPLEX EXAMPLES AVAILABLE

It would be helpful to provide examples related to workflows and multiple windows.

In examples, I have not seen (on windows GUI) that multiple windows could be easily created and

attached to workflows. However, I could be - as a new comer - wrong.

After server, cloud hosting, virtual machines, etc. technology became popular and stable, so many people use c++ for their applications. U++ can provide an alternative in this area too.

Here, more examples, complex ones, are needed to show this and help an user to begin with. This could include examples of console applications on client and servers, how they could interact with each other with different methods, like SSH, Web sockets, Proxy services, etc.

4) DISTRIBUTION PACKAGES

Amrein-Marie has compiled one rpm and tested on Fedora and Centos. In the spec file, I found that he did several years ago, and thereafter not. Well, now its there, it could be made available on Sourceforge.

Cross-platform packages needs to be used, like FPM or any other such technologies that detects dependencies and does the needful.

I do not agree with Mirek on his idea that the responsibility of a developer ends by providing a tarball. For an overloaded single developer, he is - in fact - right to say this. But no developer group would say this, if they are working as a developers community.

5) EXTENSIVE DOCUMENTATION FOR CROSS-PLATFORM

From the other thread, you could see that there was no documentation on building umk on console. Everything is based on building theIDE on GUI. Thus umk remained under-developed.

Hello Mindtraveller ,

Mindtraveller wrote on Wed, 28 December 2016 23:12MrSarup wrote on Wed, 28 December 2016 22:53The fact that it is extremely powerful and has not progressed so much shows that there needs to be a change somewhere, strategically speaking.
What exactly do you suggest?
BAN MIREK FROM PARTICIPATING IN THIS FORM! In particular Mirek should not participate in normal postings and spend his time here at the cost of development of U++. Other community members should take this responsibility instead .

Beyond that, my suggestions are above.

One should create an ACTION GROUP to achieve cross-platform quality, where everyone contributes a little so that the sole developer Mirek is not left alone.

Without this, the Usability of U++, on the international platform, is limited. With this understanding of how incompatible U++ on cross-platform is, I question if the time spent on further development is in proportion to its usability.

Subject: Re: U++ 2017 beta
Posted by mirek on Mon, 02 Jan 2017 20:42:33 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 01 January 2017 22:03
- rename MSC15 to MSC14

- return to registry check of MSC. If not found, suggest full directory scan (with prompt first, to avoid long scans).

Done in 10625 revision (except for now, it is only registry check).

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Mon, 02 Jan 2017 21:31:43 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Mon, 02 January 2017 06:09
Make distribution available that works on all platforms!

OK, going to garage to fetch my old CP/M machine.

I hope I will be able to make C++11 work on it somehow, maybe if I hack those old tape drives and punching cards to it?

Subject: Re: U++ 2017 beta
Posted by cbpporter on Tue, 03 Jan 2017 10:21:46 GMT
View Forum Message <> Reply to Message

mirek wrote on Sun, 01 January 2017 23:03
- fix include paths issue (if cbpporter will be kind enough to provide info)

I would gladly help, but now, on the first official work day on my main dev machine with 10625 the paths are detected and the build method labeled as MSC14. But this machine worked before, so it should work now.

And if memory serves me correctly, the same paths are detected. I'm starting to suspect that it depends from machine to machine and/or number of MSC versions you have installed and Windows versions. It looks like U++ detected the correct include paths, but the MSC installer did not add all the ucrt or um files there, only some.

Anyway, I'll check the computer I used for the beta review to see the paths.

As for 10625, MSC14 is detected and labelled and works. A fresh install will wait a bit after you click "Accept" until theIDE pops up, just enough to think that it crashed. A simple popup with an oscillating progress bar and "preparing for first launch" would alleviate these problems, but it is not necessary.

Quote:OK, so here is the plan for 2017:

Windows:

- rename MSC15 to MSC14

- return to registry check of MSC. If not found, suggest full directory scan (with prompt first, to avoid long scans).

- update MINGW to the point that debugger works.

- fix include paths issue (if cbpporter will be kind enough to provide info)

Tarballs:

- provide separate 'umk only' make option

- info on using umk to compile

- ? remove .spec ?

- if possible, option to use clang instead of gcc (for distros using buggy 4.8.* GCC)

Looks like a solid plan! U++ code is great right now and except for new features I doubt it has much room to improve (maybe even more move inside the lib?), but do feel free to prove me wrong! :)

So the focus should be on usability. Nothing too fancy, but the package should work out of the box in 99% of the cases on Windows. It is a bit of a self defeating process to have a bit site (this is a pretty big site for a project "nobody" uses) that claims up and down the merits and productivity gains with U++ and then you spend 30 minutes getting it to work. Or 1 minute in my case since I know what is wrong since I had to fix it so many times before. I believe you on the site claims, but newcomers will not.

Anyway, as I said before, part of this is my fault. I have little free time and when I have some, once I track down a bug and report it once, the time is all gone and don't have time to keep checking back, making sure it is fixed.

I'll try my best to improve upon this in 2017. I have at least 6 bugs, 2 minor enhancements and one major one that I have to port into each new U++ version, so I hate updating to a new nightly.

Must use a lot of Redmine...

Subject: Re: U++ 2017 beta
Posted by amrein on Tue, 03 Jan 2017 13:22:57 GMT
View Forum Message <> Reply to Message

Hi mirek

Here is how you could use gcc version in a script:

```
gcc_version=$(gcc -dumpversion | sed -e 's/\.\([0-9][0-9]\)/\1/g' -e 's/\.\([0-9]\)/0\1/g' -e 's/^[0-9]\{3,4\}$/&00/')

if [ $gcc_version -gt 40900 ]
then
  echo "gcc version $(gcc -dumpversion) is greater than 4.9.0"
else
  echo "gcc version $(gcc -dumpversion) is lower than 4.9.0"
fi
```

Why should you still dump upp-devel.spec? At present it works out of the box in last snapshots.

Subject: Re: U++ 2017 beta
Posted by mirek on Tue, 03 Jan 2017 13:59:31 GMT
View Forum Message <> Reply to Message

amrein wrote on Tue, 03 January 2017 14:22
Why should you still dump upp-devel.spec? At present it works out of the box in last snapshots.

That is the catch. It works now, but I am unable/unwilling to maintain it. So once it gets broken (and it will), it will be unnoticed.

I really think that tarball is one thing (one that I am able to maintain) and packages another step...

Mirek

Subject: Re: U++ 2017 beta

Posted by amrein on Wed, 04 Jan 2017 00:04:23 GMT
View Forum Message <> Reply to Message

The previous version has never broke in years! When a snapshot didn't compile, is was because of unmatched gcc or library version or because of source code errors.
To be clear, running 'make' gave the same error.

The difficult part for most people was to run the long rpmbuild command with its parameters. Now it's really easy to use: on all rpm based linux distributions yuu just run 'rpmbuild -ta snapshot.tar.gz'.

On Fedora rpmbuild will automatically use g++ and on other linux distributions it will use clang++ (to prevent the use of gcc < 4.9.0).

I modified the POSIX/X11 installation guide with topic++ so now most people will have their answer without asking in the forum.
In the future, I guess that I will have to revert back to gcc according to new linux distribution versions.

If you still want to remove upp-devel.spec than, well, does it really matter? If it's broken, at least rpm packagers will have something to start from. If you remove it, most people will use the standard 'make' procedure.

Is there a team responsible for U++ packaging? I would like to discuss automatic building on debian and other linux/bsd based distributions.
For example, the 'debian' file in root need to be renamed. Something like 'buildrequires.debian'. This 'debian' file prevents me from building a standard debian package because those packages need to have a debian directory with several files within...

Subject: Re: U++ 2017 beta
Posted by MrSarup on Wed, 04 Jan 2017 06:28:39 GMT
View Forum Message <> Reply to Message

Hello Amrein -Marie,
amrein wrote on Wed, 04 January 2017 01:04
If you still want to remove upp-devel.spec than, well, does it really matter? If it's broken, at least rpm packagers will have something to start from. If you remove it, most people will use the standard 'make' procedure.
I must add and support you on your objection to the idea of removing the spec file. Even if Mirek populated the idea of removing the spec file, I dare say so blatantly that this idea is a garbage idea. Perhaps Mirek has not done a lot of activity on different platforms or is less informed about trouble-shooting on other platforms other than the damn Bill's platforms.

For this, I would like to share my experience during build. At the time I started to work on my Centos Server with U++, i.e. to compile it, I could not easily with make.
The dependencies you have listed in the other thread on discussion on my problem, were not sufficient. There was a binary still missing on my server. It did not get identified from the list you

mentioned.
Thereafter, I used the spec file. Only then yum package manager could identify all dependencies and compiled it.
Later, you took the daunting challenge to make rpm and provide both, *src.rpm and *rpm. With either of these, I could work with umk.

Without your help, i.e. in the absence of rpms, I had to invest a lot of time and energy. This could be saved in case of all new comers.

The Linux world stopped working with a "tar ball only" approach several decades ago, when the idea of package managers got popular. It appears that this is not understood here. Let's say, will the Linux community announce the following for Centos:

yum -y install upp

No! If not, that idea is garbage!!! As simple. Period.

This is a classic example of the restrictive development process of U++ that centers around Mirek's taste, time, approaches, etc.
The reason is that he is alone, or predominantly working on this project and there are not many active developers available.

Thus, as I suggested above, an action group is necessary to share this work, instead of leaving to his shoulders.

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 04 Jan 2017 06:59:58 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Wed, 04 January 2017 07:28
The Linux world stopped working with a "tar ball only" approach several decades ago, when the idea of package managers got popular. It appears that this is not understood here. Let's say, will the Linux community announce the following for Centos:

I believe that you are misguided about this. E.g. by your theory, you should see .rpm packages here:

ftp://ftp.fu-berlin.de/unix/languages/gcc/releases/gcc-6.3.0 /

Where are they?

Mirek

Subject: Re: U++ 2017 beta

Posted by mirek on Wed, 04 Jan 2017 07:03:54 GMT
View Forum Message <> Reply to Message

Mingw64 in Win32 installation updated to the latest version (6.1.0), seems to have fixed gdb issue. Please test!

Mirek

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 04 Jan 2017 07:08:09 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Wed, 04 January 2017 08:28
The Linux world stopped working with a "tar ball only" approach several decades ago, when the idea of package managers got popular. It appears that this is not understood here. Let's say, will the Linux community announce the following for Centos:

yum -y install upp

No! If not, that idea is garbage!!! As simple. Period.

This is a classic example of the restrictive development process of U++ that centers around Mirek's taste, time, approaches, etc.
The reason is that he is alone, or predominantly working on this project and there are not many active developers available.

Sorry there, but the must be the most wrong thing I heard all year.

Linux has moved on decades ago but hasn't really made any progress.

Software distribution on Linux is the worst out of all OSes and has always been so. It is baby level super primitive stuff. It is so bad that you CAN NOT do software distribution for Linux as an entire platform. You can't. Nobody can. Not a single developer. It is just that hard.

So people have adapted (instead of fixing Linux) and moved part of responsibility downstream. The developer provides buildable sources for one environment, often using automake and friends to maximize compatibility and completely different people not affiliated with the developer will choose when/if/how to package for their distribution. And if the developer updates, he can't push an update. The external packager chooses to update if he wishes.

Sure, there are exceptions, but in general this is the work-flow. Otherwise, even a hello world application would take you literally months to package. There are over a dozen very popular Linux distros. And over another 50 in somewhat wide use. And hundreds more that are stable. And hundreds more. There are many package formats. For each version even the most common .so can change, so everything needs to built. And even if you packaged for all, some people still expect a tarball or they will call you out for not being OSS enough.

Even if you manage to create a package that works for multiple Linux versions, half of them won't accept it into their repo. You need to set up your custom third party repo or go back to tarballs. So you have to set up an maintain a dozen or more servers. Users need to add the custom repo.

So yeah, software distribution on Linux in near impossible as a developer and this stupid situation happens only on Linux.

The only mistake that U++ does in its distribution of tarballs is not using standard tools and automake. The auto* tools work, but ideologically, they are the worst thing ever. They rely on decades worths of tips and tricks, clever ways to detect Linux differences and do a million things for even the simplest ./configure. And this is bad because you shouldn't have to have such a mammoth of an architecture just to build. Something that needs such effort to build is bad.

Period.

The build system I'm working on right now knows this. It is designed to build arbitrarily complex projects with a short list of folders and a couple compilation options. The entirety of autoconf and friends' wisdom is discarded, not because it is bad, but because it shouldn't be needed. There is configure, make and make install. There is just on command and it works.

I think that maybe umk tried to do something similar.

Oh, and if you think I'm wrong, I dare you to prove it. Spend month trying to find a good method for Linux for my own needs and there is none.

But if you think there is, please show me the link to the standardized no-hassle tool that allows you to package a piece of software once and it will produce one or multiple binary packages capable of working out of the box (and install dependencies) on all of Linux.

Subject: Re: U++ 2017 beta
Posted by MrSarup on Wed, 04 Jan 2017 07:47:51 GMT
View Forum Message <> Reply to Message

Hello cbpporter,
cbpporter wrote on Wed, 04 January 2017 08:08
Oh, and if you think I'm wrong, I dare you to prove it. Spend month trying to find a good method for Linux for my own needs and there is none.

But if you think there is, please show me the link to the standardized no-hassle tool that allows you to package a piece of software once and it will produce one or multiple binary packages capable of working out of the box (and install dependencies) on all of Linux.

Sure, here is the link:
http://http://www.wxwidgets.org/

On the main page it declares the following:

"wxWidgets is a C++ library that lets developers create applications for Windows, Mac OS X, Linux and other platforms with a single code base."

The end product counts! The end result counts!!!

Using wxWidgets with CodeBlocks, I have an instant integration of all c++ features.

I see no reason why I should work on research of what and how U++ distribution needs to be done. There are reasons why U++ did not become popular. Difficulties to install and compile, apart of other things, is one of the many.

As a new comer, I see no reason why I should convince you, or any one else in this community, of what is good and what is bad. I came here to use this excellent source code.

The fact remains unchanged: I could not easily work with it. I am more than sure that I will have hundreds of difficulties later on.

Working now with wxWidgets, I have a support of a normal c++ community everywhere.

Consequently, the problem starts with U++ on every other level. This is a bottleneck, which is embedded within the system of U++, where many things are required by its special quality that needs to be made easy.

And that's not the case. But that's my problem. I need to choose the right open source coding that provides me a good start. After spending many hours, the only thing I could do is compile the theIDE!

What could be more ridiculous that this? My choice was wrong...

I strongly think that we all are wasting our time in making such discussions. If you think everything is right here, then please remain happy and work with it. I think this did not apply to me. The associated factors are just so gigantic that it becomes not worth working further. These factors comes along with, along with the U++ source code, that are not separable.

I should go, even though I loved the approach of U++, a very intelligent one, and find something else to work with.

Subject: Re: U++ 2017 beta
Posted by MrSarup on Wed, 04 Jan 2017 08:00:39 GMT
View Forum Message <> Reply to Message

Hello Everyone,

I have unsubscribed from all the threads, including this one. So, you need not continue posting addressing to me, even if you have a reason to do so.

I do not plan to visit this forum anymore or use U++.

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 04 Jan 2017 09:06:05 GMT
View Forum Message <> Reply to Message

MrSarup wrote on Wed, 04 January 2017 09:47
Sure, here is the link:
http://http://www.wxwidgets.org/

What I meant is a tool that takes out the hassle out of building packages for each platform, not an individual package that may or may not have managed to achieve high Linux distribution penetration.

Quote:
On the main page it declares the following:

"wxWidgets is a C++ library that lets developers create applications for Windows, Mac OS X, Linux and other platforms with a single code base."

This is 100% the same for U++. One code base that works out of the box on some platforms (some temporary problems not withstanding), works with minor hassle on a lot more and can be made to work on anything with a C++11x (or normal C++ if you revert to old versions) with a couple of hours of tweaking. At the end of the day, all you need is a C++ compiler and at msot a dozen of dependencies, most of these related to X.

That's all you need.

And yes, this should be improved. U++ is very well supported for *.deb based distros, but even that fails often. Then, you need to take the tarball and run make. And completely necessary and off-putting fact, but after you do this, you have a fully working U++.

Quote:
I see no reason why I should work on research of what and how U++ distribution needs to be done.

I see where you are coming from, but I'm afraid you can't avoid all the hassle since CentOS is not listed on the download page as a supported platform. I'm sure I could get it to work anywhere from 10 minutes to at most 1h if I ever install CentOS, but end users should have to do this.

Quote:
As a new comer, I see no reason why I should convince you, or any one else in this community, of what is good and what is bad. I came here to use this excellent source code.

The fact remains unchanged: I could not easily work with it. I am more than sure that I will have hundreds of difficulties later on.

As said, there are problems. But they are mostly related to the install, building and library location process. After this is done, you are very unlikely to encounter difficulties later, except for:
- GDB integration is not that great, so debugging works but is not the best you have ever seen
- some more rarely used methods from rarely used classes might have minor bugs. These usually get fixed if reported here, or even better on Redmine.
- nightly downloads are very stable, but still not as stable as some releases

Quote:
Consequently, the problem starts with U++ on every other level. This is a bottleneck, which is embedded within the system of U++, where many things are required by its special quality that needs to be made easy.

The only real problem I see is TheIDE. With U++ it is TheIDE or the highway. Or umk. I never wanted or liked using TheIDE, but you kind of have to and I have gotten used to it. Trying to not use it is a major hassle that eventually leads everybody to give up on this endeavor. I'm sure that a lot of the user base, if U++ worked out of the box with Visual Studio, would never boot up TheIDE again.

But at the end of the day, it works. Been using it for who knows how many years now (7+) and it works. I view it as a tax: you want Core and the widgets? Well TheIDE is your tax!

I remember in my old job, we got some senior on one of the U++ projects. Very talented guy and great coder. Didn't want to use use TheIDE, but said he'll use it for a day or two to see what's the deal with the packages and learn the code and the move over to Visual Studio. Soon, he became very angry, cursing TheIDE. I asked what's wrong. He way very explicitly cursing it, saying what kind of an IDE does not have an "Open file" option. He just wanted to open a file to edit it and couldn't do it. I showed him that the option was "Edit file", not "Open file". He never touched TheIDE again, nor U++ or the project.

Quote:
And that's not the case. But that's my problem. I need to choose the right open source coding that provides me a good start. After spending many hours, the only thing I could do is compile the theIDE!

I'm afraid that's on you. Getting TheIDE to compile and work, is the one major obstacle and blocking point. After you can compile that and are willing to use TheIDE, compiling anything becomes trivial. All you need to do is provide it with some include paths and lib references and 99% of the code out there should work.

Subject: Re: U++ 2017 beta

Posted by mirek on Wed, 04 Jan 2017 09:28:28 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 04 January 2017 10:06
I'm sure that a lot of the user base, if U++ worked out of the box with Visual Studio, would never boot up TheIDE again.

Perhaps it is time to do that... I mean, with .icpp problem gone, somebody could try again... Regroup sources to classic .libs should be easy now.

(The only trouble is layout/icon designers, but you can use theide for it, it works with commandline parameters - perhaps even rename the executable to 'upp_designer' or something).

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 04 Jan 2017 10:19:02 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 04 January 2017 11:28cbpporter wrote on Wed, 04 January 2017 10:06
I'm sure that a lot of the user base, if U++ worked out of the box with Visual Studio, would never boot up TheIDE again.

Perhaps it is time to do that... I mean, with .icpp problem gone, somebody could try again... Regroup sources to classic .libs should be easy now.

(The only trouble is layout/icon designers, but you can use theide for it, it works with commandline parameters - perhaps even rename the executable to 'upp_designer' or something).

Maybe.

A bit too late for that.

I think at this stage it is better to improve the debugger. I'm very strapped on time, but I did boot up some 2009 debugger code and it still compiles in TheIDE, so I will do some tests with improving String debugging.

Currently, 75% of my debugging time is spent with looking at strings. And in 90% of those times, I need to DUMP that string because the debugger shows me a really long line of useless information, yet it cuts off often the actual string part of the String.

The tooltip clips of screen, it has no multiline support and is generally not good.

But I'll focus on strings first. Won't have probably time for anything more.

Here is a mockup of what I'm thinking:

The top one is the current solution: ugly, bad, information overload. I have full trust in U++. I don't need to debug your string implementation. I need to debug my strings. Their content! The bottom one is my mockup. first line, full string, with far greater clipping size. Followed by length. Second line is the rest of the bullshit I don't care about.

When debugging bigger classes with multiple fields, the second line should be either put at the end of the first or dropped all together. Oh, and in the case of these classes, if the have few fields, like let's say less then 10, each field should be on its own line.

This is just a simple mock-up. I bet if I boot up an old C# project in Visual studio or an old Java project in Eclipse and copy random ideas from those debuggers, the mock up can be improved exponentially!!!

PS: since I'm here, how do you change the filename of the default application log?

File Attachments
1) mockup.png, downloaded 616 times

Subject: Re: U++ 2017 beta
Posted by Tom1 on Wed, 04 Jan 2017 11:32:17 GMT
View Forum Message <> Reply to Message

Hi,

Not sure how recent behavior this is, but for some reason:
Format("%.2f",1.23);
results in 1,23 (with comma) instead of expected 1.23 (with decimal point) when using "GUI MT" in Linux. If I select "GUI MT X11", the result is correctly 1.23 (with decimal point).

Can this be fixed before release? If the separators (or date/time formats for that matter) automatically change depending on regional settings used on each computer, reading and writing of various ASCII file formats gets completely out of control.

Best regards,

Tom

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 04 Jan 2017 11:38:25 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 04 January 2017 12:32Hi,

Not sure how recent behavior this is, but for some reason:
Format("%.2f",1.23);
results in 1,23 (with comma) instead of expected 1.23 (with decimal point) when using "GUI MT" in Linux. If I select "GUI MT X11", the result is correctly 1.23 (with decimal point).

Can this be fixed before release? If the separators (or date/time formats for that matter) automatically change depending on regional settings used on each computer, reading and writing of various ASCII file formats gets completely out of control.

Best regards,

Tom

".2f" is actually using C lib (see FloatFormatter). Can you check the same with 'printf' or 'sprintf' ? (That said, I can certainly recplace ',' with '.' in FloatFormatter, just to be sure...).

Mirek

Subject: Re: U++ 2017 beta
Posted by Tom1 on Wed, 04 Jan 2017 12:20:58 GMT
View Forum Message <> Reply to Message

Hi,

Both Format("%.2f",1.23); and sprintf(text,"%.2f",1.23); use comma as decimal separator when under "GUI MT" in Linux. With "GUI MT X11" both are clean decimal points.

In Windows both used decimal points.

I never imagined GTK could affect text formatting!

Thanks and best regards,

Tom

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 04 Jan 2017 13:02:51 GMT
View Forum Message <> Reply to Message

Then GTK must be changing your locale with (non-)portable means of setting this in the std c lib.

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 04 Jan 2017 13:39:35 GMT

View Forum Message <> Reply to Message

Tom1 wrote on Wed, 04 January 2017 13:20Hi,

Both Format("%.2f",1.23); and sprintf(text,"%.2f",1.23); use comma as decimal separator when under "GUI MT" in Linux. With "GUI MT X11" both are clean decimal points.

In Windows both used decimal points.

I never imagined GTK could affect text formatting!

Thanks and best regards,

Tom

OK, so what is the correct resolution here?

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 04 Jan 2017 13:49:03 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 04 January 2017 11:19
But I'll focus on strings first. Won't have probably time for anything more.

Here is a mockup of what I'm thinking:

Trouble with that is that it requires "special treatment" for String - debugger needs to be aware that you are at String.

Frankly, the ideal solution would be calling 'AsString' for values somehow, but that aint so easy unfortunately...

Subject: Re: U++ 2017 beta
Posted by Tom1 on Wed, 04 Jan 2017 14:29:57 GMT
View Forum Message <> Reply to Message

Hi,

@cbpporter: My locale on both Windows and Linux (Mint 18.1) computers is "Language: English, Region: Finland". Numeric formats are under Region, which is Finnish, and surely use comma as

decimal separator, but that has never affected my programs on WIN or X11. I do not know how GTK changes the sprintf and Format behavior, but that certainly is annoying and can have severe consequences since the calls to these functions are scattered all around my code in 'million' places.

@Mirek: I bet this will divide opinions: My opinion is that I should be able to trust on Format to use decimal point regardless the region. I.e. the way it has always been with WIN and X11. If region compatible Formatting output is desired, that should be addressed through e.g. some "FormatRegional()" API or some optional formatter flags which are disabled by default.

For some users it might be nice to have numeric values in dialogs shown with regional separators and display formats, but for writing e.g. CSV files (comma separated values), the regional separators will just introduce a catastrophy. (1,23,2,34,3,45 ...) Even, if list separators were different, the international file transfer would be a disastrous mess with tens of file layouts.

--

Is it possible to make U++ on GTK to behave consistently like WIN and X11 do?

Best regards,

Tom

Subject: Re: U++ 2017 beta
Posted by cbpporter on Wed, 04 Jan 2017 14:38:40 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 04 January 2017 15:49cbpporter wrote on Wed, 04 January 2017 11:19
But I'll focus on strings first. Won't have probably time for anything more.

Here is a mockup of what I'm thinking:

Trouble with that is that it requires "special treatment" for String - debugger needs to be aware that you are at String.

Frankly, the ideal solution would be calling 'AsString' for values somehow, but that aint so easy unfortunately...

Yeah, that's exactly what I'm doing. Added special logic based on type name. Small price to pay I think for the added benefits. And not just String. A few Core classes should be handled like this too, like Vector, which has abysmal debugging. This isn't 1995 anymore, we can't have it like it is. Plus, the debugger should be aware of a few more other things, like Moveable. Remove all

unnecessary information.

Think of the most fancy and elegant scripting language possible. Something people love to use and once they start to use it, become extremely loyal to it, like Python or Ruby. Now imagine how streamlined and useful debugging can look in those (there is support for full time run-time reflection and eval). That's what I want in U++. A String shouldn't have len, ptr, s, wptr, w, q and what not. What are even those? I know what they are, but I don't want to. Nobody does! The debug output should be readable for a String by a baby.

We can't do stuff like in the scripting languages, but we can hard code the debugger to pretty print a couple of hand picked types if in debug mode and if they have the proper layout.

I tested a very early and hack version of this for U++ in 2009, but there was no interest for it. Now I must do it because I can't stand to debug another string where 1/3 of it is cut off so I can have a clear look at the blasted wptr value. Or to debug a vector where I can only look at the first element. Things like this worked in Delphi like 10 years ago. They work perfectly in C#.

I don't have C# installed right now on this computer, but here is a screenshot of a decent, not great debugger:

The length of those fields in the window are enough to show most strings and there is no extra unneeded info.

File Attachments

1) image001.gif, downloaded 497 times

Subject: Re: U++ 2017 beta
Posted by mirek on Wed, 04 Jan 2017 14:47:47 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 04 January 2017 15:38mirek wrote on Wed, 04 January 2017 15:49cbpporter wrote on Wed, 04 January 2017 11:19
But I'll focus on strings first. Won't have probably time for anything more.

Here is a mockup of what I'm thinking:

Trouble with that is that it requires "special treatment" for String - debugger needs to be aware that you are at String.

Frankly, the ideal solution would be calling 'AsString' for values somehow, but that aint so easy unfortunately...

Yeah, that's exactly what I'm doing. Added special logic based on type name. Small price to pay I think for the added benefits. And not just String. A few Core classes should be handled like this too, like Vector, which has abysmal debugging. This isn't 1995 anymore, we can't have it like it is. Plus, the debugger should be aware of a few more other things, like Moveable. Remove all unnecessary information.

Think of the most fancy and elegant scripting language possible. Something people love to use and once they start to use it, become extremely loyal to it, like Python or Ruby. Now imagine how streamlined and useful debugging can look in those (there is support for full time run-time reflection and eval). That's what I want in U++. A String shouldn't have len, ptr, s, wptr, w, q and what not. What are even those? I know what they are, but I don't want to. Nobody does! The debug output should be readable for a String by a baby.

We can't do stuff like in the scripting languages, but we can hard code the debugger to pretty print a couple of hand picked types if in debug mode and if they have the proper layout.

Been there - that is what micio with MI2 debugger was trying. For me, it was never really usable. But maybe you will have a better luck :)

Subject: Re: U++ 2017 beta
Posted by Klugier on Wed, 04 Jan 2017 21:32:37 GMT
View Forum Message <> Reply to Message

cbpporter wrote on Wed, 04 January 2017 10:06
I remember in my old job, we got some senior on one of the U++ projects. Very talented guy and great coder. Didn't want to use use TheIDE, but said he'll use it for a day or two to see what's the deal with the packages and learn the code and the move over to Visual Studio. Soon, he became very angry, cursing TheIDE. I asked what's wrong. He way very explicitly cursing it, saying what kind of an IDE does not have an "Open file" option. He just wanted to open a file to edit it and couldn't do it. I showed him that the option was "Edit file", not "Open file". He never touched TheIDE again, nor U++ or the project.

Hello cbpporter,

Good catch :lol: - "Edit file" is now called "Open file".

Sincerely,
Klugier

Subject: Re: U++ 2017 beta

Posted by mr_ped on Thu, 05 Jan 2017 04:01:37 GMT
View Forum Message <> Reply to Message

About decimal comma.

Well, in Java world you should always provide also the Locale to be used by the formatter. So then you do thing like str = Format(Locale.US, "%.2f", value) when you are exporting data for machine, or you call with other (none=default) to get formatting for human output, which can include also things like thousands separator/etc (and is not suitable for machine export/import).

Tom1:

BTW, if your OS (GTK) is set to ",", then it sort of does what it should, and I would rather like to see introduction of this Java-like "provide target Locale" logic, than some hardcoded hacks to get "." every time. As when you want to format numbers for your human users, you should follow their OS settings.

I do believe that you can set locale to the "C" even in your source even now (maybe at the start of application, if you don't want to format outputs with default OS settings), to get the printf behaviour with dot. Too lazy to google for some example, sorry. :)

Subject: Re: U++ 2017 beta
Posted by mirek on Thu, 05 Jan 2017 07:20:28 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Thu, 05 January 2017 05:01About decimal comma.

Well, in Java world you should always provide also the Locale to be used by the formatter. So then you do thing like str = Format(Locale.US, "%.2f", value) when you are exporting data for machine, or you call with other (none=default) to get formatting for human output, which can include also things like thousands separator/etc (and is not suitable for machine export/import).

Actually, that is in U++ for ages:

String Format(int language, const char *fmt, __List##I(E__NFValue));

Only you have to use U++ formatters instead of C ones...

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Thu, 05 Jan 2017 07:23:36 GMT
View Forum Message <> Reply to Message

Klugier wrote on Wed, 04 January 2017 22:32cbpporter wrote on Wed, 04 January 2017 10:06
I remember in my old job, we got some senior on one of the U++ projects. Very talented guy and great coder. Didn't want to use use TheIDE, but said he'll use it for a day or two to see what's the deal with the packages and learn the code and the move over to Visual Studio. Soon, he became very angry, cursing TheIDE. I asked what's wrong. He way very explicitly cursing it, saying what kind of an IDE does not have an "Open file" option. He just wanted to open a file to edit it and couldn't do it. I showed him that the option was "Edit file", not "Open file". He never touched TheIDE again, nor U++ or the project.

Hello cbpporter,

Good catch :lol: - "Edit file" is now called "Open file".

Sincerely,
Klugier

Sorry, but reverted. Creates the wrong impression that we are somehow 'opening/editing/saving/closing' files, which is not true.

Mirek

Subject: Re: U++ 2017 beta
Posted by mirek on Thu, 05 Jan 2017 07:38:00 GMT
View Forum Message <> Reply to Message

Tom1 wrote on Wed, 04 January 2017 15:29
@Mirek: I bet this will divide opinions: My opinion is that I should be able to trust on Format to use decimal point regardless the region. I.e. the way it has always been with WIN and X11. If region compatible Formatting output is desired, that should be addressed through e.g. some "FormatRegional()" API or some optional formatter flags which are disabled by default.

Commited, please check!

Mirek

Subject: Re: U++ 2017 beta
Posted by Tom1 on Thu, 05 Jan 2017 08:17:51 GMT
View Forum Message <> Reply to Message

Hi Mirek,

Now Format() works on GTK like on Windows and X11. (Note to other readers: sprintf and Sprintf still follow the GTK locale for decimal separator, which is not consistent with Windows/X11 behavior.)

Thanks and best regards,

Tom
