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.