
Subject: should ultimate++ support dynamically linked libraries?

Posted by [piotr5](#) on Mon, 09 Dec 2013 19:38:16 GMT

[View Forum Message](#) <> [Reply to Message](#)

I stumbled upon this:

http://www.akkadia.org/drepper/no_static_linking.html

and it made me think, maybe ultimate++ isn't all that good when it comes to dynamically linked stuff? several external packages are getting built statically into u++ apps, upgrading them in the system will not upgrade them in u++. this is an important thing to consider. also I didn't think of how nice it must be for attackers when everything has a static address. but this and the other arguments on that site IMHO are not really important for c++ development. even without namespace, templated class-members are unlikely to name-clash. randomized data and code address could probably be implemented with some specialized lib. (create several functions with a template and overwrite them randomly at run-time.) also that site nicely says that within the same project you'd better use static linking.

however, the idea basically assumes all the libs you will use are available as u++ packages, therefore no support for external help-systems and no priority to scanning headers outside of the nest. also building and installing of shared libs isn't that easy to accomplish. should this approach be changed?

should we switch to the no-static paradigm?(total votes: 8)

yes 2/(25%)

no 4/(50%)

don't care 2/(25%)

Subject: Re: should ultimate++ support dynamically linked libraries?

Posted by [mirek](#) on Tue, 10 Dec 2013 14:35:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

piotr5 wrote on Mon, 09 December 2013 14:38 I stumbled upon this:

http://www.akkadia.org/drepper/no_static_linking.html

and it made me think, maybe ultimate++ isn't all that good when it comes to dynamically linked stuff? several external packages are getting built statically into u++ apps, upgrading them in the system will not upgrade them in u++.

Contra-argument is that upgrading them would not break existing apps...

However, in POSIX, U++ is using external dynamically linked apps where possible (e.g. SSL, zlib, ...). What is not quite supported is e.g. transforming U++ itself into dynamic library. It is mostly because C++ API is relatively hard to maintain and we prefer modularity there.

Mirek

Subject: Re: should ultimate++ support dynamically linked libraries?

Posted by [Novo](#) on Tue, 10 Dec 2013 21:50:48 GMT

[View Forum Message](#) <> [Reply to Message](#)

At some point Ubuntu got a new version of libc (I believe that was version 12.04), and my shared hosting provider kept the old one, so I had to create a VM with older version of Ubuntu to be able to compile and run my app with my hosting provider. I personally would prefer to be able to link it 100% statically on Linux.

And dynamic libraries is a back door for hackers. They allow to install hooks on Windows and use LD_PRELOAD on Linux.

Upp apps are small. Static linking makes them also fast.

Subject: Re: should ultimate++ support dynamically linked libraries?

Posted by [akspring](#) on Sun, 26 Jan 2014 12:37:46 GMT

[View Forum Message](#) <> [Reply to Message](#)

Novo wrote on Tue, 10 December 2013 22:50

And dynamic libraries is a back door for hackers. They allow to install hooks on Windows and use LD_PRELOAD on Linux.

Upp apps are small. Static linking makes them also fast.

From what I understand, dlls, vlc controls, and anything like chromium and ie controls are equivalent to handing your computer over to military trained cyborg ninja assassins from the future. You may as well forget about security if you include these common place controls. And they are everywhere.

I like to think U++ apps by themselves would be safe to use, but its really just a guess. I use Windows, so security for me and everyone else is really a joke. Same with Linux.

I don't think anyone cares about that much, so whats left is what will make U++ great? - I think its size, speed, and portability. With 7z/LZMA2, I can get a 15mb U++ application with debug info down to 1.8mb.

And 7z is really garbage when compared to costly closed source compression apps, so I bet I could have a self unpacking executable with debug info in under a megabyte. And without debug info my executables are about 3mb, 2mb with Windows compression, and under a megabyte with LZMA2. Just imagine if I paid for a high quality compression app. 500kb?

Its really nice to see something effective and fast, have many nice features like SQL and File Systems and be able to use it on any platform without needing something like I don't know, .NET? What are they doing now 250mb for the framework? I thought 30mb was a lot and that was just with 2.0

With U++ no matter how much I modulate my dlls, the only thing people will have to do is replace

their single executable and my next big thing is ready to go. No other advertised big name open source cross platform software can compete like U++. My apps are fast, portable, lightweight, ready to use, and the Core framework gives me plenty of functionality. And my compile times are amazing. I feel it delivers in many ways where standard C++ IDEs fail as well as compensates for many C#.NET features I want, without the pain of the managed and restrictive CLI. I dropped CodeBlocks and haven't looked back
