Subject: SO for Draw, Esc and other packages
Posted by Shire on Sat, 09 Sep 2006 12:09:40 GMT
View Forum Message <> Reply to Message

Does standard libraries allow dynamic module design (dll on Windows)?
Many classes have exports inline methods with access to private and protected members. Linking with these classes causes error.

Draw package depends on plugins/bmp, but plugins/{bmp,jpg,png,tiff} depends on Draw by objects StreamRaster and StreamRasterEncoder.
Is possible to join plugins/bmp and Draw?

IMHO, dll design will be good for solutions with many executables to prevent overhead in common code.

Subject: Re: SO for Draw, Esc and other packages
Posted by mirek on Sat, 09 Sep 2006 13:55:18 GMT
View Forum Message <> Reply to Message

Shire wrote on Sat, 09 September 2006 08:09Does standard libraries allow dynamic module design (dll on Windows)?
Many classes have exports inline methods with access to private and protected members. Linking with these classes causes error.

It does, or better said, it did... I am afraid this was not checked for quite a long time. But I remember that once it worked

Anyway, AFAIK, inline methods should not (did not) cause problems.

Quote:
Draw package depends on plugins/bmp, but plugins/{bmp,jpg,png,tiff} depends on Draw by objects StreamRaster and StreamRasterEncoder.
Is possible to join plugins/bmp and Draw?

Well, more things to solve...

Quote:
IMHO, dll design will be good for solutions with many executables to prevent overhead in common code.

Yes, but... also introduces .dll hell and often slows down startup times. So far, easy maintainace of apps was major design criteria.

I am maintaining more than 20 U++/Oracle apps for my main customer, which already could be

classified as "many executables". Anyway, they occupy about 70MB of hard disc space. You can find many ".dll modularised" apps that occupy significantly more. The idea that changeing single .dll would break more than single app would be a nightmare.

Actually, if you plan to develop some desktop environment based on U++ (a good idea , which IMO is the only place where dynamic libraries make sense, I would rather tried another approach to the problem - "join" all applets into single binary and make small "invokers" to call it. BTW, afaik, this is what trolltech is using for Qtopia...

OK, enough ranting (as you see, I really despise .dlls), I am adding to ToDo that we should reestablish SO compilation.

Mirek

Subject: Re: SO for Draw, Esc and other packages
Posted by Shire on Sat, 09 Sep 2006 14:58:20 GMT
View Forum Message <> Reply to Message

Quote:Anyway, AFAIK, inline methods should not (did not) cause problems.

These methods inlines in calling module and tries to access private members from other module. Static linking hides this problem.

Quote:Yes, but... also introduces .dll hell and often slows down startup times. So far, easy maintainace of apps was major design criteria.

dll hell appears when libraries became system-wide and shared between many untrusted vendors (like System32 directory on Windows). Local dll per version with correct name does not make hell. First startup time increases, but following start of other executable, which uses common dll, will be much faster - dll will be in cache.

Quote:The idea that changeing single .dll would break more than single app would be a nightmare.

Well, don't change dll Make only bugfix, binary compatible with previous, and you never see nightmares. Significant changes must increase version number and change dll name.

Quote:as you see, I really despise .dlls

When develop environment with dynamic/plugin architecture, is important to share common environment between all parts. Dll (or .exe exports) is good solution in this case.

Thank you for answer.

Subject: Re: SO for Draw, Esc and other packages
Posted by mirek on Sat, 09 Sep 2006 15:06:15 GMT
View Forum Message <> Reply to Message

Quote:
Significant changes must increase version number and change dll name.

Significant changes occur almost each week - I mean changes that break .dll compatibility - this can happen in C++ way too easily....

Quote:

When develop environment with dynamic/plugin architecture, is important to share common environment between all parts. Dll (or .exe exports) is good solution in this case.

Yes, plugins are the only excuse for .dlls

Even so, I would not hesitate not to share U++ library even for plugins. It is 800KB after all...

Mirek

Subject: Re: SO for Draw, Esc and other packages
Posted by Shire on Sat, 09 Sep 2006 23:33:19 GMT
View Forum Message <> Reply to Message

Quote:Significant changes occur almost each week - I mean changes that break .dll compatibility - this can happen in C++ way too easily....

Yes, I agree... But IMHO, "each week" is too often for one module. This can be if library is in alpha/beta stage. Stable release needs only small, non destructable fixes.

Quote:Even so, I would not hesitate not to share U++ library even for plugins. It is 800KB after all...

I'm not so worry about size (but 800KB overhead per module is evil). I'm worry about situation when modules begin to exchange common objects and each module have another version of common library...

Let's create polling?

Subject: Re: SO for Draw, Esc and other packages
Posted by mirek on Sun, 10 Sep 2006 07:29:22 GMT
View Forum Message <> Reply to Message

Quote:
Yes, I agree... But IMHO, "each week" is too often for one module. This can be if library is in alpha/beta stage. Stable release needs only small, non destructable fixes.

Well, we do not have one module (that makes it even worse).

Even much worse thing is that you would have carefully think about what fix in destructable and what is not. With static linking, C++ compiler in most cases decides for you...

Static linking is simply way more flexible for C++.

Quote:
I'm not so worry about size (but 800KB overhead per module is evil). I'm worry about situation when modules begin to exchange common objects and each module have another version of common library...

I would never base plugin interface on full C++ classes - that is way too fragile. MAYBE on stable and simple C++ interfaces.

Mirek
