Subject: Re: Does the provided upp.spec works for you and on which distro?
Posted by cbpporter on Tue, 02 Sep 2008 12:19:43 GMT
View Forum Message <> Reply to Message

Don't get thing wrong please. We both appreciate and need help in general, but a lot of you suggestions are related to parts which actually work. It is not as much resistance to change, as it is a lack of interest and man-power to improve something which works quite well rather than focus on the parts which don't work too well.

Regarding creation of .so: I believe we are open to the usage of .so. I don't know who really needs it, but it would definitely be great to have a dll version too. It would also make things easier for those who don't want to use TheIDE. If you're willing to do this, be my guest. I will do my best to help you, but I'm not interested in this on my own, because it would not benefit me: I distribute 2-3 binary systems: one is the installer, and the rest are the application executable. With only 2-3 exe's, code sharing between .so is not that great on memory footprint reduction, and I get the added disadvantage of slower function calls.

But in practice it is not that easy. First of all, we've got all the templates that can't be embedded in the .so. Then there's the versioning, API/ABI issue. We would need a rigorous system to track ABI

in const because it did not change the status of the object. We added const to be correct, but this definitely would have broken compatibility with binary .so because of name mangling.

So I think that dll's are a good idea, but only if you build a version specific to you application and ship it with your application. Getting a general .dll is a lot harder. If you are aware of a solution that would solve such issues, then please tell us.

Quote:Cpp an .h file names policy (all lower case for better portability): No need for policy, our programmers are used to current file names.
I'm sorry, but I consider lower case names as deprecated as I do non-case sensitive paths. Even under Windows I always use proper cased names for files, and also except to get such names. So "Ctrl.h" is not a caseless or lower case name which was written with a capital "C" because Windows is not case-sensitive, it is really called "Ctrl.h" and referring to it as "ctrl.h" is an error IMO. And since all Unixes are case sensitive and Windows is case neutral, I don't think we have any issues left with this scheme. Also, if we change to lower case on Unix, all programs will cease to compile until we manually modify all include clause.

The same applies to using more longer and intuitive names. How can we change the name of a file and except old code to compile. How can we change the name of a class and still except to link with the .so that we are going to produce?

Quote:Something easier than "Assembly/Nest/Package": "Assembly/Nest/Unit"?
Actually I don't care what they are named, but I kind of like their meaning and implication on code layout. A little clean up on interface level and I'm sure things would seem a little cleaner for new people. Also, "assembly" sounds familiar to .Net crowd, but has different semantics, so a change would not be a bad idea.

Doxygen: I'm all for reusing technology which works (i.e. doxygen) and wouldn't mind using it instead of Topic++. Topic may be more intimately tied to A++, but I don't think this mattes that much since docs are not autogenerated.

On the other hand, using doxygen would increase compilation times. I'll go crazy if I use another 0.5 second time per build. I would hate to have to abandon both the language and framework that I have grown to love and switch over to .NET because of trivial issues like compilation speed which should have been fixed decades ago.

And docs are coming along .

On the other hand, I don't see why we should argue about little issues like filenaming, when we need a better C++ parser for example. Code sequences that short out Assist++ are hard to circumvent and they spread like a virus.

Out of all these things, I think that .so should be a priority if you still need it. If we get that process working, adding support for TheIDE to build a dynamic lib and link your project to it would definitely be doable.