Subject: Re: Few questions
Posted by mr_ped on Mon, 08 Nov 2010 08:44:31 GMT
View Forum Message <> Reply to Message

Boost includes can be maybe fixed just by adjusting build method?
Still if you want some of those built parts, create a package.

Quote:
I am still very surprised that this was not already done considering how massive and useful Boost is.

Well, surprising, but it's like that, the boost libs are mentioned by U++ users very rarely.
I can't speak for others, but for me the whole STL and Boost were too complex to feel well with them, I was doing pure C++ for a long time (even doing my own containers on the fly when I needed them, but in my area of problems I did usually need just static pools and other simple concepts which were easy to manage without libs). That's what turned me to U++, it's so small (considering what it does), that I forced myself to learn a bit more then C++.

This is also maybe answering a bit of your "pushing the programmer" question. You look to be extraordinary bright person with very good memory and logic, so you can remember those extended syntax sugar of boost/spirit/Phoenix, as you demonstrate. For me it's burden, I would have very hard time to learn all those new options, and to use them on the fly from head. From what you do showcase it looks very good, and I can imagine to adopt it if I would use it daily, but if I keep using C++ just rarely like now, I can't afford to learn things like this, too complex/abstract for me, so it would take serious time to learn, and I would forgot it next 2 weeks without using it.
Everyone has different limits, skills and needs. (and once you have hammer in hand, many things looks suddenly like a nail)

So if you don't find U++ too much weird, and you can use it straightaway, it's just good for you (and I bet some people here will envy you a bit, including me).

Quote:Heh, so no multi-year-long review process then like I am used to then?
To bazaar not, you just ask for SVN access and commit, that's basically it.
To core the process is everything but years long. Usually you either have Mirek's attention and then it takes as much time as it's needed (from minutes to several iterations and changes over months), or you don't have his attention, then it can take longer or get forgotten at all. But usually changes to core are small, so easy to review and adopt.

We don't have time here for long review... the code either works and can be used, or doesn't work. If it turns wrong later, it will be refactored out anyway, so no point to hesitate too much, U++ is far from overly rigid system. There's serious will to not break backward compatibility just for the sake of change (because core people are running considerable amount of commercial apps on U++ and have to adapt them to every such change), but when something is broken by design, it gets fixed, no matter how much it breaks old apps.

Quote:So the amount of packages is relatively low overall then? I shall work on fixing that as I get needs.
Yes, new packages pop out mostly when somebody needs it. Although some of bazaar stuff was

done just for experimental fun, but most of it is result of somebody using it at his work.

Quote:I do question one design aspect though, why use an unused integer as a differentiator between pick/move and copy constructors and so forth?
This is clearly question at Mirek, although I'm not sure if he's reading such long posts carefully enough to notice.
I don't mind your copy constuctor way either, looks quite natural to me.
I wouldn't worry about registers pollution, didn't check the assembly of compilation, but I think most of the modern compilers can optimize that part out, at least I think somebody claimed so.
Anyway, I didn't hit performance problems there yet (usually when you need deep-copy, it will take so much CPU time, that handling one int parameter more would be "for free" even if optimizer would be unable to cut it out? IMHO)

Try to use TheIDE a bit too if you can? I think it may irritate you a bit, but the feedback would be very welcome, also it may help you to understand how many people U++ use and look at. (as an RAD GUI app dev tool, so they can produce some forms/reports with few clicks and put it into DB .. that maybe also partly answers why so few people miss those boost goodies)

Clang was unable to compile U++ sources for long time, that's why the support is lacking. Since their new improvements I think U++ will be extended in next months to fully support it.

About Assist++ rewrite... the performance can be the decisive factor. Assist++ is outperforming most of the C++ parsers/analyzers because of it's flaky (or almost non existent) macro/template support. Unless you have much more clever parser, you can't do much better as Assist with the same performance. Let's say you don't have much clever parser, just more valid one, so more slower. In such case I think TheIDE would have to extend some caching schemes (right now the Assist doesn't use any permanent cache at all IIRC, just parse all the source upon start of TheIDE and selecting the package to create temporary cache). So it's not only about parser replacement, it would require the whole Assist++ design change in case the parser would be not fast enough.
(I personally did use commercial extension Visual Assist years back, and since then I didn't see anything so good. Assist++ is not bad, it helps me a bit, but it's not as amazing, plus the global/local references separation is not to my taste either, but I got used to it now, and can work effectively even in TheIDE)