Subject: NTL and C4251
Posted by jmansion on Mon, 24 Apr 2006 12:26:59 GMT
View Forum Message <> Reply to Message

I'm looking at Microsoft's missive on exporting classes that expose STL classes at http://support.microsoft.com/kb/q168958/

It looks like some of the classes can't be instantiated because they have nested classes.

Does NTL suffer from similar problems?

I'm not too committed to STL with the possible exception of wanting to support std::string in some places.

James

BTW is NTL effectively homed here now? The 'obvious' home at http://www.ntllib.org/ looks rather moribund, with 'latest news' stuck in mid 2003.

Subject: Re: NTL and C4251
Posted by mirek on Mon, 24 Apr 2006 13:26:03 GMT
View Forum Message <> Reply to Message

Well, I am not sure, for several reasons.

First, article seems to be dealing with vc6.0- issues. U++/NTL does not compile with pre-7.1 compilers anyway.

Second, NTL itself does not support DLLs outside TheIDE. AFAIK TheIDE has specific hacks to compile process that makes those ugly declspec issues for importing/exporting stuff unnecessary (EXPORTS are automatically generated, just like in linux/.so - no explicit export/import declarations are necessary).

Anyway, to tell the truth, U++ is designed to provide standalone executables, no .dlls at all.

And yes, this is the right place for NTL

Mirek

Subject: Re: NTL and C4251
Posted by jmansion on Mon, 24 Apr 2006 13:34:30 GMT
View Forum Message <> Reply to Message

Well, I can tell you that the problem is retained in VC2003. Haven't checked 2005 yet.

I'll have to extract NTL and have a look myself, then. I'm looking for something I can bundle into my source tree to remove issues with different STLs anyway (and so I can force it all to use my own memory allocator stuff too).

Thanks anyway.
James

BTW have you considered working with Walter Bright to use Digital Mars C++ under Win32?

Subject: Re: NTL and C4251
Posted by jmansion on Mon, 24 Apr 2006 14:04:41 GMT
View Forum Message <> Reply to Message

>TheIDE has specific hacks to compile process that makes those
>ugly declspec issues for importing/exporting stuff unnecessary
>(EXPORTS are automatically generated, just like in linux/.so -
>no explicit export/import declarations are necessary).

Oh - I'd hardly call the situation on Linux ideal, hence the work the gcc team has been doing to make control of visibility much easier and the performance issues seen in OOo and KDE.

>Anyway, to tell the truth, U++ is designed to provide
>standalone executables, no .dlls at all.

Ouch. That's really a big disadvantage - it cuts out the use of SWIG to help automate subsystem testing etc or to provide services to JNI, P/Invoke or even to dynamically load extensions. I know I've typically released monolithic in-house binaries built static, but I'd not want to be forced to do this.

I'll just have to look at NTL and uSTL more closely, I guess. And that's probably answered whether I could move to U++ as my main development env. Still handy for its target GUI apps though I guess.

Subject: Re: NTL and C4251
Posted by mirek on Mon, 24 Apr 2006 14:27:02 GMT
View Forum Message <> Reply to Message

jmansion wrote on Mon, 24 April 2006 10:04>TheIDE has specific hacks to compile process that makes those
>ugly declspec issues for importing/exporting stuff unnecessary
>(EXPORTS are automatically generated, just like in linux/.so -
>no explicit export/import declarations are necessary).

Oh - I'd hardly call the situation on Linux ideal, hence the work the gcc team has been doing to make control of visibility much easier and the performance issues seen in OOo and KDE.

I agree that performance sucks with dynamic loading of .so. However, I think this is rather the problem of that specific implementation than the problem of visibility only.

In any case, standalony binary (like one produced by U++) beats them all

Quote:
>Anyway, to tell the truth, U++ is designed to provide
>standalone executables, no .dlls at all.

Ouch. That's really a big disadvantage - it cuts out the use of SWIG to help automate subsystem testing etc or to provide services to JNI, P/Invoke or even to dynamically load extensions. I know I've typically released monolithic in-house binaries built static, but I'd not want to be forced to do this.

Well, the real meaning of standalone here is that you can ship U++ binary without any aditional .dll and it works on everything since Win95 or on most current Linux 386 distros. Anyway, on Linux, the build in fact is shared (against a limited number of system .so - namely glibc, stdc++, xlib and xft).

You can however use 3rd party .dlls, you can build .dlls (even OLE/COM components), and you can even build in "SO" mode where all library subsystems are build as dll.

Quote:
And that's probably answered whether I could move to U++ as my main development env.

Well, as always, you cannot satisfy everybody.... The mission goals are low development / maintainance costs. DLL hell does not contribute to either

Mirek

Subject: Re: NTL and C4251
Posted by unodgs on Mon, 24 Apr 2006 17:29:00 GMT
View Forum Message <> Reply to Message

jmansion wrote on Mon, 24 April 2006 09:34

BTW have you considered working with Walter Bright to use Digital Mars C++ under Win32?

I tried to build upp libs with dm c++. Unfortunately even with newset beta it fails to compile main

upp lib - core (problems with templates).
Hopefuly Walter is fixing his compiler to be able to compile boost so maybe it will help to compile upp too (main reason I want to use dmc is speed of compilation...)

Subject: Re: NTL and C4251
Posted by mirek on Mon, 24 Apr 2006 19:29:38 GMT
View Forum Message <> Reply to Message

unodgs wrote on Mon, 24 April 2006 13:29jmansion wrote on Mon, 24 April 2006 09:34

BTW have you considered working with Walter Bright to use Digital Mars C++ under Win32?

I tried to build upp libs with dm c++. Unfortunately even with newset beta it fails to compile main upp lib - core (problems with templates).
Hopefuly Walter is fixing his compiler to be able to compile boost so maybe it will help to compile upp too (main reason I want to use dmc is speed of compilation...)

I hope some day we will gather enough resources to resolve this problem forever (I mean, C++ compiler is HARD thing to implement, but that makes it wonderful task to do, does not it?

Mirek

Subject: Re: NTL and C4251
Posted by unodgs on Mon, 24 Apr 2006 20:44:42 GMT
View Forum Message <> Reply to Message

luzr wrote on Mon, 24 April 2006 15:29unodgs wrote on Mon, 24 April 2006 13:29jmansion wrote on Mon, 24 April 2006 09:34

BTW have you considered working with Walter Bright to use Digital Mars C++ under Win32?

I tried to build upp libs with dm c++. Unfortunately even with newset beta it fails to compile main upp lib - core (problems with templates).
Hopefuly Walter is fixing his compiler to be able to compile boost so maybe it will help to compile upp too (main reason I want to use dmc is speed of compilation...)

I hope some day we will gather enough resources to resolve this problem forever (I mean, C++ compiler is HARD thing to implement, but that makes it wonderful task to do, does not it?

Mirek

Once I had to write something similar to c compiler (using bison and flex) and it even worked (it produced 16-bit asm output for tasm). I remeber that I spent lots of houres working on it..

Probably it would take you about 3 years to implement full C++ compiler (and once you get it you will have to reimplemnt this to be compatabile with new 0x standard...)
Of course I wish you a luck !

PS: Personally I would switch to sth similar to D but without gc and with multiple inheritance (That may seem silly but the worst thing in D for me is that I have to use new syntax to create object..)

Subject: Re: NTL and C4251
Posted by mirek on Mon, 24 Apr 2006 20:49:11 GMT
View Forum Message <> Reply to Message

Quote:
Probably it would take you about 3 years to implement full C++ compiler

Yes, I guess that estimate is about to be right...

Well, let us finish U++ first

Mirek

Subject: Re: NTL and C4251
Posted by gprentice on Tue, 25 Apr 2006 04:41:09 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 02:04>

>Anyway, to tell the truth, U++ is designed to provide
>standalone executables, no .dlls at all.

Ouch. That's really a big disadvantage - it cuts out the use of SWIG to help automate subsystem testing etc or to provide services to JNI, P/Invoke or even to dynamically load extensions. I know I've typically released monolithic in-house binaries built static, but I'd not want to be forced to do this.

Judging by Mirek's response, it seems he meant U++ is designed to allow you to avoid .dlls, not that you can't use and create them.

There's even a package template for creating a dll but it seems to not be in the latest U++ distribution ...

Graeme

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 07:45:01 GMT
View Forum Message <> Reply to Message

Actually, thinking about the issue, the real important limitation is that you cannot use U++/C++ interfaces at .dll levels except slightly experimental SO mode. Example would be U++ application with U++ plugins - in practice, while both using U++, each should contain it own copy of the library and comunicate using some C based interface.

Which makes quite a sense to me - C++ interfaces across dll modules are always problematic. Think compiler name mangling differences or object layout problems....

Mirek

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 08:17:48 GMT
View Forum Message <> Reply to Message

> (main reason I want to use dmc is speed of compilation...)

Indeed - its the fastest compiler I've ever used by some margin. Haven't tried latest Borland stuff though.

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 08:23:59 GMT
View Forum Message <> Reply to Message

> I hope some day we will gather enough resources
> to resolve this problem forever

Perhaps start with LVM, and 'just' replace the front end.

There are plenty of tools out there that can help out which are a tad better than flex and yacc - consider ragel for scanners, d-parser, lemon etc etc.

There's a choice of non-GPL free STLs now, and plenty of C library code that's non-GPL free.

James

Subject: Re: NTL and C4251
Posted by gprentice on Tue, 25 Apr 2006 08:24:34 GMT
View Forum Message <> Reply to Message

luzr wrote on Tue, 25 April 2006 19:45Actually, thinking about the issue, the real important limitation is that you cannot use U++/C++ interfaces at .dll levels except slightly experimental SO mode. Example would be U++ application with U++ plugins - in practice, while both using U++, each should contain it own copy of the library and comunicate using some C based interface.

Which makes quite a sense to me - C++ interfaces across dll modules are always problematic. Think compiler name mangling differences or object layout problems....

Mirek

But if the plugin and executable used the same compiler and build options etc. it probably work wouldn't it? - at least with VC++ on Windows? Getting the plugin to match the executable would be a problem though as you say.

Would event dispatch work with the dll plugin?? - would the dll need its own thread with an event loop if the plugin had a visual component?

Would the dll need a different version of GUI_APP_MAIN ?

Graeme

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 08:26:24 GMT
View Forum Message <> Reply to Message

If anyone is interested in this sort of thing and hasn't yet read 'Imperfect C++' then I recommend it.

See http://synesis.com.au/publishing/imperfect/cpp/.

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 08:41:50 GMT
View Forum Message <> Reply to Message

Quote:
But if the plugin and executable used the same compiler and build options etc. it probably work wouldn't it? - at least with VC++ on Windows?

Quite a lot of variables.... (add "VC++ the same version"). However, it would work - that is the SO mode Well, the important thing there is that there are no explicit exports anywhere in U++ code (as mentioned before).

Quote:
Would event dispatch work with the dll plugin?? - would the dll need its own thread with an event loop if the plugin had a visual component?

All I can say is that it works with U++ based OLE controls .dll used from MFC application. There were things to fix due to crazy MFC design, but it works.

Quote:
Would the dll need a different version of GUI_APP_MAIN ?

Yes, most likely yes. However, you are not required to use GUI_APP_MAIN, that is more or less a fix to save you platform specific #ifdefs.

BTW, I may agree with "main hijacking" complaints about those APP_MAIN. However, after writing that platform specific initialization for 20th time (in examples), I simply got bored and lazy...

Mirek

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 08:45:51 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 04:23.
There are plenty of tools out there that can help out which are a tad better than flex and yacc - consider ragel for scanners, d-parser, lemon etc etc.

I never believed in compiler generators. I guess that hand-written parsers are much easier to implement and maintain.

However, syntax parsing is the minor problem (I believe that I can write C++ _syntax_ parser in less than month). The real troubles come with semantics...

Mirek

Subject: Re: NTL and C4251
Posted by unodgs on Tue, 25 Apr 2006 11:54:21 GMT

View Forum Message <> Reply to Message

luzr wrote on Tue, 25 April 2006 04:45jmansion wrote on Tue, 25 April 2006 04:23.
There are plenty of tools out there that can help out which are a tad better than flex and yacc - consider ragel for scanners, d-parser, lemon etc etc.

I never believed in compiler generators. I guess that hand-written parsers are much easier to implement and maintain.

However, syntax parsing is the minor problem (I believe that I can write C++ _syntax_ parser in less than month). The real troubles come with semantics...

Mirek

True. That's why gcc team replaced their parsers with hand-written ones.
Semantics... - c++ draft stnadard from 96 in html has over 2mb. Probably more than half of it is about semantic...

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 12:07:16 GMT
View Forum Message <> Reply to Message

Well, I agree that you can have more control over error reporting with a roll-your-own, but personally I would still look to a generator for the bits that can be easily automated, so then I can concentrate on building an AST that makes it as easy as possible to build the symbol tables and other structures, and having written compilers by hand before I definitely welcome these tools.

I find CoCo/R the easiest to use for simple tasks but I doubt its really up to C++ unless you you an external scanner. You have to defer a lot of C++ analysis anyway so the grammar isn't going to resolve identifiers to types as you go anyway.

Clearly, modern C++ semantics in practice are hugely dependent on the template and inline handling and getting the type searches right in such cases - and in making sure that error messages are sane and the generators won't help you here, but using LLVM as the back end surely will.

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 12:20:42 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 08:07I would still look to a generator for the bits that can be easily automated

Actually, I never got that part about "easily automated". With well designed C++ code, it is IME as easy to write descent parser right away as to create syntax description for "code generator" (or even to describe grammar in any formal way).

BTW, when developing Esc interpreter, I have never had any formal grammar for the language - descent parser C++ code is as good as the formal grammar description.

Quote:
AST
CoCo/R

What are these?

Quote:
sane and the generators won't help you here, but using LLVM as the back end surely will.

Yes, I agree about using LVM (if that is what I think: "platform independent assembler-like language"). Yet another option is to produce plain C....

Mirek

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 12:33:17 GMT
View Forum Message <> Reply to Message

>BTW, when developing Esc interpreter, I have never had any >formal grammar for the language - descent parser C++ code is >as good as the formal grammar description.

Ugh! Do your spurs jingle!

I think that's a mistake.

AST -> Abstract Syntax Tree

CoCo/R -> a recursive descent parser generator. Pat Terry had a book on the C/C++ version which is now out of print so the book is online. The Java and C# versions make life a doddle, and Pat has a new (and very good IMO) book on them.

Info is here: http://www.ssw.uni-linz.ac.at/Research/Projects/Coco/

I utterly dispute any suggestion that its not a timesaver. Its just so easy to iterate the language design itself, and it makes writing 'little languages' a pleasure. The output is quite readable, and

will probably not be too different to what you write by hand.

I still favour a hand built scanner and LEMON for my current project, because I think I can go fastest this way. Tho I harbour a concern that actually my carefully written scanner will be only slightly faster than I'd get from re2c or ragel.

We'll have to agree to differ.

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 13:03:58 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 08:33
CoCo/R -> a recursive descent parser generator. Pat Terry had a book on the C/C++ version which is now out of print so the book is online. The Java and C# versions make life a doddle, and Pat has a new (and very good IMO) book on them.

Info is here: http://www.ssw.uni-linz.ac.at/Research/Projects/Coco/

Ah, yes, I think that it is exactly what I am speaking about

I have gone through examples and I believe that the grammar description there is as long as the descent parser I would write for that...

OK, I believe that it can save some time as long as you have description ready before starting the job...

Mirek

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 13:19:56 GMT
View Forum Message <> Reply to Message

>OK, I believe that it can save some time as long
>as you have description ready before starting the job...

You must type a lot faster than I do then. I think its actually most handy when evolving the grammar. I like to write the grammar, and then write some candidate inputs and see if they feel right, for the things I want to express. And then change things around a bit and try again.

Having that sort of tool makes it easy - I don't add the AST code and semantic actions until I'm happy with the structure, and there's no way I could iterate that fast even using Java or C# with a hand-written parser, let alone C++.

I can't believe that anyone would argue that parser creation isn't a whole lot faster with a good generator tool, particularly if you care about whether your off-the-cuff implementation actually handles an ambiguous grammar in a particular way through accident - I'd rather have a formal grammar specification and a tool that can warn me of ambiguities. Not least, it also makes it easier to create alternate implementations in other languages.

Its certainly the case that a hand-written parser can be faster and have better diagnostics support, if you work hard enough at it.

James

Subject: Re: NTL and C4251
Posted by unodgs on Tue, 25 Apr 2006 13:43:07 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 09:19>OK, I believe that it can save some time as long
>as you have description ready before starting the job...

You must type a lot faster than I do then. I think its actually most handy when evolving the grammar. I like to write the grammar, and then write some candidate inputs and see if they feel right, for the things I want to express. And then change things around a bit and try again.

James

Notice that grammar for C++ is already definied and approved, so there is no need for expeiments with grammar!

Subject: Re: NTL and C4251
Posted by jmansion on Tue, 25 Apr 2006 14:06:21 GMT
View Forum Message <> Reply to Message

>Notice that grammar for C++ is already
>definied and approved, so there is no
>need for expeiments with grammar!

I disagree. The grammar as defined cannot be implemented directly - it requires infinite lookahead to determine which symols refer to types and which to variable declarations. There may be other issues too.

You need to transfrom it to one that can build a basic parse tree and then do a lot of the work subsequently, even in a plain class declaration. Personally I would value an ability to iterate

cheaply at that stage, where I'm working with something that has a reasonable declarative correspondence to the ANSI 'grammar', rather than C++ code, but clearly YMMV.

James

Subject: Re: NTL and C4251
Posted by mirek on Tue, 25 Apr 2006 18:10:11 GMT
View Forum Message <> Reply to Message

jmansion wrote on Tue, 25 April 2006 09:19
You must type a lot faster than I do then.
I can't believe that anyone would argue that parser creation isn't a whole lot faster with a good generator tool

Just for reference, I am speaking about code like this:

http://upp.sourceforge.net/reference$CParser.html

Quote:
I'd rather have a formal grammar specification and a tool that can warn me of ambiguities.

Not least, it also makes it easier to create alternate implementations in other languages.

I guess above two points are quite valid - I have to agree with those advantages.

Quote:
Its certainly the case that a hand-written parser can be faster and have better diagnostics support, if you work hard enough at it.

Couriously, I always thought that the only problem of my hand-written parsers is somewhat worse perfomance compared to generated ones

Mirek

Subject: Re: NTL and C4251
Posted by jmansion on Wed, 26 Apr 2006 08:02:28 GMT
View Forum Message <> Reply to Message

> Couriously, I always thought that the only
> problem of my hand-written parsers is somewhat
> worse perfomance compared to generated ones

Parser or scanner?

Anyway - are you using recursive descent throughout? (Sorry, haven't looked at CParser yet).

I seem to remember that when we wrote the Acorn BBC Micro C compiler, it got a big boost when the expression parser was moved from a recursive descent routine to a precedence engine.

But you'd have to ask Dave Christensen because that was a very long time ago, and I think he wrote that bit.

Anyway - its probably not worth worrying about this, 'cos I doubt that anyone has stomach for an implementation now. And I'd rather you guys wrote the MacOS port and integrated a real web widget and spreadsheet anyway.

Subject: Re: NTL and C4251
Posted by mirek on Wed, 26 Apr 2006 08:10:51 GMT
View Forum Message <> Reply to Message

jmansion wrote on Wed, 26 April 2006 04:02
Anyway - are you using recursive descent throughout? (Sorry, haven't looked at CParser yet).

CParser is a little bit misleading name - in fact it is primitive lexical tool for languages with similar lexical structure to C (I mean, same identifier, literal and comment rules).

Quote:
Anyway - its probably not worth worrying about this, 'cos I doubt that anyone has stomach for an implementation now.

Well, it is relevant to some degree even now as there is mixed heurestic-descent parser to get Assist++ info out of C++ sources in TheIDE... (of course, heurestic part IMHO makes it hard to implement using code generator anyway

Mirek
