Subject: How BLITZ works?
Posted by hojtsy on Wed, 25 Jan 2006 13:20:42 GMT
View Forum Message <> Reply to Message

I see that the Blitz (compiltation speedup system) creates composite source files where several cpp files are included. Are there anything else that the Blitz does? Do you have some approximate numbers for how much is the compilation speed improvement? Would it be possible to use Blitz as a separate application to speed up the compilation of a 1000-file project, which is not developped in the TheIde?

Subject: Re: How BLITZ works?
Posted by mirek on Wed, 25 Jan 2006 17:30:31 GMT
View Forum Message <> Reply to Message

Well, average .cpp file has say 500 lines, but includes 100000 lines of headers. That is why combining C++ files into single "blitz" file is such a good idea. However, interesting part of blitz is that it has to detect which files to combine and when... (it checks them for #ifdef include guards and also excludes files changed withing one hour ago - those files are likely to be worked on).

As for speed improvements, I think you can count on 4x speedup when using gcc/linux. Actually, compilation itself is even faster, but the linker spoils it (And Tom is still reluctant about implementing uld on linux In any case, it is not hard to measure speedup for yourself - just switch BLITZ off in output mode dialog.

Problem with BLITZ used outside TheIDE is that this system expects to compile C++ and extesively uses information from packages and .cpp files. It is e.g. impossible to use similar technique with traditional makefiles. In other words, in order to have BLITZ, you need the similar project organization like the one used in TheIDE.

Subject: Re: How BLITZ works?
Posted by hojtsy on Wed, 25 Jan 2006 18:36:30 GMT
View Forum Message <> Reply to Message

Can you tell more about how to detect which files can be combined, and how to select the macros which should be undefined? Or is this described somewhere already? My first thought is that all macro definitions should be gathered from the cpp itself and undefined after each cpp included. I think that macro definitions from the #included files should not be undefined, because the headers are included only once anyway if they are protected with included guards, as they should be.

Subject: Re: How BLITZ works?
Posted by mirek on Wed, 25 Jan 2006 19:21:37 GMT
View Forum Message <> Reply to Message

Well, by default, only .cpp files with include guards are qualified for blitz (this is easily checkable).

You can affect this using

#pragma BLITZ_APPROVE // to force the file to be BLITZed
#pragma BLITZ_PROHIBIT // to disallow

And you are correct about TheIDE going through the file, recording all #define(s) and placing conditional #undef(s) at the end of blitz file (#if(s) are ignored, that is why #undef(s) have to be conditional).

Actually, most of these actions are performed in header-dependency code and results are cached (per file timestamp).

I have planned to make Topic++ about BLITZ for quite a long time, but there is always something else to do.....

Respective code is in ide/Common/Hdepend.cpp and ide/Builders/CppBuilder.cpp.

BTW, BLITZ is by default active in debug mode only (where it has the most sense).

Subject: Re: How BLITZ works?
Posted by mirek on Wed, 25 Jan 2006 19:23:17 GMT
View Forum Message <> Reply to Message

Ooops. Correction: .cpp files that include only blitz-approved .h files (with guards or per pragma).

Subject: Re: How BLITZ works?
Posted by hojtsy on Wed, 25 Jan 2006 20:19:25 GMT
View Forum Message <> Reply to Message

From the code it seems that Blitz is not checking for other issues which could cause problems when compiled in the same object file, such as:
- global variables with file visibility (static)
- global functions with file visibility (static)
- classes with identical names defined in two different cpp files

Another possible issue could be that when you "blitz" together different cpp files, you are "infecting" every cpp file with the headers included in other files. These headers may contain some macro definitions which break other cpp files or headers.

So it seems that the prerequisite of blitz is that the code satisfies some stylistic requirements, which are not enforced by the compiler.

Subject: Re: How BLITZ works?
Posted by mirek on Wed, 25 Jan 2006 20:35:18 GMT
View Forum Message <> Reply to Message

Yes, that is correct. Nothing is perfect... (Just a note, you cannot use two classes with identic names - possible name clash when linking, as methods are always defined as non-static globals).

It is up to you to decide whether 4x times speedup is worth the trouble. And you can always switch BLITZ off, either using #pragma or in Output mode dialog (even on per-package basis).

From my perspective, U++ development in Linux would be nearly impossible without BLITZ... (there is a huge difference between 40 minutes or 10 minutes to compile TheIDE). And while it can affect your code in ways you mention, usually worst thing to happen is compile time error. In reality, most of C++ code works with BLITZ just fine out of box and rest can be fixed in minutes.

BTW, one thing I forgot to mention: You can, with some effort, resolve the issues with static variables in specific cases (U++ needs that to implement stuff linke INIT_BLOCK in Core/Defs.h):

BLITZ_INDEX__

is defined with specific number for each file included in BLITZ group. Somewhat ugly, but working...

Subject: Re: How BLITZ works?
Posted by hojtsy on Thu, 02 Feb 2006 14:14:47 GMT
View Forum Message <> Reply to Message

I made a change in a source file, which excludes that file from Blitz-ing, and looked at what was recompiled after this. I expected that the whole package needs to be recompiled because the set of files taking part in the Blitz-compilation was changed. But suprisingly only the changed file was recompiled. How does TheIde avoid multiple-definition errors in these cases? The same functions are defined in the object compiled from the old not-recompiled Blitz compilation, and the new object created after the change, aren't they?

Subject: Re: How BLITZ works?
Posted by mirek on Thu, 02 Feb 2006 19:34:16 GMT
View Forum Message <> Reply to Message

I am not 100% sure what scenario happened. Maybe that single file was excluded during previous build.

Actual blitz build is quite simple - you have a set of .cpp files to compile. You gather all of them not changed for more than hour and approved for blitz, combine them into single $blitz.cpp and replace them in the set of files to compile with that $blitz.cpp. $blitz.cpp is saved only if it is different from already existing one.

The rest of build is the same as without blitz, just performed with altered set of sources.

Normally, when you modify the file that was previously part of blitz, two files (from compiler perspective) are recompiled - $blitz.cpp, now shorter one file and the file you have modified.

Mirek

Subject: Re: How BLITZ works?
Posted by hojtsy on Thu, 02 Feb 2006 20:59:41 GMT
View Forum Message <> Reply to Message

I didn't know about the one-hour rule. That could be the reason why the file was originally excluded from blitz, so the blitz-set did not changed when I modified the file.
