Subject: Global variables in Upp
Posted by jerson on Sun, 12 Sep 2010 01:22:12 GMT
View Forum Message <> Reply to Message

Earlier http://www.ultimatepp.org/forum/index.php?t=msg&goto=284 87&#msg_28487

I have been able to finally solve the problem in my case. The behaviour is mainly due to the way Upp uses Blitz to compile. I think that behaviour needs to be documented somewhere.

Let us say, my package has 3 files, main.cpp, file1.cpp and file2.cpp All these files have a common Hfile that declares a global variable say gvar. Of course, it has the header locks in place to prevent multiple declarations.

On the first build after IDE is opened, blitz usually builds all the 3 files together and shows like so on the output pane
BLITZ : main.cpp file1.cpp file2.cpp
and the project builds fine.
Finer detail, Blitz compiles it in a single pass like this
Header file
main.cpp (global gets declared here as H file is in here first)
file1.cpp (H file gets locked out)
file2.cpp (H file gets locked out)

Now, if I touch any 1 of the 3 files, only that file gets compiled and the global variable gets declared again in the file which is compiled.
BLITZ file2.cpp (H file gets declared here as this is the only file being compiled now)

Now, during the link phase, the first build case links correctly.

In the second case, the linker has a problem. The header lock does not seem to work. This is because, in case1, main.cpp has the gvar already declared. Now, file2.cpp has gvar declared too since it was compiled all alone by BLITZ. So, I keep getting a multiple definition for gvar from the second build onwards.

How did I solve it? Simple. Declaration of gvar is removed from .H file and put in main.cpp. Now all other Cpp files refer to the variable as extern var.

in main.cpp for example MyStruct gvar;
in file1.cpp extern MyStruct gvar;
in file2.cpp extern MyStruct gvar;

Now, irrespective of how blitz compiles the files, gvar is always in main.cpp and the others know where to find the gvar.

I spent quite a bit of time figuring this out, so I hope my post will help other newbies like me.

Regards

Subject: Re: Global variables in Upp
Posted by dolik.rce on Sun, 12 Sep 2010 07:52:27 GMT
View Forum Message <> Reply to Message

Hi Jerson,

There is a nice helper for this kind of situation in U++. The macro GLOBAL_VAR (probably undocumented too). It creates a function returning static variable, so in effect it can be used as global variable accessible wherever you declare the function and BLITZ-safe. The usage is simple, e.g.:GLOBAL_VAR(int,gvar);
gvar() = 5; //setting
int b = gvar(); //getting
The price is typing the extra two parenthesis, but as global variables are not used very often, it should be not a big trouble.

Best regards,
Honza

Subject: Re: Global variables in Upp
Posted by jerson on Sun, 12 Sep 2010 08:12:09 GMT
View Forum Message <> Reply to Message

Hi Honza

Yes I realized a major part of the power in Upp is undocumented. I was looking through the code examples, references etc and figured out a few things. What makes it difficult is that one does not know where to look for such information since the help file has not been updated in quite a while.

If there would be some utility which can scan the source files and put all the function prototypes in one place, it will be a great help in finding such functions. Is this something that google code browser can help with?

Regards

Subject: Re: Global variables in Upp
Posted by dolik.rce on Sun, 12 Sep 2010 10:56:58 GMT
View Forum Message <> Reply to Message

jerson wrote on Sun, 12 September 2010 10:12If there would be some utility which can scan the source files and put all the function prototypes in one place, it will be a great help in finding such functions. Is this something that google code browser can help with?
Actually I was thinking about such utility just a few days ago It should not be hard to modify the parser a bit to produce a topic with function prototypes. Even better would be if it could work with the current docs and check just for undocumented functions/classes. It could make it easier to maintain the docs up to date. I'll look into it when I have some spare time

Honza

Subject: Re: Global variables in Upp
Posted by jerson on Sun, 12 Sep 2010 11:09:13 GMT
View Forum Message <> Reply to Message

Great, that's good to hear. Let me know if I can help someway.

Subject: Re: Global variables in Upp
Posted by jerson on Mon, 13 Sep 2010 15:22:38 GMT
View Forum Message <> Reply to Message

I am using the s7dataarchiver project by mubeta as an excellent guide on Upp techniques. I found some things that really are not documented anywhere in the existing Upp docs. Here is one such example - the .icpp The documentation does not speak of the difference between iml_header.h, iml.h and iml_source.h either. Life would be so much easier only if these things were available as a document.

http://www.ultimatepp.org/forum/index.php?t=msg&goto=28532&&srch=s7dataarchiver#msg_28532

Subject: Re: Global variables in Upp
Posted by cbpporter on Mon, 13 Sep 2010 17:15:48 GMT
View Forum Message <> Reply to Message

Could you post some documentation of the things you learned and are not available? You don't necessarily have to format them (but it would help).

Subject: Re: Global variables in Upp
Posted by jerson on Tue, 14 Sep 2010 01:17:26 GMT
View Forum Message <> Reply to Message

A couple of things I learned

1. Declaring a global variable/struct in a H file that has header locks does not ensure that the variable will not be defined again. See this thread message 28688.

What I found a possible solution is to keep the declaration to the variable in the H file as extern var and declare the variable in any one of the C files that make my project.

the second alternative to this is to put the definition of the globals into the .icpp file. Now, the linker is happy and does not shout about duplicate definitions. This variable could well be the Image class you want your entire package to share.

in the H file, I put
extern T_Settings Settings;
extern T_Measure Measurements;

in the icpp file I put
T_Settings Settings;
T_Measure Measurements;

From the docs, this is what I got
.icpp
This file type is recognized as .cpp source file, but unlike regular .cpp, which might be placed into the library first and eventually eliminated by linker, .icpp one is always linked as object file. This forces the file to be always linked. The rationale of this is that you can put module initialization code into .icpp that is linked into the executable even when code is not referenced from other files.

2. I learned how to split my file into understandable modules like the s7dataarchiver code. Earlier, I had one BIG cpp file that had all the functions exposed by the class. Now, I have the functions grouped logically into separate CPP files.

3. Many utility functions are just _not documented_ for example hexstring, stringtohex etc.

4. Wav files support is lacking as yet. It is not a major game changer for me at the moment, but I need to code this part to output a siren/klaxon sound on fault conditions.

Overall, I think Upp has everything and more. It just needs a little work on the docs and I'm willing to help in any way.

Regards

Subject: Re: Global variables in Upp
Posted by cbpporter on Tue, 14 Sep 2010 06:18:37 GMT
View Forum Message <> Reply to Message

Well yes, point number 1 is well known about C/C++ and has nothing to do with U++. Include guards protect you from multiple declarations, but you must manually make sure that there is only one definition that the linker sees. Variables MUST be defined with extern if you want to spare yourself future problems. This applies to variables in namespaces too. And static variables in classes.

As always, you can come up with clever layouts for your cpp/h files that will work for your case. But I recommend going with one of the standard ways. These are all convention based and deterministic, i.e. after you decided what file includes what item, the same item will always appear in the same file and in the same logical section and there are no questions related to what goes

into a h and what into a cpp. There is no thinking involved, just applying the pattern you know .

As for point number 3, the names and the parameter lists are pretty self explanatory until they get documented. Be prepared to do a little exploration work.
