Subject: Re: In case someone has trouble including export.h
Posted by James Kuyper on Tue, 29 Jan 2002 15:33:16 GMT
View Forum Message <> Reply to Message

Ruediger Kupper wrote:

> Hi!
>
> We just ran into a problem with IDL5.5's export.h file, when
> including it from a c++ source file:
> I tested g++-2.95 and g++-3.0, and both (most annoyingly)
...

> extern char *IDL_CDECL IDL_VarMakeTempFromTemplate
> IDL_ARG_PROTO((IDL_VPTR template, int type, IDL_StructDefPtr sdef,
> IDL_VPTR *res, int zero));
>
> P.S.:
> The IDL_ARG_PROTO(...) macro is defined near the beginning of the file.
> It acts as a null-filter, reproducing it's argument. It can be used to
> completely turn off prototypes in this file, for compatibility to
> non-ANSI-compilers. We could have utilised this by #define-ing
> IDL_CC_NOT_ANSI, which also seemed to fix the above mentioned problem.
> But I felt this being much more interfering than simply respelling the
> name.

How did that work? If it turns off prototypes, you shouldn't be able to
compile the code using C++. Prototypes are optional in C, but mandatory
in C++.

Subject: Re: In case someone has trouble including export.h
Posted by James Kuyper on Tue, 29 Jan 2002 19:57:45 GMT
View Forum Message <> Reply to Message

Ruediger Kupper wrote:

> Hi!
>
> We just ran into a problem with IDL5.5's export.h file, when
> including it from a c++ source file:
> I tested g++-2.95 and g++-3.0, and both (most annoyingly)
> choked on it, due to the following reason:
>
> The original export.h shipping with IDL 5.5 featured a prototype
> declaration with a variable named "template". This choice of name
> effectively prevents the code from compiling, when included from C++
> code (even when included 'extern "C" {}').

...

> The IDL_ARG_PROTO(...) macro is defined near the beginning of the file.
> It acts as a null-filter, reproducing it's argument. It can be used to
> completely turn off prototypes in this file, for compatibility to
> non-ANSI-compilers. We could have utilised this by #define-ing
> IDL_CC_NOT_ANSI, which also seemed to fix the above mentioned problem.
> But I felt this being much more interfering than simply respelling the
> name.

I've been exchanging e-mail about this with you, and I think we've
reached a point where I should bring my results back to this forum.

The export.h file was apparantly intended to work with C++, as indicated
by the "#ifdef __cplusplus" lines in it. However, since it contains the
C++ keyword 'template', used as a variable name, it can't actually be
compiled in C++.
Your solution was to compile it by #defining IDL_CC_NOT_ANSI. However,
that removes the arguments from every function prototype declared using
the IDL_ARG_PROTO() macro.

In C, that's legal: missing arguments means that the actual number and
types of the arguments are unspecified. It's up to the programmer to
make sure that the arguments are of the correct type and number.

However, in C++, if a function declaration contains no arguments, that
indicates that the function takes no arguments. It's an error to call
the function with arguments. Therefore, the following dumb example
cannot be compiled with C++, whether or not you #define IDL_CC_NOT_ANSI:

```
/* This shouldn't be needed. export.h should #include it: */
#include <stdio.h>

#include "export.h"

void func(void) { }

int main(void)
{
    IDL_LONG length=0;
    IDL_TIMER_CB callback = &func;
    IDL_TIMER_CONTEXT context=0;
    IDL_TIMER_CONTEXT_PTR pcontext=&context;

    IDL_TimerSet(length, callback, 1, pcontext);
    (*callback)();

    return 0;
```

}

Subject: Re: In case someone has trouble including export.h
Posted by Vapuser on Tue, 29 Jan 2002 20:23:03 GMT
View Forum Message <> Reply to Message

Ruediger Kupper <Ruediger.Kupper@Physik.Uni-Marburg.De> writes:

> Hi!
>
> We just ran into a problem with IDL5.5's export.h file, when
> including it from a c++ source file:
> I tested g++-2.95 and g++-3.0, and both (most annoyingly)
> choked on it, due to the following reason:
>
> The original export.h shipping with IDL 5.5 featured a prototype
> declaration with a variable named "template". This choice of name
> effectively prevents the code from compiling, when included from C++
> code (even when included 'extern "C" {}').
>
> Solution:
> With variable names being arbitrary (all that matters is the type and
> order of arguments for defining a proper prototype), you may feel free
> to rename it, so as not to interfere with the C++ compiling process.
>
>
> The original call was:
>
> extern char *IDL_CDECL IDL_VarMakeTempFromTemplate
> IDL_ARG_PROTO((IDL_VPTR template, int type, IDL_StructDefPtr sdef,
> IDL_VPTR *res, int zero));
>
> P.S.:
> The IDL_ARG_PROTO(...) macro is defined near the beginning of the
> file. It acts as a null-filter, reproducing it's argument. It can be
> used to completely turn off prototypes in this file, for compatibility
> to non-ANSI-compilers. We could have utilised this by #define-ing
> IDL_CC_NOT_ANSI, which also seemed to fix the above mentioned
> problem. But I felt this being much more interfering than simply
> respelling the name.
>
>
> Hope that this may be helpful,
> regards,
> Ruediger.
>

I have to admit, this newsgroup is a never-ending source of
information to me. I'm about to upgrade my OS to IRIX 6.5.13f and I
was just checking to see if there were any implications for me
regarding IDL. And while the problem you discuss isn't specifically
a problem for me (yet, I'm hoping to stay with IDL 5.3 for the
nonce) nevertheless it could be were I forced to upgrade to 5.5,
since I depend rather heavily on several linkimage routines.

So, for your time and consideration in making your post, I'd just
like to say thanks. I've now squirreled it away in case I run into
this later!

whd
--
William Daffer: 818-354-0161: William.Daffer@jpl.nasa.gov

Subject: Re: In case someone has trouble including export.h
Posted by Ruediger Kupper on Wed, 30 Jan 2002 13:29:53 GMT
View Forum Message <> Reply to Message

> Ruediger Kupper wrote:
>
>> Hi!
>>
>> We just ran into a problem with IDL5.5's export.h file, when
>> including it from a c++ source file:
>> I tested g++-2.95 and g++-3.0, and both (most annoyingly)
>> choked on it, due to the following reason:
>>
>> The original export.h shipping with IDL 5.5 featured a prototype
>> declaration with a variable named "template". This choice of name
>> effectively prevents the code from compiling, when included from C++
>> code (even when included 'extern "C" {}').
>
> ...
>
>> The IDL_ARG_PROTO(...) macro is defined near the beginning of the
>> file.

...

>> We could have utilised this by #define-ing
>> IDL_CC_NOT_ANSI, which also seemed to fix the above mentioned problem.
>> But I felt this being much more interfering than simply respelling the
>> name.

James Kuyper wrote:
> ...
>
> The export.h file was apparantly intended to work with C++, as indicated
> by the "#ifdef __cplusplus" lines in it. However, since it contains the
> C++ keyword 'template', used as a variable name, it can't actually be
> compiled in C++.
> Your solution was to compile it by #defining IDL_CC_NOT_ANSI. However,
> that removes the arguments from every function prototype declared using
> the IDL_ARG_PROTO() macro.
>
> In C, that's legal: missing arguments means that the actual number and
> types of the arguments are unspecified. It's up to the programmer to
> make sure that the arguments are of the correct type and number.
>
> However, in C++, if a function declaration contains no arguments, that
> indicates that the function takes no arguments. It's an error to call
> the function with arguments.

Thank you for your expertise in this matter.

As I mentioned, there is a very simple way to solve the
problem, i.e., modifying the export.h file by renaming the
variable called "template" to some arbitrary name (not being
a C++ keyword).
This makes export.h compile fine when included into C++ source.

Concerning the (mis-)use of the IDL_ARG_PROTO() macro, which
I mentioned as a passible alternative, you are of course
perfectly right. This will not work with any compiler
compliant to the C++ standard, and I am sorry, if this
suggestion has led to anyone's confusion.

However, the quick test that a ran, using g++-2.95.4, was
successful and did not yield any compiler errors. This
obviously reflects the fact that g++-2.95.4 was not quite up
to the standard in this issue. Trying to compile it on
g++-3.0.2 led to errors, in the way you suggested it should
happen.

So let me once more apologize, and in summary pin down the
following two conclusions:

o if you want to compile C++ code including IDL5.5's
export.h, you will need to alter the variable name "template"

and
o please use up-to-date compilers.

Regards,
Rüdiger Kupper