

Posted by [Randall Skelton](#) on Fri, 17 Nov 2000 08:00:00 GMT

Obviously, my last question was a little too general... the jabs about programming in FORTRAN for too long are well deserved. Nevertheless, if I have a misconception here about IDL please point it out rather than just poking fun...

I cannot seem to get my structure (defined in a local file `structAheader__define.pro`) to be recognized without explicitly executing `.compile/.run` first. i.e.

```
IDL> print, !PATH
./:/home/dougal/aopp/rhskelto/idl:/usr/local/PACK/idl-5.3/lib /hook:
IDL> .run runlog
% Compiled module: GET_ATOCCULT.
IDL> get_atoccult
% Attempt to call undefined procedure/function: 'STRUCTATHEADER__DEFINE'.
% Execution halted at: GET_ATOCCULT      11
    /local/home/mulligan/aopp/rhskelto/src/runlog_idl/runlog.pro
%
    $MAIN$
IDL>
```

BUT, if I tell IDL that structatheader__define.pro exists things work?

```
IDL> .run structAtheader__define
% Compiled module: STRUCTATHEADER__DEFINE.
IDL> .run runlog
% Compiled module: GET_ATOCCULT.
IDL> get_atoccult
>>>> >>>>>>> WORKS FINE >>>>>>>>>>>>>>
```

On Fri, 17 Nov 2000, Doug Reynolds wrote:

```
> Date: Fri, 17 Nov 2000 15:31:20 GMT
> From: Doug Reynolds <dsreyn@ll.mit.edu>
> Newsgroups: comp.lang.idl-pvwave
> Subject: IDL Bug? (Re: include files in IDL programs)
>
> In article <3A153BC0.360BE8A7@acsys.it>,
> Nando <f.iavarone@acsys.it> writes:
>> Randall Skelton wrote:
>>
>>> Hello all,
>>>
```

```

>>> This is a longshot but is there any way to have an 'include file' in IDL.
>>> i.e. I have a data structure which is rather complicated (and big in type)
>>> and I don't want to see it in every program/subroutine that I write. Is
>>> there anyway just to have it included with a simple '#include blah.pro' or
>>> something similar?
>>>
>>> Thanks in advance,
>>>
>>> Randall
>>
>> If you have a file containing code (as batch), you can use the '@filename'
>> to include that codes in your program.
>> If the case of strucured data type, it coul be better to use the __define
>> procedure.
>>
>> For example suppose you have the structure:
>>
>> struct = {structTest, $
>>           pippo: 0L,$
>>           pluto: lonarr(5)}
>>
>> In the first case if you have the file "struct.definition",
>> in your code you can insert that lines using: @struct.definition
>> It works as "#define " of C. IDL simply replace the @struct.definition with
>> the contents of the file.
>>
>>
>> In the second case you can have the file "structTest__define.pro", containing
>> the declaration of your struct:
>>
>>
>>
>> pro structTest__define
>>
>>     struct = {structTest, $
>>               pippo: 0L,$
>>               pluto: lonarr(5)}
>>
>> end;
>>
>>
>>
>>
>> After structTest__define.pro compilation, in your code you can use the
>> statement
>>
>> myStruct = {structTest}.
>>

```

```

>>
>> The difference between the two techniques is that in the first case struct is
>> your variable; in the second one you define a "new data type" structTest that
>> you can use to "declare" all variables you need:
>>
>>
>> myStruct1 = {structTest}
>> myStruct2 = {structTest}
>>
>>
>> myStruct1 and myStruct2 are two different variables of the same type.
>>
>> bye.
>
> Believe it or not, this example doesn't work with the Sun/Solaris version of
> IDL 5.4. I have to compile the procedure with the structure definition before
> attempting to use it:
>
> IDL> test = {structTest}
> % Attempt to call undefined procedure/function: 'STRUCTTEST__DEFINE'.
> % Execution halted at: $MAIN$
> IDL> .r structTest__define
> % Compiled module: STRUCTTEST__DEFINE.
> IDL> test = {structTest}
>
> It turns out that this problem isn't unique to structure definitions - the
> Solaris version of IDL doesn't match routine names containing uppercase
> letters. For example, I defined two routines called searchTest.pro and
> searchtest.pro:
>
> ; searchTest.pro
> pro searchTest
>   print, 'You found me'
> end
>
> ; searchtest.pro
> pro searchtest
>   print, 'You found me'
> end
>
> Then in IDL:
>
> IDL> searchtest
> You found me
> IDL> searchTest
> % Attempt to call undefined procedure/function: 'SEARCHTEST'.
> % Execution halted at: $MAIN$
> IDL> .r searchTest

```

> % Compiled module: SEARCHTEST.
> IDL> searchTest
> You found me
>
> Doug
>

Subject: Re: structure__define
Posted by [Hugh Evans](#) on Mon, 04 Dec 2000 08:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

"Randall Skelton" <rhskelto@atm.ox.ac.uk> wrote in message
news:Pine.LNX.4.21.0011171650140.14194-100000@mulligan.atm.ox.ac.uk...
>
> My OS is Linux and I never would have guessed there would be a problem
> with case sensitivity...
>

In my own humble experiences, I've always found case sensitivity to cause more problems than it solves, even when used properly. But then I grew up on a case insensitive file system, and still fail to see the benefits of having two files with identical names, barring case, i.e. why not just "hello_world.c" instead of "HelloWorld.c", the first is more readable, and only one character longer.
Hugh
