Subject: IDL Bug? (Re: include files in IDL programs)
Posted by dsreyn on Fri, 17 Nov 2000 08:00:00 GMT
View Forum Message <> Reply to Message

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
>> somthing 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
