Subject: functions considered as variables
Posted by Gary Bust on Wed, 04 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

This might be related to my previous post - I don't know. Anyway, I am buzzing away, my driver code calling other routines just fine. I have compliled a .pro file that has a bunch of routines in it, and all of a sudden, I am stopped in my driver routine at:

```
junk = getlatlon(alt_gps)
```

with the error notice: variable getlatlon not found.

So I do a help, and sure enough it says getlatlon is a function. So, before returning to main, i.e. right where the code stopped, I just type the above from the command line - and it works fine. What gives?

Thanks a lot in advance.

-Gary

Subject: Re: functions considered as variables
Posted by Selwyn Yee on Fri, 06 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

I've seen this before. It happens when, in your .pro file, you are trying to call a function which has not yet been compiled. Either the function is in a
different module, or is in the same .pro file, but below your calling statement.

Because IDL is an interpretive language, it executes and/or compiles the code in your .pro file from top to bottom. Your function call:

```
junk = getlatlon(alt_gps)
```

looks just like you are subscripting an array, if IDL hasn't yet compiled your
function getlatlon.

The solution is to use the forward_function command in IDL. The syntax is:

```
forward_function function1, function2, ...
```

So for you, you would precede your function call with:

forward_function getlatlon

Hope this does the trick for you.

--
Selwyn M.T. Yee Phone: (650) 604-2861
Airborne Sensor Facility Fax: (650) 604-4987
NASA Ames Research Center Email: syee@mail.arc.nasa.gov

gbust@arlut.utexas.edu wrote:
>
> This might be related to my previous post - I don't know. Anyway, I am
> buzzing away, my driver code calling other routines just fine. I have
> compliled a .pro file that has a bunch of routines in it, and all of a sudden,
> I am stopped in my driver routine at:
> junk = getlatlon(alt_gps)
> with the error notice: variable getlatlon not found.
>
> So I do a help, and sure enough it says getlatlon is a function. So, before
> returning to main, i.e. right where the code stopped, I just type the above
> from the command line - and it works fine. What gives?
>
> Thanks a lot in advance.
>
> -Gary
>

Subject: Re: functions considered as variables
Posted by davidf on Wed, 11 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Dirk Fabian (dirk@uwast.astro.wisc.edu) writes:

>> The solution is to use the forward_function command in IDL. The syntax
>> is:
>>
>> forward_function function1, function2, ...
>>
>
> Had this problem just the other day.

>
> The other (and coding free) solution is to just compile your driver
> function twice. On the first pass, it will misinterpret the information
> in the driver, but on the second pass, the support functions will be
> interpretted properly as functions (instead of arrays).
>
> I'm not sure if this will work if you have very intricate among your
> support functions.
>
> but IDL> .run foo.pro
> IDL> .run foo.pro
>
> will work for the simple things.

Yes, but for the wrong reasons. Better to write your code
correctly in the first place. (Or, at least understand
what causes error messages and how to fix the underlying
problem and not just the symptom.)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: functions considered as variables
Posted by dirk on Wed, 11 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3500B36D.A8@mail.arc.nasa.gov>,
Selwyn Yee <syee@mail.arc.nasa.gov> wrote:
> I've seen this before. It happens when, in your .pro file, you are
> trying to call a function which has not yet been compiled. Either the
> function is in a
> different module, or is in the same .pro file, but below your calling
> statement.
> Because IDL is an interpretive language, it executes and/or compiles the
> code in your .pro file from top to bottom. Your function call:
[snip]
>
> The solution is to use the forward_function command in IDL. The syntax
> is:

>
>   forward_function function1, function2, ...
>

Had this problem just the other day.

The other (and coding free) solution is to just compile your driver function twice. On the first pass, it will misinterpret the information in the driver, but on the second pass, the support functions will be interpretted properly as functions (instead of arrays).

I'm not sure if this will work if you have very intricate among your support functions.

but IDL> .run foo.pro
     IDL> .run foo.pro

will work for the simple things.

- Dirk

Subject: Re: functions considered as variables
Posted by J.D. Smith on Wed, 11 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

gbust@arlut.utexas.edu wrote:
>
> This might be related to my previous post - I don't know. Anyway, I am
> buzzing away, my driver code calling other routines just fine. I have
> compliled a .pro file that has a bunch of routines in it, and all of a sudden,
> I am stopped in my driver routine at:
>   junk = getlatlon(alt_gps)
> with the error notice: variable getlatlon not found.
>
> So I do a help, and sure enough it says getlatlon is a function. So, before
> returning to main, i.e. right where the code stopped, I just type the above
> from the command line - and it works fine. What gives?
>
> Thanks a lot in advance.
>
> -Gary
>
> -----== Posted via Deja News, The Leader in Internet Discussion ==-----
> http://www.dejanews.com/   Now offering spam-free web-based newsreading

I expect you did some .run or .compile statements in between there.
This is indeed an interesting problem. I replicated it with the single

.pro file (name t2.pro), containing:

```
pro t2
  print,t1(4)
end
function t1,x
  return,x^2
end
```

I get:

```
IDL> t2
% Variable is undefined: T1.
% Execution halted at:  T2                2
 /u/jdsmith/idl/pro/mylib/test/t2.pro
%                      T2                2
 /u/jdsmith/idl/pro/mylib/test/t2.pro
%                      $MAIN$
IDL> help
% At  T2                 2 /u/jdsmith/idl/pro/mylib/test/t2.pro
%     T2                 2 /u/jdsmith/idl/pro/mylib/test/t2.pro
%     $MAIN$
T1              UNDEFINED = <Undefined>
Compiled Procedures:
   $MAIN$  CDEF       CLOAD      MYKEYS2     T2
Compiled Functions:
   FILEPATH

and then,

IDL> retall
IDL> .run t2
% Compiled module: T2.
% Compiled module: T1.
IDL> t2
% Variable is undefined: T1.
% Execution halted at:  T2                2
 /u/jdsmith/idl/pro/mylib/test/t2.pro
%                      $MAIN$
IDL> help
% At  T2                 2 /u/jdsmith/idl/pro/mylib/test/t2.pro
%     $MAIN$
T1              UNDEFINED = <Undefined>
Compiled Procedures:
   $MAIN$  CDEF       CLOAD      MYKEYS2     T2
Compiled Functions:
   FILEPATH           T1
```

As you can see, T1 is both undefined and a compiled function. The manual quotes me:

To determine if it is compiling an array subscript or a function call, IDL checks its internal table of known functions. If it finds a function name that matches the unknown element in the command (fish, in the above example), it calls that function with the argument specified. If IDL does not find a function with the correct name in its table of known functions, it assumes that the unknown element is an array, and attempts to return the value of the designated element of that array.

which seems not to be the case here.

If I repeat the process, with:

```
IDL> .run t2
% Compiled module: T2.
% Compiled module: T1.
IDL> t2
      16
```

it works. I think this probably has to do with an incorrect bit of parsing in the IDL compiler... perhaps the function hash table it uses is not updated at the right time. Quite strange, but perfectly avoidable.

The real solution to this headache is to put supporting functions in a .pro file *before* those functions which refer to them, that way they will always be compiled in time, and you'll never have this problem. You could also put the function in its own .pro file, since IDL will always look on the path first (but, apparently, not always at its own internal table of compiled functions). IDL's belated introduction of [] for array subscripting will eventually eliminate this problem entirely (by requiring it's use instead of ()). Imagine the inefficiency of checking for a function each time a () subscript is made (actually only when the module is compiled, but recompiling is one of my chief activities)... yet another reason to adopt [] as your subscripting mechanism. I hate those wasted cycles.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875

Ithaca, NY 14853 |*|