Subject: Re: newbie seeks to debug "% Syntax error."
Posted by weitkamp on Fri, 04 Jul 2008 22:18:31 GMT
View Forum Message <> Reply to Message

Tom:

> If I understand correctly (and please correct me if I don't!) IDL
> tries to evaluate "regrid(...)"
>
> 0 as a function or procedure on $IDL_PATH
>
> 1 as an array
>
> I'm guessing the etiology here is that IDL fails to find the
> definition, then gets the syntax error at the '=', which is illegal in
> an array.

That's right. It has happened to me that IDL doesn't find the routine called, even if its *.pro file *is* actually somewhere in the !PATH.
In any case, adding

```
COMPILE_OPT strictarr
```

in the routine code (see the online help for COMPILE_OPT) will at least force IDL to interpret parentheses as function calls (and require square brackets for array indexing).

In the worst case, this will at least lead to more reasonable error messages (i.e., "Attempt to call undefined procedure/function" rather than a "Syntax error"). In the best case, it may lead to routines running properly that stopped with an error before.

>) So I'm guessing I need to find out who supplied this code, and find
> out from them where 'regrid' is defined. Am I missing something?

Far from understanding your sophisticated bash scripting, I would guess that you're right on this one too...

Good luck
Timm

Tom Roche wrote:
> I'm very new to IDL (though not to coding), so I wanna make sure I
> understand the following problem correctly before I go up my food
> chain for help:
>
> My advisor just dumped got a big bucket o' IDL (of which he is not the
> author) in my lap. I copy/modified the top-level .pro file

>
> $ pushd ~/jjw/IDL
> $ cp regrid_mozt42_yyf.pro regrid_mozt42_tlr.pro
> $ chmod a-w regrid_mozt42_yyf.pro
>
> and am editing my copy. After reading some introductions to IDL I can
> run it and do some printf-style debugging. The problem I'm having is
>
> IDL> regrid_mozt42_tlr
>> % Compiled module: REGRID_MOZT42_TLR.
>
>> resfield = regrid(var,oldgrid=oldgrid, newgrid=newgrid, /use_grids, $
>> ^
>> % Syntax error.
>> At: /home/tlr/jjw/IDL/ncregrid.pro, Line 180
>
>> resfield = regrid(var[*,*,*,0],oldgrid=oldgrid, newgrid=newgrid, /use_grids, $
>> ^
>> % Syntax error.
>> At: /home/tlr/jjw/IDL/ncregrid.pro, Line 195
>
>> thisfield = regrid(var[*,*,*,j],oldgrid=oldgrid, newgrid=newgrid, /use_grids, $
>> ^
>> % Syntax error.
>> At: /home/tlr/jjw/IDL/ncregrid.pro, Line 200
>
>> % Compiled module: NCREGRID.
>> % Attempt to call undefined procedure/function: 'NCREGRID'.
>> % Execution halted at: REGRID_MOZT42_TLR 21 /home/tlr/jjw/IDL/regrid_mozt42_tlr.pro
>
> If I understand correctly (and please correct me if I don't!) IDL
> tries to evaluate "regrid(...)"
>
> 0 as a function or procedure on $IDL_PATH
>
> 1 as an array
>
> I'm guessing the etiology here is that IDL fails to find the
> definition, then gets the syntax error at the '=', which is illegal in
> an array.
>
> In my .bash_profile I have
>
> $ fgrep -e 'IDL' ~/.bash_profile
>> # for IDL
>> export IDL_DIR="/usr/local/rsi/idl"
>> export IDL_LIB="${IDL_DIR}/lib"
>> if [[-z "${IDL_PATH}"]] ; then

>>   IDL_PATH="~/jjw/IDL:${IDL_LIB}"
>
> ~/jjw/IDL is the bucket o' IDL
>
> else
>>   IDL_PATH="~/jjw/IDL:${IDL_LIB}:${IDL_PATH}"
> fi
>> export IDL_PATH
>
> and the scriptlet
>
> for DIR in ${IDL_PATH//:/ } ; do
>   for CMD in \
>   "find ${DIR} -type f -name '*.pro' | xargs grep -nie 'pro\|
> function' | fgrep -e 'regrid' | fgrep -ve ';'" \
>   ; do
>   echo -e "${CMD}"
>   eval "${CMD}"
>   done
> done
>
> produces
>
>> find ~/jjw/IDL -type f -name '*.pro' | xargs grep -nie 'pro\|function' | fgrep -e 'regrid' | fgrep -ve ';'
>> /home/tlr/jjw/IDL/mozem_regrid.pro:71:FUNCTION RegridSliceAW, data, newgrid=newgrid,
oldgrid=oldgrid, $
>> /home/tlr/jjw/IDL/mozem_regrid.pro:210:   grid->GetProperty, nlon=nlon, nlat=nlat, wlat=wlatn
>> /home/tlr/jjw/IDL/mozem_regrid.pro:211:   oldgrid->GetProperty, wlat=wlato
>> /home/tlr/jjw/IDL/mozem_regrid.pro:340:   oldgrid->GetProperty, wlat=wlato
>> /home/tlr/jjw/IDL/mozem_regrid.pro:341:   grid->GetProperty, wlat=wlatn
>> /home/tlr/jjw/IDL/ncregrid.pro:73:pro ncregrid, filename, oldgridname, newgridname,
oshiftlon=oshiftlon, $
>> /home/tlr/jjw/IDL/regrid_mozt42_tlr.pro:3:PRO regrid_mozt42_tlr
>> find /usr/local/rsi/idl/lib -type f -name '*.pro' | xargs grep -nie 'pro\|function' | fgrep -e 'regrid' |
fgrep -ve ';'
>
> (I also looked at the lines with comments, but there were no
> results of interest. I'm guessing IDL does not do end-of-line
> comments? like this java fragment
>
>   return false;   // TODO: throw exception
>
>) So I'm guessing I need to find out who supplied this code, and find
> out from them where 'regrid' is defined. Am I missing something?
>
> TIA, Tom Roche <Tom_Roche@pobox.com>

Subject: Re: newbie seeks to debug "% Syntax error."
Posted by Tom Roche on Sat, 05 Jul 2008 02:02:54 GMT
View Forum Message <> Reply to Message

Tom Roche Fri, 4 Jul 2008 14:05:42 -0700 (PDT)
>> I'm guessing the etiology here is that IDL fails to find the
>> definition, then gets the syntax error at the '=', which is illegal
>> in an array.

Timm Fri, 4 Jul 2008 15:18:31 -0700 (PDT)
> That's right. It has happened to me that IDL doesn't find the
> routine called, even if its *.pro file *is* actually somewhere in
> the !PATH. In any case, adding

> COMPILE_OPT strictarr

> in the routine code (see the online help for COMPILE_OPT) will at
> least force IDL to interpret parentheses as function calls (and
> require square brackets for array indexing).

Thanks! BTW, I notice

http://127.0.0.1:54669/help/topic/com.rsi.idl.doc.core/COMPI LE_OPT.html
>>> We recommend the use of

>>> COMPILE_OPT IDL2

...

>>> * IDL2 -- A shorthand way of saying:

>>> COMPILE_OPT DEFINT32, STRICTARR

>>> * DEFINT32 -- IDL should assume that lexical integer constants
>>> default to the 32-bit type rather than the usual default of
>>> 16-bit integers. This takes effect from the point where the
>>> COMPILE_OPT statement appears in the routine being compiled and
>>> remains in effect until the end of the routine. The following
>>> table illustrates how the DEFINT32 argument changes the
>>> interpretation of integer constants.