Subject: Re: translating an array name to a string
Posted by John-David T. Smith on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
>> Second, your check to see if a variable is undefined is rather
>> convoluted. It involves two passes to get it right. I prefer instead
>> to use the N_ELEMENTS command to immediately determine whether a
>> variable is undefined. Unlike *assigning* an undefined variable,
>> which does produce an error, simply taking the N_ELEMENTS of an
>> undefined variable will not cause an error.
>
> Ooops, this is actually a mistake on my part, at least on versions of
> IDL earlier than v5.3.
>
> ROUTINE_NAMES(NAME, FETCH=1) will
>
> * succeed if a variable exists and is defined
> * return an undefined value if the variable exists but is undefined
> * utterly fail if the variable doesn't exist, stopping execution

That's not the behavior I see in 5.3. I get the 2nd behavior for both
of the latter two cases, and your first code version works as
advertised. (I like it better too.)

JD

--
J.D. Smith | WORK: (607) 255-6263
Cornell Dept. of Astronomy | (607) 255-5842
304 Space Sciences Bldg. | FAX: (607) 255-5875
Ithaca, NY 14853 |

Subject: Re: translating an array name to a string
Posted by Craig Markwardt on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
> Second, your check to see if a variable is undefined is rather
> convoluted. It involves two passes to get it right. I prefer instead
> to use the N_ELEMENTS command to immediately determine whether a
> variable is undefined. Unlike *assigning* an undefined variable,
> which does produce an error, simply taking the N_ELEMENTS of an
> undefined variable will not cause an error.

Ooops, this is actually a mistake on my part, at least on versions of
IDL earlier than v5.3.

ROUTINE_NAMES(NAME, FETCH=1) will

 * succeed if a variable exists and is defined
 * return an undefined value if the variable exists but is undefined
 * utterly fail if the variable doesn't exist, stopping execution

This doesn't do what you want. However, I think the better approach
than using CATCH, is to use the VARIABLES keyword with ROUTINE_NAMES
to find out if the variable exists first. This is my revised version.

```
; *******
forward_function routine_names

catch, err
if err NE 0 then begin
  catch, /cancel
  message, 'Assign operation failed'
endif

; Protect against an already-defined variable
vnames = routine_names(variables=1)
wh = where(strupcase(var_name) EQ vnames, ct)
if ct GT 0 then begin
    catch,/cancel
    message,'A variable named '+var_name+' already exists.'
endif

; Still here... we need to export ourself to the main level
dummy=routine_names(var_name,myvar,store=1)
catch, /cancel
; *******
```

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: translating an array name to a string
Posted by Craig Markwardt on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

"J.D. Smith" <jdsmith@astro.cornell.edu> writes:
> This is definitely nicer looking, and it reminded me of a caveat. If
> you attempt to fetch a variable which doesn't yet exist, an undefined
> variable will be created on that level for you.

For versions of IDL 5.3 and greater. :-) I should document this.

> I think our methods offer equal protection against certain types of
> failure, but I also think call_function provides additional insurance
> against RSI deciding specifically to remove our capacity to use
> routine_names() (which they might do if we keep talking about it so much
> and people catch on!). It is simple to parse *compiler* statements like
> forward_function for disallowed names. It is impossible (OK, very, very
> awkward), to prohibit the use of classified *strings*. This is probably
> paranoid, but that's why I chose call_function.

Yikes! Paranoid indeed.

This is one argument in favor of an open version of IDL, so language
sabotage like this wouldn't be possible.

Craig

--
--- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
--- --------------

Subject: Re: translating an array name to a string
Posted by John-David T. Smith on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Hi JD--
>
> I agree that the use of ROUTINE_NAMES is has some potential problems
> because it is undocumented. I agree with you also that you need to
> protect your usage of ROUTINE_NAMES with a CATCH handler, since there
> are a lot of ways for things to go wrong.
>
> But I don't agree with your implementation :-).
>
> First, the awkward use of CALL_FUNCTION can be avoided by using the
> FORWARD_FUNCTION declaration. This is always safe, even if the

> function being declared is a built-in one.
>
> Second, your check to see if a variable is undefined is rather
> convoluted. It involves two passes to get it right. I prefer instead
> to use the N_ELEMENTS command to immediately determine whether a
> variable is undefined. Unlike *assigning* an undefined variable,
> which does produce an error, simply taking the N_ELEMENTS of an
> undefined variable will not cause an error.
>
> Finally, users need to be aware that the capability to use
> ROUTINE_NAMES to create new variables at another calling level has
> only come with IDL version 5.3. This is documented at
> http://cow.physics.wisc.edu/~craigm/idl/idl.html under Introspection,
> by the way.
>
> So here is my revised version of your code :-) It's shorter and the
> flow control is primarily linear.
>

```
> ; *******
> forward_function routine_names
>
> catch, err
> if err NE 0 then begin
>   catch, /cancel
>   message, 'Assign operation failed'
> endif
>
> ; Protect against an already-defined variable
> if n_elements(routine_names(var_name,fetch=1)) GT 0 then begin
>     catch,/cancel
>     message,'A variable named '+var_name+' already exists.'
> endif
>
> ; Still here... we need to export ourself to the main level
> dummy=routine_names(var_name,myvar,store=1)
> catch, /cancel
> ; *******
```

This is definitely nicer looking, and it reminded me of a caveat. If
you attempt to fetch a variable which doesn't yet exist, an undefined
variable will be created on that level for you.

I think our methods offer equal protection against certain types of
failure, but I also think call_function provides additional insurance
against RSI deciding specifically to remove our capacity to use
routine_names() (which they might do if we keep talking about it so much
and people catch on!). It is simple to parse *compiler* statements like

forward_function for disallowed names. It is impossible (OK, very, very
awkward), to prohibit the use of classified *strings*. This is probably
paranoid, but that's why I chose call_function. In any case I will
modify my method to include the n_elements() test (which I was stupid
not to think of).

JD

--
```
 J.D. Smith                  |   WORK: (607) 255-6263
 Cornell Dept. of Astronomy  |         (607) 255-5842
 304 Space Sciences Bldg.    |    FAX: (607) 255-5875
 Ithaca, NY 14853            |
```

Subject: Re: translating an array name to a string
Posted by Craig Markwardt on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hi JD--

I agree that the use of ROUTINE_NAMES is has some potential problems
because it is undocumented. I agree with you also that you need to
protect your usage of ROUTINE_NAMES with a CATCH handler, since there
are a lot of ways for things to go wrong.

But I don't agree with your implementation :-).

First, the awkward use of CALL_FUNCTION can be avoided by using the
FORWARD_FUNCTION declaration. This is always safe, even if the
function being declared is a built-in one.

Second, your check to see if a variable is undefined is rather
convoluted. It involves two passes to get it right. I prefer instead
to use the N_ELEMENTS command to immediately determine whether a
variable is undefined. Unlike *assigning* an undefined variable,
which does produce an error, simply taking the N_ELEMENTS of an
undefined variable will not cause an error.

Finally, users need to be aware that the capability to use
ROUTINE_NAMES to create new variables at another calling level has
only come with IDL version 5.3. This is documented at
http://cow.physics.wisc.edu/~craigm/idl/idl.html under Introspection,
by the way.

So here is my revised version of your code :-) It's shorter and the
flow control is primarily linear.

```
; *******
forward_function routine_names

catch, err
if err NE 0 then begin
  catch, /cancel
  message, 'Assign operation failed'
endif

; Protect against an already-defined variable
if n_elements(routine_names(var_name,fetch=1)) GT 0 then begin
    catch,/cancel
    message,'A variable named '+var_name+' already exists.'
endif

; Still here... we need to export ourself to the main level
dummy=routine_names(var_name,myvar,store=1)
catch, /cancel
; *******
```

Craig

"J.D. Smith" <jdsmith@astro.cornell.edu> writes:

```
>  By the way, for those of you using routine_names for heavy magic... you
>  might consider examining the following extra-cautions snippet to export
>  a variable to the $MAIN$ level:
>
>     var_free=0
>     catch, err
>     if err ne 0 then begin
>        ;; An undefvar indicates routine_info ran and
>        ;; the variable is free
>        if !ERROR_STATE.NAME ne 'IDL_M_UNDEFVAR' then begin
>          catch,/cancel
>   message,"Can't complete operation... Try obj=sp_sel()"
>        endif
>        var_free=1
>     endif
>
>     ;; If we need to check if the variable name is available, do so.
>     if var_free eq 0 then $
>       rn=call_function('routine_names',var_name,FETCH=1)
>
>     if n_elements(rn) ne 0 then begin
>   catch,/cancel
>   message,'A variable named '+var_name+' already exists.'
>     endif
>
>     ;; Still here... we need to export ourself to the main level
>     rn=call_function('routine_names',var_name,myvar,store=1)
>
```

> basically the idea is to wrap routine_names as a string in
> call_function, to allow your routine to compile even if RSI yanks or
> renames it (it wouldn't compile if you tried to call it directly).
> You'll get an error, of course, which will be caught. You have to
> discriminate between errors caused by the successful operation of
> routine_names(), and those caused by incorrect arguments, changed
> keywords (IDL_M_KEYWORD_BAD), or other mutations routine_names() has
> undergone (such as vanishing altogether -- IDL_M_UPRO_UNDEF). You
> obviously have to have a backup plan too, to tell your users what to do
> in case routine_names() has broken. But it's better than your program
> not running at all though.
>
> JD
>
> --
> J.D. Smith | WORK: (607) 255-6263
> Cornell Dept. of Astronomy | (607) 255-5842
> 304 Space Sciences Bldg. | FAX: (607) 255-5875
> Ithaca, NY 14853 |

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------

Subject: Re: translating an array name to a string
Posted by promashkin on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Just a speculation along the lines.
I faced this matter about a year ago, and was given an advice from a RSI developer, not to use variable names as titles and not to define meaningful variable names at runtime. Instead, organize the data internally so that a meaningful name san be stored in a structure field, for example (or an object property).
The reason I had that question is that I came to IDL from Igor Pro, whose proprietary macro language uses "waves" with meaningful (hopefully) names that you create yourself. It took me a few days to realize the difference, but once I did, I never needed to use variable names directly again. It is the persistence of arrays in Igor, absence

of a structure data type and Igor's inability to create temporary arrays
on the fly that makes you use meaningful names. Then, you must keep
track of what's being used all by yourself, with the help of
"nameofwave" functions. Pretty cumbersome.
Cheers,
Pavel

Subject: Re: translating an array name to a string
Posted by Craig Markwardt on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Williams <d.williams@qub.ac.uk> writes:

```
>  Alex Schuster wrote:
>>
>>  David Williams wrote:
>>
>>>  Hi. This is probably a very basic IDL question, so apologies if that's
>>>  the case. I'm looking for a way to translate the name of an array (e.g.
>>>  "DATACUBE1") into a string that I can use in titles and/or in feedback
>>>  at the prompt. I want to make my routines more user-friendly, and I hate
>>>  forcing a title
>>
>>  Have a look at the OUTPUT keyword to HELP:
>>
>
>  Vielen Dank, Alex! That was just what I needed.
```

Pardon me, but if you know the name of your variable already, then it
seems that you don't really have a problem. If you know the variable
name is MOVIE, then why not hardcode that into the output string?

On the other hand, if you are in a procedure or function, it might
indeed be useful to know the name of variable at the *caller's* level.
In that case you would want to use the ROUTINE_NAMES function, which
is unfortunately undocumented.

I have documented what I and the newsgroup have discovered about this
very useful function. You will find it here:

http://cow.physics.wisc.edu/~craigm/idl/idl.html

listed under Introspection.

Good luck!
Craig

--
 --- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 --- --------------

Subject: Re: translating an array name to a string
Posted by John-David T. Smith on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster wrote:
>
> David Williams wrote:
>
>> Hi. This is probably a very basic IDL question, so apologies if that's
>> the case. I'm looking for a way to translate the name of an array (e.g.
>> "DATACUBE1") into a string that I can use in titles and/or in feedback
>> at the prompt. I want to make my routines more user-friendly, and I hate
>> forcing a title
>
> Have a look at the OUTPUT keyword to HELP:
>
```
> IDL> help, movie
> MOVIE           BYTE      = Array[256, 256, 64]
> IDL> help, movie, output=output
> IDL> print, output
> MOVIE           BYTE      = Array[256, 256, 64]
> CLU> print, (str_sep( output[0], " "))[0]
> MOVIE
```

Also try:

```
print,routine_names(movie,/ARG_NAME)
```

Don't try looking this one up in the manual. Neither of these
operations are supported. RSI reserves the right to remove or retool
routine_names() AND/OR the output format of help. That's the price you
pay.

By the way, for those of you using routine_names for heavy magic... you
might consider examining the following extra-cautions snippet to export
a variable to the $MAIN$ level:

```
  var_free=0
  catch, err
  if err ne 0 then begin
```

```
      ;; An undefvar indicates routine_info ran and
      ;; the variable is free
      if !ERROR_STATE.NAME ne 'IDL_M_UNDEFVAR' then begin
        catch,/cancel
 message,"Can't complete operation... Try obj=sp_sel()"
      endif
      var_free=1
   endif

   ;; If we need to check if the variable name is available, do so.
   if var_free eq 0 then $
     rn=call_function('routine_names',var_name,FETCH=1)

   if n_elements(rn) ne 0 then begin
 catch,/cancel
 message,'A variable named '+var_name+' already exists.'
   endif

   ;; Still here... we need to export ourself to the main level
   rn=call_function('routine_names',var_name,myvar,store=1)
```

basically the idea is to wrap routine_names as a string in
call_function, to allow your routine to compile even if RSI yanks or
renames it (it wouldn't compile if you tried to call it directly).
You'll get an error, of course, which will be caught. You have to
discriminate between errors caused by the successful operation of
routine_names(), and those caused by incorrect arguments, changed
keywords (IDL_M_KEYWORD_BAD), or other mutations routine_names() has
undergone (such as vanishing altogether -- IDL_M_UPRO_UNDEF). You
obviously have to have a backup plan too, to tell your users what to do
in case routine_names() has broken. But it's better than your program
not running at all though.

JD

--
 J.D. Smith | WORK: (607) 255-6263
 Cornell Dept. of Astronomy | (607) 255-5842
 304 Space Sciences Bldg. | FAX: (607) 255-5875
 Ithaca, NY 14853 |

Subject: Re: translating an array name to a string
Posted by David Williams on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Alex Schuster wrote:
>

> David Williams wrote:
>
>> Hi. This is probably a very basic IDL question, so apologies if that's
>> the case. I'm looking for a way to translate the name of an array (e.g.
>> "DATACUBE1") into a string that I can use in titles and/or in feedback
>> at the prompt. I want to make my routines more user-friendly, and I hate
>> forcing a title
>
> Have a look at the OUTPUT keyword to HELP:
>

Vielen Dank, Alex! That was just what I needed.

== =
David R. Williams, Tel.: (+44 1232) 273509
APS Division,
Pure & Applied Physics Dept.,
Queen's University,
Belfast,
BT7 1NN. http://star.pst.qub.ac.uk/~drw/
== =

Subject: Re: translating an array name to a string
Posted by Alex Schuster on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Williams wrote:

> Hi. This is probably a very basic IDL question, so apologies if that's
> the case. I'm looking for a way to translate the name of an array (e.g.
> "DATACUBE1") into a string that I can use in titles and/or in feedback
> at the prompt. I want to make my routines more user-friendly, and I hate
> forcing a title

Have a look at the OUTPUT keyword to HELP:

```
IDL> help, movie
MOVIE           BYTE      = Array[256, 256, 64]
IDL> help, movie, output=output
IDL> print, output
MOVIE           BYTE      = Array[256, 256, 64]
CLU> print, (str_sep( output[0], " "))[0]
MOVIE
```

Alex
--
Alex Schuster   Wonko@weird.cologne.de   PGP Key available

alex@pet.mpin-koeln.mpg.de

Subject: Re: translating an array name to a string
Posted by Martin Schultz on Thu, 19 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Williams wrote:
>
> Hi. This is probably a very basic IDL question, so apologies if that's
> the case. I'm looking for a way to translate the name of an array (e.g.
> "DATACUBE1") into a string that I can use in titles and/or in feedback
> at the prompt. I want to make my routines more user-friendly, and I hate
> forcing a title
>
> Say I want to display a frame from a movie of images, called MOVIE. I
> use tvim, and I currently say something like:
>
>   tvim,MOVIE,$
>   title='frame No.'+ARR2STR(i,/trim)+' of your datacube'.
>
> (ARR2STR is a SolarSoftWare IDL routine which turns variable values into
> strings, and the TRIM keyword removes unnecessary spaces - very handy!)
>
> What I'd like to be able to do is say:
>
>   ...
>   title='frame No.'+ARR2STR(i,/trim)+' of '+ARRNAME2STRING(movie)
>
> so that I can instantly show which array I'm looking at, as well as
> which frame.
>
> Nothing obvious springs to my mind, so I thought I'd ask this newsgroup.
>
> Thanks in advance for any help,
> Dave Williams.<d.williams@qub.ac.uk>
>
> --

Dave, the question is not so basic as it seems --- at least if you
start thinking about what your user may wish to see as title. I guess
there is a reason why the netcdf people have invented the long_name
attribute... But to solve your actual problem; this can probably be
done with some mystic use of Routine_Info(); Reimar Bauer from Juelich
(r.bauer@fz-juelich.de) is an expert on this.

Cheers,
Martin

--
[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[

Subject: Re: translating an array name to a string
Posted by David Williams on Fri, 20 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

--
Thanks to everyone for their suggestions. Looks like it'll keep me going
for some time.

Dave

== =
David R. Williams, Tel.: (+44 1232) 273509
APS Division,
Pure & Applied Physics Dept.,
Queen's University,
Belfast,
BT7 1NN. http://star.pst.qub.ac.uk/~drw/
== =

Subject: Re: translating an array name to a string
Posted by David Williams on Fri, 20 Oct 2000 07:00:00 GMT
View Forum Message <> Reply to Message

> On the other hand, if you are in a procedure or function, it might
> indeed be useful to know the name of variable at the *caller's* level.
> In that case you would want to use the ROUTINE_NAMES function, which
> is unfortunately undocumented.
>
> I have documented what I and the newsgroup have discovered about this
> very useful function. You will find it here:
>
> http://cow.physics.wisc.edu/~craigm/idl/idl.html
>
> listed under Introspection.
>

> Good luck!
> Craig

Thanks a lot, Craig, for this and the rest of your contibrutions on the matter.

Dave.

== =
David R. Williams, Tel.: (+44 1232) 273509
APS Division,
Pure & Applied Physics Dept.,
Queen's University,
Belfast,
BT7 1NN. http://star.pst.qub.ac.uk/~drw/
== =