Subject: Re: Keyword checking
Posted by David Fanning on Wed, 29 May 2002 13:43:45 GMT
View Forum Message <> Reply to Message

Randall Skelton (rhskelto@atm.ox.ac.uk) writes:

> Surely there must be something simpiler than my 2-step approach ;)

Nope. That's it. :-)

   http://www.dfanning.com/tips/keyword_check.html

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Keyword checking
Posted by Jaco van Gorkom on Wed, 29 May 2002 13:52:10 GMT
View Forum Message <> Reply to Message

"Randall Skelton" <rhskelto@atm.ox.ac.uk> wrote in message
 news:Pine.LNX.4.33.0205291352100.19744-100000@mulligan.atm.o x.ac.uk...

```
>
> pro test, B=b
>   if n_elements(b) eq 0 and arg_present(b) eq 1 then $
>     message, 'You passed an undefined variable as a keyword'
>   if n_elements(b) gt 0 then b = 'passed'
> end
>
```
> Surely there must be something simpiler than my 2-step approach ;)

If the difference between "passing nothing" and "not passing anything"
is really important to you, then this is it :-)
Of course, you could always write this into your own function
KEYWORD_UNDEFINED()...

cheers,
  Jaco

Subject: Re: Keyword checking
Posted by Jaco van Gorkom on Wed, 29 May 2002 14:04:39 GMT
View Forum Message <> Reply to Message

"Jaco van Gorkom" <j.c.van.gorkom@fz-juelich.de> wrote in message
news:ad2mat$3b09$1@zam602.zam.kfa-juelich.de...

```
>  "Randall Skelton" <rhskelto@atm.ox.ac.uk> wrote in message
>   news:Pine.LNX.4.33.0205291352100.19744-100000@mulligan.atm.o x.ac.uk...
>>
>>  pro test, B=b
>>    if n_elements(b) eq 0 and arg_present(b) eq 1 then $
>>      message, 'You passed an undefined variable as a keyword'
>>    if n_elements(b) gt 0 then b = 'passed'
>>  end
>>
>>  Surely there must be something simpiler than my 2-step approach ;)
>
>  If the difference between "passing nothing" and "not passing anything"
>  is really important to you, then this is it :-)
>  Of course, you could always write this into your own function
>  KEYWORD_UNDEFINED()...
```

Well, try to, maybe. As I sent this, it occurred to me that arg_present()
works only in the present function, sort of. Still, there may be
undocumented ways?

  jaco

Subject: Re: Keyword checking
Posted by Paul Van Delst[1] on Wed, 29 May 2002 14:15:47 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:

```
>
>  I have run into a great deal of trouble checking keywords in IDL and I
>  thought I would relay my thoughts and frustrations.  I am trying to
>  prevent a user from passing an undefined variable in a keyword...
>
>  Imagine,
>
>  pro test, A=a, B=b
>    print, n_params()
>    print, arg_present(a), keyword_set(a), n_elements(a), size(a,/type)
>    print, arg_present(b), keyword_set(b), n_elements(b), size(b,/type)
>  end
>
>  If I call this routine as:
```

```
>
>  IDL> .reset
>  IDL> test, a=1, b=undefvar
>  % Compiled module: TEST.
>             0
>        0      1          1          2
>        1      0          0          0
>
>  So, in order to prevent a user from passing junk in a keyword, I have:
>
>  pro test, B=b
>    if n_elements(b) eq 0 and arg_present(b) eq 1 then $
>      message, 'You passed an undefined variable as a keyword'
>    if n_elements(b) gt 0 then b = 'passed'
>  end
>
>  Now,
>
>  IDL> .reset
>  IDL> test, b=junk
>  % Compiled module: TEST.
>  % TEST: You passed an undefined variable as a keyword
>
>  Surely there must be something simpiler than my 2-step approach ;)
```

Why the arg_present? Do you specifically want to tell the user that what they passed was undefined? I thought arg_present was for using when you wanted to return something in the variable? If the data is for input only, won't n_elements() suffice?

me confused.

paulv

```
--
Paul van Delst             Religious and cultural
CIMSS @ NOAA/NCEP/EMC      purity is a fundamentalist
Ph: (301)763-8000 x7274    fantasy
Fax:(301)763-8545                    V.S.Naipaul
```

Subject: Re: Keyword checking
Posted by thompson on Wed, 29 May 2002 14:55:20 GMT

View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:

> Randall Skelton wrote:
>>

>> I have run into a great deal of trouble checking keywords in IDL and I
>> thought I would relay my thoughts and frustrations. I am trying to
>> prevent a user from passing an undefined variable in a keyword...
>>

(stuff deleted)

> Why the arg_present? Do you specifically want to tell the user that what they
> passed was undefined? I thought arg_present was for using when you wanted to
> return something in the variable? If the data is for input only, won't
> n_elements() suffice?

> me confused.

I agree. I can't think of any legitimate reason why one would want to generate
an error message if an undefined keyword was passed.

In the case of an input keyword, and undefined keyword should be treated
exactly as if the keyword was not passed at all. The reason for this is quite
simple. If one has embedded subroutines with keyword inheritance, there has to
be some way to pass the keywords along. For example

```
 pro test1, key1=key1
 test2, key1=key1
 return
 end

 pro test2, key1=key1
 if n_elements(key1) ne 0 then help, key1 else print, 'KEY1 not passed'
 return
 end
```

This obviously very trivial example is enough to illustrate my point. If the
user calls the procedure TEST1, the keyword KEY1 will be passed along to TEST2
whether or not the user called TEST1 with that keyword. If the user simply
calls TEST1 without the keyword, it will still appear in TEST2 as an undefined
value. There's nothing wrong with that.

In the output case, all one cares about is whether or not there's a variable to
receive the output. It doesn't matter if the variable was defined previously
or not.

I can only conclude that trapping an undefined keyword as an error is bad IDL
programming practice.

William Thompson

Subject: Re: Keyword checking
Posted by Liam E. Gumley on Wed, 29 May 2002 15:22:14 GMT
View Forum Message <> Reply to Message

William Thompson wrote:
[stuff deleted]
> I agree. I can't think of any legitimate reason why one would want to generate
> an error message if an undefined keyword was passed.
[stuff deleted]

As David points out, there are ways to do this sort of checking. However
I must agree with William. In my mind, the whole point of keywords is
that they are *optional*. If a valid defined variable is not passed for
a keyword, then the called routine must take some default action. If the
keyword is important enough that the developer thinks argument checking
wizardry is necessary, then perhaps it should be a positional parameter
instead.

Cheers,
Liam.
Practical IDL Programming
http://www.gumley.com/

Subject: Re: Keyword checking
Posted by R.Bauer on Wed, 29 May 2002 16:02:58 GMT
View Forum Message <> Reply to Message

Randall Skelton wrote:

Dear Randall,

keyword_set
 should only used to test variables to be true or false

arg_present
 is used to find out if a parameter is called by reference
   this gives true and by value gives false

n_elements
 returns the number of elements of a parameter

We have set up a test case in our lessons
I have posted some time ago.

I believe all is clear if you once filled out these forms

regards

Reimar

1)
There are three functions given

```
   FUNCTION test1,minimum=min_val
      IF KEYWORD_SET(min_val) THEN RETURN,1 ELSE RETURN,0
   END


   FUNCTION test2,minimum=min_val
      IF ARG_PRESENT(min_val) THEN RETURN,1 ELSE RETURN,0
   END


   FUNCTION test3,minimum=min_val
      IF N_ELEMENTS(min_val) GT 0 THEN RETURN,1 ELSE RETURN,0
   END
```

Fill out the form:

CALL	test1	test2	test3
PRINT, testX( )			
PRINT, testX(minimum=0)			
PRINT, testX(minimum=10)			
PRINT, testX(minimum=-10)			
mv=0 & PRINT, testX(minimum=mv)			
mv=10 & PRINT, testX(minimum=mv)			

PRINT, testX(minimum=mv2)			

2)
There are three functions given

```
   FUNCTION test1,min_val
      IF KEYWORD_SET(min_val) THEN RETURN,1 ELSE RETURN,0
   END


   FUNCTION test2,min_val
      IF ARG_PRESENT(min_val) THEN RETURN,1 ELSE RETURN,0
   END


   FUNCTION test3,min_val
      IF N_ELEMENTS(min_val) GT 0 THEN RETURN,1 ELSE RETURN,0
   END
```

Fill out the form:

CALL	test1	test2	test3
PRINT, testX( )			
PRINT, testX(0)			
PRINT, testX(10)			
PRINT, testX(-10)			
mv=0 & PRINT, testX(mv)			
mv=10 & PRINT, testX(mv)			
PRINT, testX(mv2)			

```
>
> I have run into a great deal of trouble checking keywords in IDL and I
> thought I would relay my thoughts and frustrations.  I am trying to
> prevent a user from passing an undefined variable in a keyword...
>
> Imagine,
>
> pro test, A=a, B=b
>   print, n_params()
>   print, arg_present(a), keyword_set(a), n_elements(a), size(a,/type)
>   print, arg_present(b), keyword_set(b), n_elements(b), size(b,/type)
> end
>
> If I call this routine as:
>
> IDL> .reset
> IDL> test, a=1, b=undefvar
> % Compiled module: TEST.
>            0
>        0      1          1          2
>        1      0          0          0
>
> So, in order to prevent a user from passing junk in a keyword, I have:
>
> pro test, B=b
>   if n_elements(b) eq 0 and arg_present(b) eq 1 then $
>     message, 'You passed an undefined variable as a keyword'
>   if n_elements(b) gt 0 then b = 'passed'
> end
>
> Now,
>
> IDL> .reset
> IDL> test, b=junk
> % Compiled module: TEST.
> % TEST: You passed an undefined variable as a keyword
>
> Surely there must be something simpiler than my 2-step approach ;)
>
```

> Cheers,
> Randall

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
-- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml
== =======