Subject: keyword_set bug or feature
Posted by phil on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Thanks to those who responded to my ? concerning common blocks.

I just can across another 'feature' and was wondering if others felt
that it is really a 'bug' or if there is some way to over come it. It
has to do with keywords in functions or procedures. I use several and
check to see if the are set using

```
  IF NOT(keyowrd_set(key)) THEN BEGIN
    ;set a default value here
  ENDIF
```

The problem comes in if the user wants to set key = 0. If so then it
appears to the above test that the keyword is not set even though in
the function call the user typed

```
  Result = somefunction(var,key=0)
```

What gives? Can I just add key = key + 1 and key = key - 1 around the
testing if statement?

Any other suggestions appreciated.
--
/** ****************/
Phil Williams
Postdoctoral Researcher "One man gathers what
MRI Facility another man spills..."
The Ohio State University -The Grateful Dead
email: phil@peace.med.ohio-state.edu
URL: http://justice.med.ohio-state.edu:1525
/** ****************/

Subject: Re: keyword_set bug or feature
Posted by zawodny on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article <3p03k3$ip1@kwuz.nerc-keyworth.ac.uk> wmc@unixa.nerc-keyworth.ac.uk writes:
>
> Another solution is to set key=[0], ie the array containing 0.
> Then keyword_set returns true, and IDL happily uses [0] and 0 fairly
> interchangably.
>

Yes, but in practice this might appear in code as follows:

```
pro MYPRO, v1, v2, key1=key1, key2=key2
...
if KEYWORD_SET([key1]) then ...
...
return
end
```

The problem here being that when IDL tries to put key1 into a single element vector and key1 has not been passed in the call, IDL will barf with "variable key1 undefined".

I think you were suggesting that the user type instead

```
IDL> mypro,v1,v2,key1=[0]
```

If so I think you are asking too much of the user and would be better off just changing the code to use N_ELEMENTS() instead of KEYWORD_SET().

--
Joseph M. Zawodny (KO4LW) NASA Langley Research Center
Internet: j.m.zawodny@larc.nasa.gov MS-475, Hampton VA, 23681-0001
TCP/IP: ko4lw@ko4lw.ampr.org Packet: ko4lw@n4hog.va.usa.na

Subject: Re: keyword_set bug or feature
Posted by zawodny on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article <3p0030$d0s@post.gsfc.nasa.gov> thompson@orpheus.nascom.nasa.gov (William Thompson) writes:
>
> This is a very common error that people make. They want to use KEYWORD_SET()
> to test for the *existence* of keyword variables. KEYWORD_SET() is *only* for
> keywords which can take True or False values, and has the (quite correct in my
> opinion) property that not passing something is the same as passing it as
> False.
>
> To test for the *existence* of keywords that can take arbitrary values, use the
> function N_ELEMENTS() instead.

While Bill Thompson is certainly knowledgeble and we are for the most part better off because he takes the time to answer peoples questions, his answers sometimes have a tone of arrogance (or at least a bit of "If you do not do it my way, then you are wrong").

In this instance he is again basically correct. Keywords come in two flavors. Those that take the logical values True and False and those

that do not. The former are used primarily for program control (do something special when a particular keyword is set). The latter is used to pass optional "valued" parameters in a function or proceedure.

In an environment where code is shared or codeveloped, it is important to define and obey certain rules or standard practices. When using keywords in such an environment it is best (not the *only* proper way) to set logical keywords using /. For example

```
 mypro,v1,v2,/logical_key
```

and to test for them being set by using the KEYWORD_SET() function. Non-logical keywords are best given a value (notice I did not say "best set") using the = operator and tested for by using the N_ELEMENTS() function.

It is not, however, an *error* to do otherwise since

```
 mypro,v1,v2,logical_key=1
```

does indeed produce the same behavior as above. Errors produce erroneous results or cause improper functionality. Similarly,

```
 mypro,v1,v2,/non-logical_key
```

is A way to set the value of non-logical_key equal to 1.

These are not wrong ways or *errors*, but merely bad habits. If you code by habit or your software users have bad habits then you must be careful. The only instance where a problem appears is when you use the logical test KEYWORD_SET() on a non-logical keyword which might have a valid value of zero. Otherwise, try to get into a better habit, but in a pinch use whatever works.

It is interesting to note that
 KEYWORD_SET(0)   = 0 ,
and KEYWORD_SET(2)   = 1 as they obviously should be,
but KEYWORD_SET([0]) = 1 !

I'm sure this tells many of us just how KEYWORD_SET() is coded.

I think that what a number of folks here are asking for is a new function, call it KEYWORD_EXISTS(), which returns true if the keyword was used in the call to a proceedure or function. Such a function would be trivial to write,

```
function KEYWORD_EXISTS(key)
return,(n_elements(key) ne 0)
```

end

and might actually get some use if provided within IDL as a complement
to the KEYWORD_SET() function. Seeing

 if KEYWORD_EXISTS(key) then ...

in some foreign code is a bit more readable than

 if(N_ELEMENTS(key) ne 0) then ...

,but not by much.

I hope that this explains the KEYWORD issues a bit for the newcommers.

--
Joseph M. Zawodny (KO4LW) NASA Langley Research Center
Internet: j.m.zawodny@larc.nasa.gov MS-475, Hampton VA, 23681-0001
TCP/IP: ko4lw@ko4lw.ampr.org Packet: ko4lw@n4hog.va.usa.na

Subject: Re: keyword_set bug or feature
Posted by wmc on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article 5pc@info.epfl.ch, llobet@elpp1.epfl.ch (Xavier Llobet i Sales EPFL-CRPP 1015 Lausanne CH) writes:
>
> In article <PHIL.95May12090139@peace.med.ohio-state.edu>,
> phil@peace.med.ohio-state.edu (Phil) writes:

> =It has to do with keywords in functions or procedures. I use several and
> =check to see if the are set using
> = IF NOT(keyowrd_set(key)) THEN BEGIN
> =   ;set a default value here
> = ENDIF

> =The problem comes in if the user wants to set key = 0. If so then it
> =appears to the above test that the keyword is not set even though in
> =the function call the user typed

> = Result = somefunction(var,key=0)

> You could try
>   IF N_ELEMENTS(key) EQ 0 THEN BEGIN

Another solution is to set key=[0], ie the array containing 0.
Then keyword_set returns true, and IDL happily uses [0] and 0 fairly

interchangably.

- William Connolley

Subject: Re: keyword_set bug or feature
Posted by thompson on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

phil@peace.med.ohio-state.edu (Phil) writes:

> Thanks to those who responded to my ? concerning common blocks.

> I just can across another 'feature' and was wondering if others felt
> that it is really a 'bug' or if there is some way to over come it. It
> has to do with keywords in functions or procedures. I use several and
> check to see if the are set using

> IF NOT(keyowrd_set(key)) THEN BEGIN
> ;set a default value here
> ENDIF

> The problem comes in if the user wants to set key = 0. If so then it
> appears to the above test that the keyword is not set even though in
> the function call the user typed
>
> Result = somefunction(var,key=0)

> What gives? Can I just add key = key + 1 and key = key - 1 around the
> testing if statement?

This is a very common error that people make. They want to use KEYWORD_SET() to test for the *existence* of keyword variables. KEYWORD_SET() is *only* for keywords which can take True or False values, and has the (quite correct in my opinion) property that not passing something is the same as passing it as False.

To test for the *existence* of keywords that can take arbitrary values, use the function N_ELEMENTS() instead.

Bill Thompson

Subject: Re: keyword_set bug or feature
Posted by llobet on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

In article <PHIL.95May12090139@peace.med.ohio-state.edu>,
phil@peace.med.ohio-state.edu (Phil) writes:

[...]
=I just can across another 'feature' and was wondering if others felt
=that it is really a 'bug' or if there is some way to over come it. It
=has to do with keywords in functions or procedures. I use several and
=check to see if the are set using
=
=   IF NOT(keyowrd_set(key)) THEN BEGIN
=   ;set a default value here
=   ENDIF
=
=The problem comes in if the user wants to set key = 0. If so then it
=appears to the above test that the keyword is not set even though in
=the function call the user typed
=
=   Result = somefunction(var,key=0)
=
=What gives? Can I just add key = key + 1 and key = key - 1 around the
=testing if statement?
=
=Any other suggestions appreciated.

You could try

     IF N_ELEMENTS(key) EQ 0 THEN BEGIN

-xavier

Subject: Re: keyword_set bug or feature
Posted by sterner on Fri, 12 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

phil@peace.med.ohio-state.edu (Phil) writes:

> I just can across another 'feature' and was wondering if others felt
> that it is really a 'bug' or if there is some way to over come it. It
> has to do with keywords in functions or procedures. I use several and
> check to see if the are set using

>   IF NOT(keyowrd_set(key)) THEN BEGIN
>   ;set a default value here
>   ENDIF

> The problem comes in if the user wants to set key = 0. If so then it

> appears to the above test that the keyword is not set even though in
> the function call the user typed
>
>   Result = somefunction(var,key=0)

> What gives? Can I just add key = key + 1 and key = key - 1 around the
> testing if statement?

I always use a statement like the following:

  if n_elements(key) eq 0 then key=default_value

This is not something you would guess as the obvious solution
but n_elements returns 0 if its argument is undefined. You only
need a default value when your keyword is undefined so this works.
It also works the same for ordinary positional arguments.

What is missing is a way to determine if an output keyword was
requested in the call. Often the variable in such a keyword is
undefined since it is an output of the routine. I don't know
anyway to find out if the caller actually requested the output keyword
or not so any such keywords must always be set. Occasionally it
may take a lot of processing to find the desired output. It would be
nice if you could determine if it was really requested and only do it
then.

Ray Sterner   sterner@tesla.jhuapl.edu
The Johns Hopkins University   North latitude 39.16 degrees.
Applied Physics Laboratory   West longitude 76.90 degrees.
Laurel, MD 20723-6099
WWW Home page: ftp://fermi.jhuapl.edu/www/s1r/people/res/res.html

---

Subject: Re: keyword_set bug or feature
Posted by thompson on Tue, 16 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

soc@festival.ed.ac.uk (Stephen O'Connell) writes:

 (stuff deleted)

> Thanks Chris, I was totally wrong. Mark Rivers, thanks to him too,
> so sorry. I got confused with n_params.

> I'll never post again (ah I'm more thck skinned than that!)

I certainly don't think it was a dumb question. I'm sorry if I gave that
impression. In fact, it's a very common one, as I said in my original post.

Happy IDLing,

Bill Thompson

Subject: Re: keyword_set bug or feature
Posted by soc on Tue, 16 May 1995 07:00:00 GMT
View Forum Message <> Reply to Message

Chris-Chase-S1A (chase@stewart.jhuapl.edu) wrote:
: >>>>> "Rob" == Stephen O'Connell <soc@festival.ed.ac.uk> writes:

: Rob> Keyword is more than just 1/0 switches! And if you are using them
: Rob> to pass values, you just might mistakenly set one equal to zero.
: Rob> Andd although its true that you can use n_elements, you're still
: Rob> buggered if you have more than one keyword...so it is a bug
: Rob> really, in my opinion. Maybe ver. 4.0 has it sorted - does
: Rob> anyone know ?

: It is not a bug. KEYWORD_SET works exactly as documented in the "IDL
: Reference Guide". Furthermore, the guide states: "This function is
: especially useful in user-written procedures and functions that
: process keywords that are interpreted as being either true (keyword
: present and nonzero) or false (keyword was not used, or was set to
: zero)." Thus, its purpose is not for checking if a keyword is used to
: pass values. Just because KEYWORD_SET does not do what you wish does
: not mean it has a bug. Use

: N_ELEMENTS(var) NE 0

: to check for defined keywords. You can make your own function
: "KEYWORD_DEFINED()" if you think it is more convenient than
: "N_ELEMENTS(var) NE 0".

: Chris

Thanks Chris, I was totally wrong. Mark Rivers, thanks to him too,
so sorry. I got confused with n_params.

I'll never post again (ah I'm more thck skinned than that!)

Rob

Subject: Re: keyword_set bug or feature
Posted by thompson on Thu, 25 May 1995 07:00:00 GMT

soc@festival.ed.ac.uk (Stephen O'Connell) writes:

> Mark Rivers (rivers@cars3.uchicago.edu) wrote:

> : In article <PHIL.95May12090139@peace.med.ohio-state.edu>,
phil@peace.med.ohio-state.edu (Phil) writes:

> : >
> : > IF NOT(keyowrd_set(key)) THEN BEGIN
> : > ;set a default value here
> : > ENDIF
> : >
> : >The problem comes in if the user wants to set key = 0. If so then it
> : >appears to the above test that the keyword is not set even though in
> : >the function call the user typed
> : >

> : keyword_set is intended for use with switches, i.e. parameters which can either
> : be 0 or 1. 0 or not present means switch not set, 1 means switch set.

> : If you want to detect that the keyword is present, even if the value is zero,
> : then you should use n_elements().

> Keyword is more than just 1/0 switches! ...

I agree. There are basically two kinds of keywords, the kind where you can pass back-and-forth any kind of IDL variable, and those which can be either "set" or "not set" (on/off). This distinction is evident from the typical syntax used for the two kinds. For example, in the IDL statement

```
IDL> PLOT, X, Y, XRANGE=[100,200], /YNOZERO
```

the keyword XRANGE is of the first type, while YNOZERO is of the second. One could have done the same thing with YNOZERO=1, but using the /YNOZERO notation makes it clearer that this is an on/off kind of parameter.

A keyword can be considered to be "set" if

1. It is equal to a non-zero value (typically 1).

2. It is passed with the /keyword notation

and it is "not set" if

1. It is equal to zero.

2. It is passed as a non-existent variable.

3. It is not passed at all.

KEYWORD_SET was designed to sort out these possibilities. It does exactly what it was designed to do. The fact that it returns 0 if the keyword was set equal to zero is not a bug but an integral and extremely necessary part of its function.

What you're looking for is not whether a keyword was set or not, but whether it was passed. Recently, somebody posted an IDL procedure called KEYWORD_PASSED which does what you want to do.

You also write

> Andd although its true that you can use n_elements, you're still buggered if
> you have more than one keyword...

I don't understand what you mean by that. You can use N_ELEMENTS separately for each keyword passed. Perhaps you're thinking about N_PARAMS?

Cheers,

Bill Thompson
