Subject: Possible to create 'keyword_used' type function?
Posted by Brian Devour on Mon, 09 Jun 2014 21:11:35 GMT
View Forum Message <> Reply to Message

Hi all,

So, as noted here:

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

checking if keywords were used is kinda a pain in the butt, thanks to the difference between keywords passed by value or reference and the misleadingly-named functions used for these purposes.

At the bottom of the page, there's a nice example of a bit of code that will reliably check if a keyword was used, regardless of whether it was passed by value or reference:

```
pro testme, KEY=k
     if n_elements(k) ne 0 OR arg_present(k) then  $
      print,'You used KEY!' else $
      print,'You neglected KEY!'
  end
```

The operative part, of course, is the test:

if n_elements(k) ne 0 OR arg_present(k) then...

I can and have used this in my programs to check keywords. However, just on an aesthetic level it somewhat annoys me to have a long compound test, as compared to how simple 'if keyword_set(k) then...' is. I would like to create a function named something like keyword_used that would duplicate the functionality of the compound test above, but in a more compact and quicker to type format, so that I could simply do:

if keyword_used(key) then...

in my program instead of doing:

if n_elements(key) ne 0 OR arg_present(key) then...

So, I tried to do this. The first and simplest way to try this that came to mind was this:

```
function keyword_used, key1
if n_elements(key1) ne 0 or arg_present(key1) then return, 1 else return, 0
end
```

This doesn't work; 'keyword_used(key)' in a program always returns 1 regardless of whether 'key' was actually used or not in the program call. I *think* this is because when 'key' doesn't exist in the program, IDL obligingly passes it into keyword_used by reference, hence causing

'arg_present(key1)' inside keyword_used to return true. (I'm not sure if I'm describing that correctly; these things still confuse me somewhat.)

I did some more experimentation with trying to use scope_varfetch and routine_info to try and figure out what variable names were used in various calls and to go up a level and examine the variables existing inside the routine that calls keyword_used, but I relatively quickly got lost in the complexities thereof. (I'm an astronomer, not a programmer, and it shows...) I did some searches on this general topic, but I didn't see anywhere anyone trying this particular task. Is it even possible to write a simple function to do this?

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Chip Helms on Mon, 09 Jun 2014 22:27:40 GMT
View Forum Message <> Reply to Message

With a little bit of messing around (while watching code run) I managed to cobble this together:

```
 ; This function is  equivilent to "n_elements(var) ne 0 or arg_present(var)"
 ; key := variable to be checked
function keyword_used, key
    ; grab list of variables at scope of parent routine
   varnames = scope_varname(key,level=-2, count=nvar)
    ; determine if keyword is used as per arg_present function
    ; also protect against varnames begin undefined
   argtest = (nvar ne 0) ? total(varnames ne '') ne 0 : 0b
    ; return results of test for keyword presence
   return, n_elements(key) ne 0 or argtest
end
```

I don't think it will work unless you're calling it inside of another routine since calling it at $main$ will make it try to look at the level above $main$. You could probably add a check that uses scope_level to determine if keyword_used was called from $main$ (if it was, scope_level will return a value of 2 I think, but I could be wrong). Here's an example routine that uses keyword_used:

```
 ; demonstrate function
pro test, var=invar
if keyword_used(invar) then print, 'KEYWORD_USED indicates keyword present' else $
   print, 'KEYWORD_USED indicates keyword missing'
if n_elements(invar) ne 0 or arg_present(invar) then $
   print, 'Two part check indicates keyword present' else print, 'two part check keyword missing'
end
```

From what I can tell, when you call scope_varname with the variable argument (as done here), it traces the variable back to the requested scope level. So if you use the example routine above by calling "test, var=blah" then scope_varname in keyword_used will start from "key" and see that it was named "invar" at level=-1, and "blah" at level=-2 (i.e. the level at which test was called). It's possible I'm not understanding this correctly, but it seems to be how it behaves.

Cheers,
Chip

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Chip Helms on Mon, 09 Jun 2014 22:42:16 GMT
View Forum Message <> Reply to Message

Here's a cleaner version with some documentation as well.

```
;==============================
; PROGRAMMER: Chip Helms
; DATE: 6/9/2014
; DESCRIPTION: Determines if a variable
;    is being used in any way (either having
;    been given a value or set equal to a named
;    variable as often used with keywords).
;    It is equivilent to using:
;    "n_elements(var) ne 0 or arg_present(var)"
;    Note, the result does not have any meaning
;    when called from $MAIN$ as calling the
;    function automatically guarentees 'key' is
;    set to a named variable or has a value.
;    The exception to this is the trivial case
;    of 'keyword_used()', which will return 0.
; PARAMETERS:
;    key := variable to be tested for use
;==============================
function keyword_used, key
    ; grab list of variables at scope of parent routine
   varnames = scope_varname(key,level=-2, count=nvar)
    ; determine if keyword is used as per arg_present function
    ; also protect against varnames being undefined for some reason
   argtest = (nvar ne 0) ? total(varnames ne '') ne 0 : 0b
    ; return results of test for keyword presence
   return, n_elements(key) ne 0 or argtest
end
```

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Fabzi on Tue, 10 Jun 2014 07:48:49 GMT
View Forum Message <> Reply to Message

Hi,

just out of curiosity: why do you need this? could you provide an
example where it is usefull to know if a keyword is "used"?

Thanks!

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Michael Galloy on Tue, 10 Jun 2014 15:11:09 GMT
View Forum Message <> Reply to Message

On 6/10/14, 1:48 AM, Fabien wrote:
> Hi,
>
> just out of curiosity: why do you need this? could you provide an
> example where it is usefull to know if a keyword is "used"?
>
> Thanks!
>
>

Yes, that was what I was wondering. I take very different actions
depending on whether ARG_PRESENT is true or N_ELEMENTS is greater than zero.

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Brian Devour on Tue, 10 Jun 2014 20:19:34 GMT
View Forum Message <> Reply to Message

I guess I can't say that I really *needed* this, in particular, more just that I figured it ought to be possible to do and ought to exist. I understand why keyword_set, n_elements, and arg_present work the way they do, but it just bugs me that there isn't a more general way to check a keyword in IDL by default, and I was curious how it would be done. And on a pragmatic note, it means that when I forget that I need to check this particular keyword differently (which I do altogether too often), my program wouldn't explode.

So I guess you could call it a combination of curiosity about how IDL works and lazy coding habits. (Which is probably par for the course for less serious IDL users!) I didn't intend to waste anyone's time with a frivolous request or anything; if that's the way I came across I apologize for

that.

Thanks for the code, by the way, Chip. It's a lot tighter than anything I would have come up with, for sure.

Brian

Subject: Re: Possible to create 'keyword_used' type function?
Posted by Chip Helms on Fri, 13 Jun 2014 17:13:07 GMT
View Forum Message <> Reply to Message

No problem. Thanks for distracting me from the monotony of waiting for code to run :)

Chip
