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