Subject: keyword inheritance question
Posted by Matthew Argall on Tue, 14 Jan 2014 00:23:56 GMT
View Forum Message <> Reply to Message

Say I am writing a program (myPro) that uses two other programs (libPro[1-2]) and want to be able to let the user know that the keyword they gave does not exist.

Ideally, I would use _STRICT_EXTRA to do this. I cannot in this case, though.

As a solution, I can put the keywords of libPro1 into the keyword list of myPro. Alternatively, I could just bite the bullet and use _EXTRA, letting unused keywords fall through quietly.

Any advice/preferences/pet peeves on this?

IDL> myPro, /Key1, /Key2

```
----------------
pro libPro2, KEY2=key2
end

pro libPro1, KEY1=key1
end


pro myPro, _EXTRA=extra
    libPro1, _EXTRA=extra
    libPro2, _STRICT_EXTRA=extra    ;Must use _EXTRA!
end
```

Subject: Re: keyword inheritance question
Posted by David Fanning on Tue, 14 Jan 2014 00:54:01 GMT
View Forum Message <> Reply to Message

Matthew Argall writes:

> Say I am writing a program (myPro) that uses two other programs (libPro[1-2]) and want to be able to let the user know that the keyword they gave does not exist.
>
> Ideally, I would use _STRICT_EXTRA to do this. I cannot in this case, though.
>
> As a solution, I can put the keywords of libPro1 into the keyword list of myPro. Alternatively, I could just bite the bullet and use _EXTRA, letting unused keywords fall through quietly.
>
> Any advice/preferences/pet peeves on this?

Oh, don't even get me started on keyword inheritance. It starts off

sounding like *such* a good idea, but it is like adopting a wolf puppy.
What could possibly go wrong?

Then, it turns on you and bites you in the butt. Maybe not right away
(so sweet!), but eventually, and inevitably.

If you are going to use it, I would be very, very careful and ALWAYS
pass the final keyword structure with _STRICT_EXTRA. There is only one
thing worse than something taking a bite out of your butt, and that is
searching for hours and hours for misspelled keywords. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.idlcoyote.com/
Sepore ma de ni thue. ("Perhaps thou speakest truth.")

Subject: Re: keyword inheritance question
Posted by Brian G on Wed, 22 Jan 2014 21:46:24 GMT
View Forum Message <> Reply to Message

Matthew -
  There is the powerful function Routine_Info() that allows you to query the IDL runtime for information about any function or procedure: http://www.exelisvis.com/docs/ROUTINE_INFO.html. So you can call this function on your library procedure names and use the /PARAMETERS keyword to get back an anonymous struct that tells you about the routine signature. If it's a function, you'll also need to include the /FUNCTION keyword or you won't get anything back. Note that if the routine name is unknown you will get an error thrown. You need to make sure you Restore your save files before calling Routine_Info().
  You can use that information to identify which keywords from your myPro wrapper go with libPro1, which go with libPro2, and which are extraneous. Once you've identified which keywords go with each library routine, there are a couple ways you can proceed, but the "easiest" is to copy the values from the _EXTRA struct into a new struct that you pass into the library routines with the _EXTRA keyword. The IDL runtime will then take care of mapping the new struct members into the appropriate keywords in the library routine.
  The following code shows how to do this using hardcoded routine names, though it could probably be abstracted to pass in any number of routine names as a string array parameter.

```
pro libPro1, KEY1=key1
  print, 'in libPro1, KEY1 = ' + (ISA(key1) ? key1 : '<undefined>')
end

pro libPro2, KEY2=key2
```

```
  print, 'in libPro2, KEY2 = ' + (ISA(key2) ? key2 : '<undefined>')
end

pro myPro, _EXTRA=extra
  print, 'in myPro'
  help, extra

  ; first make sure _EXTRA is defined, bail if not
  if (~ISA(extra)) then return

  info1 = Routine_Info('libPro1', /PARAMETERS)
  info2 = Routine_Info('libPro2', /PARAMETERS)

  ; first check for invalid keywords in _EXTRA
  myExtraKeywords = Tag_Names(extra)
  foreach keyword, myExtraKeywords do begin
    if ((Total(keyword eq info1.KW_ARGS) eq 0) && $
        (Total(keyword eq info2.KW_ARGS) eq 0)) then begin
      Message, 'Invalid keyword ' + keyword
    endif
  endforeach

  ; call libPro1 with the appropriate keywords from _EXTRA
  extra1 = {}
  foreach keyword, info1.KW_ARGS do begin
    w = where(keyword eq myExtraKeywords, found)
    if (found) then begin
      extra1 = Create_Struct(extra1, keyword, extra.(w[0]))
    endif
  endforeach
  libPro1, _EXTRA=extra1

  ; call libPro2 with the appropriate keywords from _EXTRA
  extra2 = {}
  foreach keyword, info2.KW_ARGS do begin
    w = where(keyword eq myExtraKeywords, found)
    if (found) then begin
      extra2 = Create_Struct(extra2, keyword, extra.(w[0]))
    endif
  endforeach
  libPro2, _EXTRA=extra2
end

pro keyword_test
  ; first call myPro with no keywords
  myPro

  ; call with only KEY1
```

```
  myPro, KEY1='only key1'

  ; then only KEY2
  myPro, KEY2='only key2'

  ; then both KEY1 and KEY2
  myPro, KEY1='key1 from both', KEY2='key2 from both'

  ; now add bad key, which will throw error
  myPro, KEY1=1, KEY2=2, KEY3=3
end
```
