Subject: Re: _EXTRA keywords not noticed
Posted by J.D. Smith on Tue, 23 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Mirko Vukovic wrote:
>
> J.D. Smith wrote:
>>
>> Another in a long line of complaints about keyword inheritance. It
>> seems that now (though I don't recall this always having been a
>> problem), a wrong keyword is not flagged.
>>
>> E.g.
>>
>> pro testex2,b=b
>> print,'test2'
>> return
>> end
>> pro testex, a=a,_EXTRA=e
>> print,'test1'
>> help,/st,e
>> testex2, _EXTRA=e
>> return
>> end
>>
>> IDL> testex,b=1
>> test1
>> ** Structure <82441b4>, 1 tags, length=2, refs=1:
>> B INT 1
>> test2
>>
>> IDL> testex,a=1,b=1
>> test1
>> ** Structure <81ad99c>, 1 tags, length=2, refs=1:
>> B INT 1
>> test2
>>
>> IDL> testex,a=1,b=1,c=1
>> test1
>> ** Structure <8234884>, 2 tags, length=4, refs=1:
>> B INT 1
>> C INT 1
>> test2
>>
>> This last one should have been an error in testex2! C is not a valid
>> keyword to testex2, but it was silently ignored. I don't think this has
>> always been the case. It's a real problem if you mistype a keyword, but
>> aren't alerted.. you may not realize that your program didn't recieve a

>> keyword argument. Has anyone else noticed this?
>>
>> JD
> i don't see a problem. if c had made it into testex2, yes, but e in
> testex cannot know what routines it will be passed on to.
> --

I don't expect e to know what is inside of it, but when the choice is
made as to where e's keywords will be directed, I recall in prior
versions that the keywords were checked. I even went back and ran IDLv4
to verify. To make my point a little more clear, consider this
different pair of procedures:

```
pro testex2,thisreallycoolkeyword=b
   if keyword_set(b) then print,'The magic answer is 42' else $
    print,'Cupiditas est radix omnia malorum.'
end

pro testex, a=a,_EXTRA=e
   if keyword_set(a) then print, 'A is set'
   help,e,/st
   testex2, _EXTRA=e
   return
end
```

when run:

```
IDLv4:
IDL4>  testex,/a,/thisreallycoolkeyword
A is set
** Structure <225e38>, 1 tags, length=2, refs=1:
   THISREALLYCOOLKEYWORDINT              1
The magic answer is 42

IDLv5:
IDL5> testex,/a,/thisreallycoolkeyword
A is set
** Structure <81c3124>, 1 tags, length=2, refs=1:
   THISREALLYCOOLKEYWORD
                  INT       =        1
The magic answer is 42
```

All is as it is supposed to be there. But suppose I accidentally
mistype the keyword, as:

```
IDLv4
IDL4>  testex,/a,/thisrealycoolkeyword
```

```
A is set
** Structure <225e78>, 1 tags, length=2, refs=1:
   THISREALYCOOLKEYWORDINT             1
% Keyword THISREALYCOOLKEYWORD not allowed in call to: TESTEX2
% Execution halted at:  TESTEX             9 test2.pro
%                      TESTEX            9 test2.pro
%                      $MAIN$

IDLv5
IDL5>  testex,/a,/thisrealycoolkeyword
A is set
** Structure <81c3124>, 1 tags, length=2, refs=1:
   THISREALYCOOLKEYWORD
                  INT       =        1
Cupiditas est radix omnia malorum.
```

As you can see, in v4 my incorrect keyword was flagged, and an error resulted. In v5, things run without interruption, as if I had not provided any extra keywords at all. Of course in my testex2, this is apparent from which message is printed, but for a general routine, the user may receive no indication that his keyword was mistyped, and that, consequently, the program isn't doing what he thinks it is.

On the other hand, this "feature" is eminently useful when inherited keywords are going to be passed to several different routines. You can just give _EXTRA=e to all of them, and only the relevant ones will be used. It's not exactly that straightforward, though. If two keywords are the same, or even begin the same, in these two or more routines, then you'll get no warning, and both will be called... an example of this is:

```
pro testex3,key3=c
   if keyword_set(c) then  $
    print,'Ein spatz in der Hand is besser als eine Taube auf dem Dach'
else $
    print,'I like candy.'
end

pro testex2,key2=b
   if keyword_set(b) then print,'The magic answer is 42' else $
    print,'Cupiditas est radix omnia malorum.'
end

pro testex, _EXTRA=e
   help,e,/st
   testex2, _EXTRA=e
   testex3, _EXTRA=e
```

```
  return
end
```

And now (v5):

```
IDL> testex
E               UNDEFINED = <Undefined>
Cupiditas est radix omnia malorum.
I like candy.

IDL> testex,/key2
** Structure <81c35ec>, 1 tags, length=2, refs=1:
  KEY2          INT             1
The magic answer is 42
I like candy.

IDL> testex,/key3
** Structure <81c360c>, 1 tags, length=2, refs=1:
  KEY3          INT             1
Cupiditas est radix omnia malorum.
Ein spatz in der Hand is besser als eine Taube auf dem Dach

IDL> testex,/key2,/key3
** Structure <81c360c>, 2 tags, length=4, refs=1:
  KEY2          INT             1
  KEY3          INT             1
The magic answer is 42
Ein spatz in der Hand is besser als eine Taube auf dem Dach
```

all as they should be. But what happens with:

```
IDL> testex,/key
** Structure <81c3634>, 1 tags, length=2, refs=1:
  KEY           INT             1
The magic answer is 42
Ein spatz in der Hand is besser als eine Taube auf dem Dach
```

Egad! As you can see, the root of the keywords being the same in both routines, both recieved it! This is actually the same behavior in V4, but no less a cause for being careful.

What would be quite useful, although likely equally impossible, is a system by which any leftover inherited keywords, that is keywords which *don't* get through to any internal routine calls, are flagged as errors. One can only dream.