Subject: Re: Keyword precedence
Posted by John-David T. Smith on Sat, 26 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Mark Hadfield wrote:
>
> "J.D. Smith" <jdsmith@astro.cornell.edu> wrote in message
> news:39A6B82E.2533B5A@astro.cornell.edu...
>> .....
>
> Whew! Thanks for that explanation, JD. I think I understand it now.
>
> First an executive summary for David:
>
> * IDL's current behaviour regarding precedence of explicit keywords vs those
> passed through from above by inheritance is too inconsistent to be useful.
> It requires the caller to know details about inheritance mechanisms further
> down the chain, which is highly undesirable.
>
> Now a few questions/comments:
>
>> You can actually see this happening if you throw an "print,
>> arg_present(color)" into KEYWORDS_PRINT_COLOR.
>
> I did. It prints 0 in all 3 cases. What does this tell me? ARG_PRESENT is
> only supposed to return 1 if a named variable is supplied. I didn't supply
> one.

I guess I neglected to say if you do pass a named variable in addition to the
explicit _EXTRA={}.

>
>> Of course, only one of these can be used, and as it
>> happens, the fully by-reference variable is chosen.
>
> This is the crux, isn't it? Could & should this design choice be changed? If
> it were, would this lead to any other surprising results?

I commented briefly before to indicate that really this should not be changed,
but _REF_EXTRA should be changed to detect duplicate keyword passings and make
it an error. Without this error, the behaviour is confusing, I agree. RSI went
to far to ease acceptance of _REF_EXTRA, I belive.

>
> This is my main point and I could leave it there but I'll add another
> tidbit:
>
> I modified MGH_EXAMPLE_KEYWORDS & played with it a

> bit more. I've put the modified code on the WWW page and attached it to this
> message. Now the main routine accepts _EXTRA keywords from its caller. Its
> usage now reflects the intention of the whole exercise better. The idea is
> that we want to pass COLOR=0 through to the printing routine but let the
> user override it. (I hope nobody thinks I ever actually *write* code like
> "some_procedure, COLOR=0, _EXTRA={color:12}".)
>
> You might like to try the following set of calls:
>
> mgh_example_keywords
> mgh_example_keywords, COLOR=12
> mgh_example_keywords, COL=12
>
> No surprises with the first one: the default value of 0 is passed through
> and printed.
>
> With the second one we get the result that's being discussed, the user's
> value of 12 overrides the default and is printed except when the "reference
> wrapper" is involved in the calling chain, in which case 0 is printed.
>
> With the third one, 12 is printed in every case! My interpretation: IDL's
> handling of keyword abbreviations is such that an abbreviated keyword takes
> precedence over a non-abbreviated one, and this overrides the "fully
> by-reference first" rule. Which certainly suggests that the "fully
> by-reference first" rule is not cast in stone.

I too played with this and discovered the same property, which I negelected to mention to avoid furthing muddying the waters. I didn't want to imply there was any such "fully by-reference first" rule, but rather that there is no rule at all (and should in fact be an error, I believe). The progression of this unfortunate situation probably had to due with RSI's discovery that _REF_EXTRA is faster than _EXTRA, so why not use it for plain by value passing too, they thought (so long as you're not going to unintentionally modify the value). I think this was a slippery slope, since overriding a *value* is very different than overriding a *variable*. The former is relatively more straightforward. If RSI had stuck to a _REF_EXTRA used only for returning values up inherited keywords in chains of calls, we wouldn't have this ambiguity... it really doesn't make sense to override the *location in memory* associated with a given inherited keyword variable at runtime. But, now that the worms are out of the can, I suppose we'll have to deal with it. Since IDL can tell whether _REF_EXTRA is being used by value or by reference (or by some combination), it can use sensible rules for overriding. I'd reccommend:

1) 2 or more by-value and 0 by-reference(arg_present==0): Default to the standard _EXTRA rules for overriding and abbreviations (Longest match first).

2) 1 or more by-value and 1 by-reference: always pass the variable, by-reference. Useful for modifying a value *and* returning a result.

3) 0 or more by-value and more than 1 by-reference: generate an error. Not too burdening since you have to go out of your way to achieve this situation.

In any case, just beware of mixing your _EXTRA metaphors in the meantime.

JD

JD

--
J.D. Smith /*\ WORK: (607) 255-6263
Cornell University Dept. of Astronomy */ (607) 255-5842
304 Space Sciences Bldg. /*\ FAX: (607) 255-5875
Ithaca, NY 14853 */