Subject: Re: Keyword precedence
Posted by Mark Hadfield on Mon, 28 Aug 2000 22:07:03 GMT
View Forum Message <> Reply to Message

"Martin Schultz" <martin.schultz@dkrz.de> wrote in message news:39AA2492.6ADB8E0@dkrz.de...
> Mark wrote at some point:
>
>> I think the rule should be, where a choice has to be made between duplicate
>> keywords, always choose the last one in the list.
>
> now for the slow-movers: does this mean that
> wrapper, color=12, _Extra=e
> and
> wrapper, _Extra=e, color=12
>
> should yield different results if there is a color tag in the
> extra structure?

You mean different results from each other, don't you? Definitely not! The "list" here is the list of "extra" keywords inside wrapper (where--to reiterate--wrapper uses reference inheritance and doesn't have a color keyword itself). The information passed in via _extra always goes onto the end of this list. Position in the call doesn't matter (I checked).

> Although this may seem overly formal, I would argue that the
> "convenience" gained by overriding keywords that are explicitly
> set with values in the extra structure does not aid in writing
> "clean" programs. If you are messing with any particular property
> in your routines, this property should always be identified by
> having its own keyword in my opinion. So, if you want a default
> color, the procedure header should be something like:
>
> pro wrapper, color=color, _EXTRA=e
>
> IF N_Elements(color) EQ 0 THEN color = 12
> plot, x,y, color=color, _EXTRA=e
>
> end
>
> and the "real" color keyword should be used (which I think and
> hope it gets).

David Fanning said the same thing. I see your point, but I think I still favour the overriding form:

```
pro wrapper, _EXTRA=e
```

```
    plot, x,y, color=12, _EXTRA=e
end
```

because it's shorter (much shorter if several properties are overridden) and therefore clearer. However I've been happily writing code in the style you advocate for the last couple of years, so it can't be all that bad.

Actually my reason for wanting to rely on keyword precedence is a little more complicated, and a little more compelling, than what I described in my previous posts. For the last week--while I haven't been writing long posts on keyword precedence--I have been preparing figures and animations for a conference presentation. I have been using widget applications that I wrote some time ago to visualise hydrodynamic model output. These applications create various object graphics elements, e.g.:

```
pro my_visualisation
    theview = new_view(UNITS=2, DIMENSIONS=[10,10])
    thexaxis = new_axis(DIRECTION=0, TICKFORMAT=...)
    theyaxis = new_axis(DIRECTION=1, TICKFORMAT=...)
    ; Add the axes to a model & add that to the view
    ; Get data
    thesurface = new_surface(STYLE=2, DATAX=...)
    ; add the surface to the model
    new_window, GRAPHICS_TREE=theview, RETAIN=2 ; (sorry Randall)
end
```

where the "new_" functions and procedures are wrappers of some sort for the object-creation functions.

This has been fine for visualisation purposes, but for presentation I wanted to tweak the results: adjust the size and colours for the output medium, get rid of extraneous tick labels and reduce the size of the margins, stuff like that. But how to get all that information into the application without adding a huge number of keywords to the routine? I found the solution in an example provided with IDL 5.4 beta, but I see that it's also used by the LIVE_STYLE routine: for each element foo add a single keyword, foo_properties, thus:

```
pro my_visualisation $
    , VIEW_PROPERTIES=view_properties $
    , XAXIS_PROPERTIES=xaxis_properties $
    , YAXIS_PROPERTIES=yaxis_properties $
    , SURFACE_PROPERTIES=surface_properties $
    , WINDOW_PROPERTIES=window_properties

    theview = new_view(UNITS=2, DIMENSIONS=[10,10], _EXTRA=view_properties)
    thexaxis = new_axis(DIRECTION=0, TICKFORMAT=...,
_EXTRA=xaxis_properties)
```

```
    theyaxis = new_axis(DIRECTION=1, TICKFORMAT=...,
_EXTRA=yaxis_properties)
    ; Add the axes to a model & add that to the view
    ; Get data
    thesurface = new_surface(STYLE=2, DATAX=..., _EXTRA=surface_properties)
    ; add the surface to the model
    new_window, GRAPHICS_TREE=theview, RETAIN=2, _EXTRA=window_properties
end
```

Now to control (say) the X axis the caller just sets xaxis_properties equal
to a structure containing the appropriate keyword:value pairs, e.g.:

```
my_visualisation, XAXIS_PROPERTIES={notext:1, minor:0}
```

Now this approach obviously relies on the "_properties" structures
overriding the defaults. When I first tried this I found that the
xaxis_properties keyword was not working and that this was because I had
happened to use inheritance by reference in new_axis. The rest (as they say)
is history...

> I agree with JD that IDL should be stricter and issue an error if
> it finds more than 1 keyword of the same name in a _ref_extra
> list, and perhaps it should even be possible to generate an error
> if the two identical keywords appear in the _extra list (another
> compile_opt flag?).

Well, I think that raising an error is a little harsh and that, as I've
said, some form of consistent precedence rule would be better. But raising
an error is certainly better than IDL's current behaviour, which is silently
inconsistent.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield/
National Institute for Water and Atmospheric Research
PO Box 14-901, Wellington, New Zealand