Subject: Re: undefined keyword variables
Posted by Liam Gumley on Sun, 31 Oct 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Mark Fardal <fardal@weka.astro.umass.edu> wrote in message
news:7vbt9gk4jk.fsf@weka.phast.umass.edu...
> A question: should you always be able to pass undefined variables as
> keywords to IDL routines?
>
> For example, PLOT is smart enough not to do anything with this undefined
> variable
>
> IDL> plot,x,y,title=donkeykong
> [plots fine]
>
> but not this one
> IDL> plot,x,y,clip=pong
> % PLOT: Variable is undefined: PONG.
> % Execution halted at: $MAIN$
> [no plot produced]
>
> IDL> help,donkeykong, pong
> DONKEYKONG UNDEFINED = <Undefined>
> PONG UNDEFINED = <Undefined>
>
> Should this be considered a bug in plot, or as normal behavior?

A bug, no. Inconsistent behavior, maybe.

>
> In general I don't know why you should be able to safely feed
> undefined variables to routines and expect them to work. But if you
> can't, it leads to annoying problems in writing interfaces to any
> routine with lots of keywords. In my case, I wanted to write a
> routine that called plot and then oplot, which use different sets of
> optional keyword parameters. The best solution I know of is to
> construct a value for the _extra keyword, but it's a pretty convoluted
> solution.

The most obvious reason to pass undefined variables as keywords is when they are supposed to be *set* in the called procedure or function. If output keywords are passed, you check them like this:

```
if n_elements(key) gt 0 and arg_present(key) ne 1 then $
  message, 'Output keyword KEY is an expression and cannot be set'
```

Keywords that are used as *input* arguments fall into two classes.

First, they can be true/false flags. In this case, they do not need to be checked. All you need to do is use KEYWORD_SET whenever the keyword is used, e.g.

```
if keyword_set(ps) then begin
  current_device = !d.name
  set_plot, 'PS'
  device, /landscape, /color, bits=8
endif
```

Second, they can be input values. In this case, they should be checked to see if they are defined, and if not, then default value(s) should be assigned, e.g.

```
if n_elements(range) eq 0 then begin
  minvalue = min(data, max=maxvalue)
  range = [minvalue, maxvalue]
endif
```

If you wish to pass extra keywords to PLOT or OPLOT or any other routine, you only need to check the keywords which you explicitly wish to use in your procedure. For example, here is a PLOT/OPLOT wrapper which lets you control plot colors more easily:

```
;---cut here---
PRO SPLOT, X, Y, $
  BACKGROUND=BACKGROUND, CHARSIZE=CHARSIZE, $
  COLOR=COLOR, WINCOLOR=WINCOLOR, PLOTCOLOR=PLOTCOLOR, $
  _EXTRA = extra_keywords

;- Check arguments
if n_params() eq 0 then message, 'Usage: SPLOT, Y or SPLOT, X, Y'
if n_elements(x) eq 0 then message, 'Argument Y is undefined'

;- Check keywords
if n_elements(background) eq 0 then background = 0
if n_elements(charsize) eq 0 then charsize = 1.0
if n_elements(color) eq 0 then color = !d.table_size - 1
if n_elements(wincolor) eq 0 then wincolor = background
if n_elements(plotcolor) eq 0 then plotcolor = !d.table_size - 1

;- Save first element of !p.multi
multi_first = !p.multi[0]

;- Erase screen and establish plot position
plot, [0], /nodata, xstyle=4, ystyle=4, $
  background=background, charsize=charsize
pos = [!x.window[0], !y.window[0], !x.window[1], !y.window[1]]
```

```
;- If Postscript output, fill background
if !d.name eq 'PS' and multi_first le 0 then $
  polyfill, [0.0,1.0,1.0,0.0], [0.0,0.0,1.0,1.0], $
    /normal, color=background

;- Fill plot window background
polyfill, [pos[0], pos[2], pos[2], pos[0], pos[0]], $
          [pos[1], pos[1], pos[3], pos[3], pos[1]], $
          /normal, color=wincolor

;- Plot the data points
case n_params() of
  1 : begin
        plot, x, /noerase, /nodata, color=color, $
          position=pos, charsize=charsize, _extra=extra_keywords
        oplot, x, color=plotcolor, _extra=extra_keywords
      end
  2 : begin
        plot, x, y, /noerase, /nodata, color=color, $
          position=pos, charsize=charsize, _extra=extra_keywords
        oplot, x, y, color=plotcolor, _extra=extra_keywords
      end
endcase

END
;---cut here---
```

Notice that the keywords I want to use (BACKGROUND, CHARSIZE, COLOR) are listed explicitly in the arguments to this procedure. Any other keywords are stuffed into EXTRA_KEYWORDS. There is no problem passing the same EXTRA_KEYWORDS structure to both PLOT and OPLOT; they just use any keywords they recognize, and ignore the rest.

Using the procedure above, and if you grab my COLORS routine from http://cimss.ssec.wisc.edu/~gumley/idl/colors.pro, you can do this:

```
IDL> device, decomposed=0
IDL> colors
IDL> x = findgen(200) * 0.1
IDL> y = sin(x)
IDL> splot, x, y, background=13, wincolor=7, plotcolor=1, color=0
```

That is, you can control the colors of the plot background, the plot window, the plot axes, and the plotted data independently. Works in Postscript too.

Cheers,
Liam.