Subject: Re: Application programming--missing features
Posted by hadfield_m on Mon, 19 Apr 1993 01:14:40 GMT
View Forum Message <> Reply to Message

Jeff de la Beaujardiere (jdlb@kukui.ifa.hawaii.edu) writes:

> Adding a "wrapper" to an intrinsic IDL routine is difficult.

> For example, consider William Thompson's <thompson@serts.gsfc.nasa.gov>
> just-posted routine PLOT_DROP for dropping bad data values when plotting
> data. That procedure, in effect, just adds a single keyword DROP_VALUE
> to the generic PLOT routine.
>
> In such an application, for each of the usual optional parameters
> accepted by PLOT one must make an entry in the procedure declaration and
> properly pass the parameter to PLOT. This involves either (a) defining
> defaults for each option or (b) tediously building up a command line and
> passing it to the EXECUTE function.

and Ray Sterner (sterner@tesla.jhuapl.edu) writes:

> There is an easier way. Its not perfect, but it is much better than
> then the tedious technique described above (which I've done myself
> in the past). The key is the IDL execute function, mentioned above.
> To show the technique I will give an example custom plot routine
> that just puts a color band behind the plot curve. This routine
> adds one new keyword to the plot routine:

[The routine "tplot" defines a single keyword "args" via which PLOT keywords
can be entered in a character string.]

> All the plot options are available as far as I know. The reason
> this technique is not perfect is that the normal plot keywords must
> all be given inside a text string, unlike the normal plot call.

Thanks to the contributors to this thread for an enlightening and
interesting discussion. I have tried to summarise some pros & cons for each
of these 3 methods below. Perhaps people would like to point out anything I've
forgotten or got wrong:

(a) Define defaults for each option:

Tedious--all keywords to be passed through must be listed in 3
places: the definition of the wrapper procedure, setting defaults,
the call to the wrapped routine.

PLOT has approx 64 input keywords (Reference Guide Jan 93 pp 1-159ff)
plus 3 output keywords. If you want to pass all the input keywords

though the wrapper routine, the call to PLOT therefore has 64 keywords. As far as I can tell there is an IDL limit on the number of keywords that can be passed to a routine of approx 60 (at least there is in IDL for Windows 3.0.1). Of course you could surely leave one or two out, but if you're going to implement most of them it is neater to implement all of them.

For some keywords, finding an appropriate default to set is problematical. For example the PLOT keyword position--see my recent post on "PLOT keywords vs System Variables". (This may not be insuperable but it's certianly irritating.)

(b) Building up a command line and passing it to the EXECUTE function:

Tedious--all keywords to be passed through must be listed in 2 places: the definition of the wrapper procedure, and the code to add each keyword (if defined) to the command string. Arguably it's less tedious than (a) because once the keywords are listed, the code can be built up by cutting & pasting in an entirely mechanical way.

EXECUTE is allegedly inefficient. (I don't think that this is an important issue for the kind of procedure we're considering here.)

Since EXECUTE can't be called recursively the wrapper procedure can't itself be wrapped in the same way.

As far as I am aware, there are no restrictions on the number of characters in a string fed to EXECUTE.

(c) Pass all keywords to the wrapped routine via a string.

MUCH less tedious to program than either of the above.

The calling syntax for the wrapper procedure is non-standard.

Variable names in the keyword-string aren't defined inside the procedure. For example, with Ray Stener's tplot, consider the following:

```
< x=findgen(100)
< y=x^2
< tplot,x,y,'xrange=[2,10]',back=128
 [Nice-looking plot appears]
< a=[2,10]
< tplot,x,y,'xrange=a',back=128
 [Plot appears without axes!]
% PLOT: Variable is undefined: A.
% PLOT: Variable is undefined: A.
```

    Of course one could patch the value of a into the keyword string
    but that's more trouble than it's worth.

    Restrictions on EXECUTE apply as for (b).

As far as I am concerned the provisional winner for most purposes is (b), because it preserves (almost) all the functionality of the wrapped routine. The wrapper procedure is certainly verbose, but largely transparent--you don't have to know about the default behaviour for each keyword to see if it's going to work.

Of course I'd be delighted to be told I've missed the point somewhere and there's a much better way of doing things.

I concur with Jeff that IDL is a lovely package/language in many ways. To someone who's done most of his scientific graphing using the NCAR Graphics Fortran routines, it's a real eye-opener.

```
 ------------------------------------------------------------ ------
| Mark Hadfield                  hadfield@wao.greta.cri.nz  |
| NIWA Oceanographic (Taihoro Nukurangi)                     |
| 310 Evans Bay Rd, Greta Point      Telephone: (+64-4) 386-1189 |
| PO Box 14-901, Kilbirnie           Fax:       (+64-4) 386-2153 |
| Wellington, New Zealand                                    |
 ------------------------------------------------------------ ------
```