Subject: Re: Application programming--missing features
Posted by chase on Fri, 16 Apr 1993 20:58:07 GMT
View Forum Message <> Reply to Message

In article <JDLB.93Apr15173546@kukui.ifa.hawaii.edu>
jdlb@kukui.ifa.hawaii.edu (J-F Pitot de La Beaujardiere) writes:

> 1) 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.

I agree that a wrapper intrinsic would be very helpful. I quite often
try to extend IDL functions and I often end up doing (b) above to
build up a command line to pass to EXECUTE. Indeed (b) is very
tedious. Typically when I use (b), I end up not allowing for all
possible keywords so my new "wrapper" function can not be used as a
complete substitute for the original. Additionally, it would never
allow use of new functionality added to the wrapped procedures via new
keywords without having to be updated. A "wrapper" ability would avoid
this. (Gee, it sounds like I want inheritance and overloading as found
in object oriented programming).

But how would you implement this? Would IDL allow a person to specify
any keyword that isn't defined in your procedure? It seems that it
would have to be done as a modification in the IDL kernel. You might
not want to allow arbitrary keywords to be given to an arbitrary
procedure because it would reduce the error checking ability.

One possible implementation:

On the other hand, if you did allow this, it could be implemented like
the UNIX Bourne shell, where additional keyword parameters on a
command line become part of the shell environment. In IDL's case
additional undefined keyword parameters could be placed in a system
variable table reserved for keywords and local to the called
procedure. Then your "WRAPPER" intrinsic function could just be
another form of EXECUTE that adds those keyword parameters to the

command. Actually, new versions of CALL_PROCEDURE or CALL_FUNCTION
would be sufficient with an optional keyword dictating that keyword
parameters stored in the local "environment" be added to the called
procedure or function. Depending on how procedure calls are compiled
by IDL this could require substantial changes in the IDL kernel.

> 2) User-defined global variables for customizing program behavior
> do not exist.
>
>> The only two options are to (a) define a new system variable
>> using DEFSYSV or (b) use common blocks. Option (a) fails
>> because N_ELEMENTS(!FOO) returns an error ("Not a legal system
>> variable") instead of zero if !FOO is undefined. Option (b) is
>> very tedious for both programmer and user because common blocks
>> are finicky beasts.
>>
>> The simplest solution would be to modify IDL such that
>> n_elements(!foo) returns 0 if !foo is undefined.

I agree that globals ala system variables would be a very useful
option. I have felt that using common blocks would not work. In fact,
I was not aware previously that (a) was possible. But the n_elements()
certainly does make that unuseable.

1) and 2) would indeed be very useful features.

I wonder if people at RSI or PVI monitor this Newsgroup's posts?

Later,
Chris
--
=============================
Bldg 24-E188
The Applied Physics Laboratory
The Johns Hopkins University
(301)953-6000 x8529