Subject: Re: Top 10 IDL Requests
Posted by Craig Markwardt on Mon, 17 Jul 2000 07:00:00 GMT
View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) writes:
> ... I think Craig in particular should have a
> look at the new curve fitting routines.

That sounds great, although I'm pretty attached to my own MPFIT
functions by now :-)

As for my top ten wish list, you probably will wish I hadn't gotten
started. Okay here's my list. It's more than ten, so I guess I'm
already starting to compete with myself. I focus mainly on the basic
programming elements which need to be improved, rather than the
feature-level stuff that we can program ourselves.

* Arrays in structures can collapse to scalars? A BUG!

  This is a problem for structures declared like this:
```
    X = {VAL: REFORM(DBLARR(1,1),1,1)}
```
  Whenever you try to extract VAL, it will come out as a scalar! This
  is clearly a language bug that must be fixed. J.D. will disagree
  but that doesn't make him more right. :-)

* The concept of an empty array (null data type).

  I've mentiond this before. It would avoid special cases for WHERE()
  commands. Also, it would make it much easier to do list processing,
  such as extraction and concatenation. Ah, Patrick Broos has the
  same idea. Like he says, we couldn't get along without the empty
  string. So why do we not have the empty array?

* The ability for WHERE to return the *complement* of the selection
  simultaneously. I do this fairly occasionally:
```
   WH1 = WHERE(X GT 0, CT1)
   WH2 = WHERE(X LE 0, CT2)
```
  Which involves two comparisons instead of one. This can be a big
  hit with large arrays. The syntax could be like this:
```
   WH1 = WHERE(X GT 0, CT1, COMPLEMENT=WH2, CT2)
```

* Short-hand notation to index arrays from the end.

  It pains me to have to use N_ELEMENTS() to find out the size of an
  array before indexing it. It's even more difficult for a
  multi-dimensional array. Shouldn't the array itself know how big it
  is?

I've wondered whether we could just be allowed to use "*" in an expression, like MATRIX(0:*-1) in place of MATRIX(0:N_ELEMENTS(MATRIX)-2), but this has problems if multiplication is used.

* Optional strict array sizing.

  A simple keyword -- such as STRICT -- to all of the array constructors to prevent them from automatically dropping any trailing dimensions of size 1. Please! I know what size my arrays should be. Don't work against me here. I would much prefer to do this:

```
    X = DBLARR(NX, NY, NZ, /STRICT)
```

  rather than this:

```
    X = REFORM(DBLARR(NX, NY, NZ), NX, NY, NZ, /OVERWRITE)
```

* Consistent behavior for REFORM, REBIN, TOTAL and TRANSPOSE.

  Make sure that REFORM, REBIN and TOTAL work the same way on scalars and arrays. For example, why doesn't REFORM(1,[1,1]) work? Yes, sometimes I *do* need a 1-element, multi-dimensional, array.

  Also, a convention for passing dimensions to these routines should be codified and enforced. For example, REBIN(ARRAY, [2,3]) does not work, but REFORM(ARRAY, [2,3]) does. Can that make sense?

* Consistent ways to compile procedures and functions.

  There should be programmatic ways to compile procedures and functions. RESOLVE_ROUTINE helps, but for example it only compiles procedures in the IDL path. There is no way to explicitly designate the path to the code, which is especially vexing if your code is in a scratch area.

  By the same token, it should be possible to compile an entire procedure from memory (as opposed to from disk).

* A way to index strings like arrays.

  I know we can use STRMID and STRPUT, but it seems that an array-like notation would fit so much better with the philosophy of IDL.

* A way to construct procedures which accept a variable number of parameters.

  The particular context I'm talking about here is some kind of wrapper procedure, which in turn calls another procedure. You end up writing something like this, which seems foolishly redudant and artificially limiting:

```
  CASE N_ARGS OF
   0: CALL_PROCEDURE, PROC
   1: CALL_PROCEDURE, PROC, ARGS.(0)
   2: CALL_PROCEDURE, PROC, ARGS.(0), ARGS.(1)
   3: CALL_PROCEDURE, PROC, ARGS.(0), ARGS.(1), ARGS.(2)
  ENDCASE
```

* A way to selectively pass a few keywords via the _EXTRA mechanism.

The _EXTRA mechanism for keyword passing is great. However, it is most useful when you are passing arguments to *one* *single* internal function. There are times when I have several internally called functions which may accept different keywords. A way to filter keywords would be convenient.

* A way to do one-click printing under Unix.

My solution was XFWINDOW. It's a hack, nobody on the net seems to like it, but I use it every day. I hate using SET_PLOT and DEVICE, and I always get it wrong. A direct graphics window should have a "print" button right *on it*.

Sorry for the long message, but you asked for it! :-)

Constructively,
Craig

--
-- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
-- --------------