Subject: Re: RSI / CreaSo survey: Whish list
Posted by steinhh on Thu, 07 Dec 1995 08:00:00 GMT
View Forum Message <> Reply to Message

In article <4a4pit$svq@rs18.hrz.th-darmstadt.de>, hahn@hrz.th-darmstadt.de (Norbert Hahn) writes:
|> I started to compile a list of improvements. Here come my thoughts,
|> unsorted:
|>
[..specific wishes..]
|>
|>
|> More ideas ???

Since I can see that your wishes are along totally different directions than mine (not that your wishes are less welcome), I thought I'd offer my suggestions as well. I'll just copy in what I said to Creaso, so bear with me if it's a bit wordy and unstructured (it even involves coming up with a new pseudo-language on the fly).

---------- ------

1. Pointers that work like *real* pointers (handles are *almost* there, just need some syntactic sugar on top).

E.g., if p is a pointer to a (possibly large) struct, I want to be able to use one simple syntactic element to refer to one of it's tag values, like a=my_function(p->tag), without having to use

HANDLE_VALUE,P,TEMP,/NO_COPY
a=my_function(TEMP.tag)
HANDLE_VALUE,P,TEMP,/SET,/NO_COPY

I can se no reason why this hasn't already been implemented, and in my opinion this is the most serious flaw in the language. One needn't use the construct "p->", something like "~p" or "p~" could be used instead, to allow for a more sensible look when using pointers to e.g., scalar variables: If p is a pointer to a scalar, then the statement would read "a=my_function(p~)". Reference to a tag in a structure would become ~p.tag or (perhaps less ambiguous) p~.tag. I'm actually going to be quite disappointed if this ability is not built into IDL pretty soon.

2. A coherent way of deciding the result dimensions in indexed operations, plus removing the "feature" of removing all trailing singular dimensions: If a is a fltarr(10,10) i want a([0],5) to be a fltarr(1), a([0],[5]) to be fltarr(1,1) and a(0,5) to be a scalar.

3. Possibility of Macros, e.g.:
#define ABORT(text) BEGIN & MESSAGE,TEXT,/CONTINUE & RETURN,-1 & END

4. Possibility of having widget applications + command line available
at the same time (like the help facility).

5. Different brackets for array subscripts and function calls.

6. A portable pseudo-language to produce true *compiled* subroutines,
much in the same way as CALL_EXTERNAL routines are used. This would
require strong typing of the procedure parameters + variables. In
addition, operations could be restricted to operate on scalars.
For some operations compiled code would speed up computation enormously
(e.g., taking an array a = findgen(100) and computing the array b,
where b(i) = a(0)*a(1)*a(2) .... a(i). This language could even
be designed to function as an "inline" language, so that I could write
e.g.,

```
 ;; Normal IDL statements
 a=fltarr(10)
 readu,unit,a
 b=fltarr(10)
 COMPILEBLOCK(A : FLTARR(N), B : FLTARR(M))
                ;; N and M would automatically become LONGs containing
                ;; the number of elements in A and B respectively.
   I : LONG
   J : LONG

   IF M NE N THEN COMPILE_ERROR("This is not allowed to do")
   B(0) = 1
   FOR I = 1L,M-1 DO BEGIN
      B(I) = B(I-1) * A(I)
   END
 END
```

Even with a very restricted pseudo-language, which would be easy
to compile, most of the operations that would otherwise need loops
could be speeded up enormously.

Stein Vidar