Subject: Re: Specification for a new array slicing function
Posted by Martin Schultz on Thu, 20 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> In article <374317CC.E1AC89EA@ssec.wisc.edu> Liam Gumley
> <Liam.Gumley@ssec.wisc.edu> writes:
>
>> Please find below a suggested specification for a new array slicing
>> function,
> [...]
>> ; result = array_slice(array, stride=stride)
>> ; help, result
>> ;
>> ; ;RESULT   FLOAT   = Array[1, 5, 2, 3, 3]
>
> IMO, the use of keyword parameters for START, STRIDE and
> COUNT is a bit "wordy" for my liking. And these items
> are really essential to the routine as such. So why not
> use positional parameters?
>
> For something that really ought to be a part of the IDL
> syntax, I would also like a shorter name (despite the
> possibility for name conflicts), like "arex", short
> for array_extract.

good points. Although I would add one more character and name it "arrex"
to avoid confusion with "ar"gument or "ar"ea etc. (only "arr"ow left
then ;-)

>
> My suggestion would be something a bit more like the native
> Fortran 9X syntax (not that I actually *know* exactly how that
> syntax works!) , e.g.:
>
> a(0:5:2,:,5:9) would be translated into
>
>   arex(a,[0,5,2],-1,[5,9])
>
Sounds nice, however, this is truely up to RSInc to implement. I assume
Liam's proposal was something we, the community, could do ourselves.

Anyway, I asked our Fortran 90 expert, and he told me the following:
- the 3 optional parameters work exactly like a DO (IDL=FOR) loop, i.e.
you have start, end , stride
- you can leave any of them empty which is implicitely defaulted to all,
all, 1

- a statement like A(::1,LM) = A(::LM,1,-1) reverses the last dimension

If RSInc would go for this, I think they should try to use the same conventions. It's already bad enough to have to rethink DO and FOR each time you change.

> Looking at the example above, you may wonder what the "-1" is
> doing there... Well, the idea is that one could use a
> nonnegative *scalar* parameter to signify extraction of a
> slice at a given position, whilst -1 really means "*", in IDL
> notation.

Then, why shouldn't it be "*" as always ?
(or even better, allow the empty field as in F90: A[:1:-1] would be identical to
   reverse(A[1:*]) in the current syntax)

```
>
> I mean - if I'm extracting an "image" out of a "cube", why
> would I want the last dimension to stick around...???
>
> So, I would like to be able to say
>
>     surface,arex(a,-1,3,-1)
>
> with no error messages! On the other hand, if I do want the
> dangling dimension, I could specify it:
>
>     surface,arex(a,-1,[3],-1)
>
```

this seems to be somewhat messy: the "syntax" would rather be
   ARRAY[start1:end1:stride1, start2:end2:stride2, ... ,
start8:end8:stride8]

instead of ARRAY[[s1:e1:str1],[s2:e2:str2], ...]
So, I don't think A[:3:] would (and should) be different from A[:[3]:]
You'll probably have to stick with good old REFORM for this.

> I would also like to see a corresponding index function,
> returning the one-dimensional indices to the extracted
> elements instead of the elements themselves. This could
> be used for assignments. I.e.:
>

```
>      a(arexi(a,-1,[3],[0,2])) = data_block
```

More generally, this points to the problem of converting 1-dimensional index arrays (as from WHERE) to multi-dimensional arrays and vice versa. We had a related discussion in this group a while ago. If I remember correctly, this was about what people expect from

```
   A[ ind1, ind2, ind3 ]   where ind1, ind2, ind3 are 1-dimensional
```

vectors > 1 element.

Here is what I see:
(1) multi-dimensional index

```
   a = findgen(10,10,10)
   b = lonarr(2,3,4)
   ; fill b with some values
   b[*,1,4] = 3
   help,a[b]
   print,a[b]
```

BUT is b not in fact interpreted as a 1-D index? Suspicion arises because a[b,1,1] will
also work (and return a 1D array).

(2) combi of 1-dimensional indices

```
   a = findgen(10,10,10)
   b1 = [1,2]
   b2 = [2,3]
   b3 = [4,8]   ; don't try b3=[3,6,7] !
   help,a[b]
   print,a[b]
```

So, YES! It would be nice if one could use a multi-dimensional array index, but there are several pitfalls here, and it appears as a non-trivial problem.

Regards,
Martin

--

|||||||||||||||\\\\\\\\\\\\\-------------------////////////// //||||||||||||||
Martin Schultz, DEAS, Harvard University, 29 Oxford St., Pierce 109,
Cambridge, MA 02138   phone (617) 496 8318   fax (617) 495 4551
e-mail mgs@io.harvard.edu   web http://www-as/people/staff/mgs/