Subject: Re: Still missing features in IDL 8
Posted by chris_torrence@NOSPAM on Wed, 13 Oct 2010 16:38:24 GMT
View Forum Message <> Reply to Message

Hi Paulo,

Great ideas! I just added #1 to the list (ha!) of potential features
for IDL 8.1.

Regarding #2, what if you could use additional indices to access array
elements within lists?

For example:

```
IDL> a = LIST(FINDGEN(10), BYTARR(5,3))
IDL> help, a[0]
<Expression>    FLOAT     = Array[10]
IDL> help, a[0,3]   ; currently throws an error in IDL8.0
<Expression>    FLOAT     =       3.00000
IDL> a[0,3] = !pi   ; currently throws an error in IDL8.0

IDL> help, a[1]
<Expression>    BYTE      = Array[5, 3]
IDL> help, a[1,4,2]   ; currently throws an error in IDL8.0
<Expression>    BYTE      =    0
IDL> a[1,4,2] = 255   ; currently throws an error in IDL8.0
```

So the first index would give the list element, and the remaining
indices would index into the array itself. Obviously you could only
have up to 7 dimensions in your contained array, but that probably
isn't a huge limitation.

Thoughts?

-Chris
ITTVIS

Subject: Re: Still missing features in IDL 8
Posted by penteado on Fri, 15 Oct 2010 00:26:55 GMT
View Forum Message <> Reply to Message

On Oct 13, 1:38 pm, Chris Torrence <gorth...@gmail.com> wrote:
> Regarding #2, what if you could use additional indices to access array
> elements within lists?
>
> For example:
>

> IDL> a = LIST(FINDGEN(10), BYTARR(5,3))
> IDL> help, a[0]
> <Expression> FLOAT = Array[10]
> IDL> help, a[0,3] ; currently throws an error in IDL8.0
> <Expression> FLOAT = 3.00000
> IDL> a[0,3] = !pi ; currently throws an error in IDL8.0
>
> IDL> help, a[1]
> <Expression> BYTE = Array[5, 3]
> IDL> help, a[1,4,2] ; currently throws an error in IDL8.0
> <Expression> BYTE = 0
> IDL> a[1,4,2] = 255 ; currently throws an error in IDL8.0
>
> So the first index would give the list element, and the remaining
> indices would index into the array itself. Obviously you could only
> have up to 7 dimensions in your contained array, but that probably
> isn't a huge limitation.
>
> Thoughts?

It took me a while to answer because I was considering the implications. It is a very clever idea. As I wrote above, I thought there would be no way to do this without changing the interpreter, and thus this would be a big problem to implement. But using additional indices means no new syntax, it is just a regular call to overloadbracketsrightside(). It seems it could even be implemented today, with IDL code, with a class inheriting from list. And this addition would be a very small break in compatibility: it would only break in the sense that what now throws an error (like a[0,3] above) would become valid. Small break, and dependent on a recent feature - a reason to implement this soon, while the likelihood of breaking code is still low.

I had been trying to figure out what the downsides could be. The clear limitation is the loss of one dimension to subscript the array, as you mentioned. How relevant this is in part depends on whether there are any discussions about changing the two 8D limits (the limit for arrays, which is probably deeply ingrained in many places in the interpreter and the API, and the limit for the overloadbrackets methods, which was introduced now, and can probably change much more easily).

My impression so far is that it would be good to implement this now. 8D arrays are probably very rare, and when one is dealing with 8 indices (or ranges of indices), things are probably somewhat unwieldy anyway, as in any situation where one is counting positional things (indices or positional arguments). The more likely situation for this problem to appear seems to be with a few nested lists, with 3-4D

arrays, where each list would consume one index, but again, it would be an unwieldy situation without the limitation anyway, and other data structures, or separating the code in more than one expression, might be more appropriate.

Maybe I will write a derived class now and start using it, to see if I encounter any unforeseen problems. Of course, for this to come with IDL's lists, it would be better to have it not written in IDL, as was the case with the implementation of lists and hashes. By the way, are the current lists and hashes like DLMs? Are they in the interpreter?

Subject: Re: Still missing features in IDL 8
Posted by penteado on Mon, 01 Nov 2010 00:00:15 GMT
View Forum Message <> Reply to Message

On Oct 13, 2:38 pm, Chris Torrence <gorth...@gmail.com> wrote:
> Regarding #2, what if you could use additional indices to access array
> elements within lists?
>
> For example:
>
> IDL> a = LIST(FINDGEN(10), BYTARR(5,3))
> IDL> help, a[0]
> <Expression> FLOAT = Array[10]
> IDL> help, a[0,3] ; currently throws an error in IDL8.0
> <Expression> FLOAT = 3.00000
> IDL> a[0,3] = !pi ; currently throws an error in IDL8.0
>
> IDL> help, a[1]
> <Expression> BYTE = Array[5, 3]
> IDL> help, a[1,4,2] ; currently throws an error in IDL8.0
> <Expression> BYTE = 0
> IDL> a[1,4,2] = 255 ; currently throws an error in IDL8.0
>
> So the first index would give the list element, and the remaining
> indices would index into the array itself. Obviously you could only
> have up to 7 dimensions in your contained array, but that probably
> isn't a huge limitation.

I was writing a class like that, inheriting from list, and that brought me a question: Should the extra dimension (of the list index) be on the left, as above, or on the right?

The notation (already valid for retrieving values) (a[1])[0] suggests that the array index should come on the left. However, writing a[1,0] suggests array dimensions, in which case the list index would make more sense on the right, as the list dimension is the slowest-varying

one.

Tough it would be a bit incoherent with the array dimension order, it seems to me that it is better to have the list index on the left. That way,

print,(a[1])[0] ;already valid

would be the same as

print,a[1,0]

instead of the more confusing

print,a[0,1]

Any thoughts on that?
