Subject: Re: Principle of Least Surprise
Posted by JD Smith on Wed, 10 Jan 2007 00:08:03 GMT
View Forum Message <> Reply to Message

On Tue, 09 Jan 2007 16:14:00 -0700, Jean H. wrote:

> ... and
> a[3:5].x=bindgen(3)
> works perfectly!

That doesn't prevent the first case I mentioned.

> It makes sense though... consider this case:
> IDL> b=replicate({x:intarr(10)},10)
> IDL> b[3].x = indgen(4)
>
> IDL must understand that the specified array is the values of X in b[3].
> If what you had written was working, IDL would not know if it is the
> values of X or if it is the 1st value of 4 entries of b...

Sure it would, easily. What is the dimension of b.x:

```
IDL> help,b.x
<Expression>    INT       = Array[10, 10]
IDL> help,b[3].x
<Expression>    INT       = Array[10]
```

I.e. the latter case has a 10 element array on its LHS, not a single index.

This is exactly analogous to:

```
IDL> b=intarr(10,10)
IDL> b[3,3]=indgen(4)
IDL> b[5,5]=indgen(4,2)
```

which IDL isn't confused about. I should equally be able to say:

```
IDL> b=replicate({x:intarr(10)},10)
IDL> b[3].x[2]=indgen(4,4)
```

and for it to do the right thing, without requiring index ranges to make it happen. Why should implicit structure arrays be less capable than their "normal" brethren?

IDL treats a LHS single index assigned to an array as a base from which to fill in the entire array.... except that is for structure dereferenced arrays. Certainly the form a[3:5] works, but it is inefficient for large assignments, since it unnecessarily generates an

index list [3,4,5] before assigning.

Here's how to see how this hurts:

```
IDL> a=make_array(/LONG,long(100.e6))
IDL> print,memory(/HIGHWATER)/1024./1024.
      382.465
```

382 MB for this array, OK. Now, let make another, one shorter:

```
IDL> b=make_array(/LONG,long(100.e6)-1L)
IDL> print,memory(/HIGHWATER)/1024./1024.
      763.934
```

Now assign it using the "base index" trick:

```
IDL> a[1]=b
IDL> print,memory(/HIGHWATER)/1024./1024.
      763.934
```

No additional memory used for the assignment from b to a, as appropriate. Now, however, use an index range for the assignment, as you are forced to do with structures:

```
IDL> a[1L:long(100.e6)-1]=b
IDL> print,memory(/HIGHWATER)/1024./1024.
      1145.40
```

Whoops, we just used 382MB for the useless exercise of creating a large index list [1,2,...,99999999] to consult while making the assignment. Not only does this waste memory, it can really slow things down.

JD
