Subject: Still missing features in IDL 8
Posted by penteado on Tue, 12 Oct 2010 21:35:02 GMT
View Forum Message <> Reply to Message

I was showing IDL's lists and hashes to a friend, who immediately noticed two important missing points:

1) A new method (or a new keyword, like, /pointer, to the existing toarray()), that would create from an arbitrary list an array of pointers, where each element in the array is a pointer to the corresponding element of the list. That is, something equivalent to

```
function topointerarray,list
ret=ptrarr(n_elements(list))
foreach el,list,i do ret[i]=ptr_new(el)
return,ret
end
```

And a new keyword to list() and list::add (such as /dereference) that would do the opposite: given a pointer array, would add to the list the variables pointed to by each element of the pointer array (or ! null in case the element is a null pointer or a pointer to an undefined variable).

Also, in some cases it would be desirable for list::toarray() and hash::tostruct() to have a no_copy keyword, that would empty the list/ hash when making the array/structure, instead of making a copy. This is already present in list() and list::add, but is also missing in hash().

Why is (1) so needed? Besides better handling those conversions (particularly when interfacing with old code that returns/expects pointer arrays), for getting around the lack of:

2) This is not a simple additional feature as (1), but rather a change to the language: Make a way for a list/hash element not be an expression. One example of the problem is this error:

```
IDL> a=list()
IDL> a.add,bindgen(3)
IDL> a.add,-dindgen(2)
IDL> print,a
  0   1   2
     -0.0000000      -1.0000000
IDL> print,(a[1])[0]
     -0.0000000
IDL> (a[1])[0]=1.0
% Expression must be named variable in this context: <DOUBLE
```

Array[2]>.
% Execution halted at: $MAIN$

This error happend because (a[1]) is an expression:

```
IDL> help,(a[1])
<Expression>    DOUBLE    = Array[2]
```

And I can see why it is, as I guess it is the result of a call to list::overloadbracketsrightside(). But this makes it very inconvenient to change pieces of list/hash elements, like in the line that caused the error above. It would be necessary to retrieve the element, assigning it to some variable, change that variable, then put it back in the list/hash. Very awkward, and unnecessary if a pointer array is used instead of the list:

```
IDL> a=ptrarr(2)
IDL> a[0]=ptr_new(bindgen(3))
IDL> a[1]=ptr_new(-dindgen(2))
IDL> print,(*a[1])[0]
      -0.0000000
IDL> (*a[1])[0]=1.0
IDL> print,(*a[1])[0]
       1.0000000
```

Which obviously works because (*a[1]) is a variable:

```
IDL> help,(*a[1])
<PtrHeapVar2>   DOUBLE    = Array[2]
```

I realize that this is a change to the language, and somewhat tricky to implement. But it is probably easier than was the addition of operator overloading.
