Subject: Re: Storing !NULL in struct
Posted by penteado on Fri, 15 Mar 2013 23:14:14 GMT
View Forum Message <> Reply to Message

I think a lot about this subject.

The problem with doing

```
c[0].t=1
```

is in the way the overloaded brackets are implemented. When c[0] gets the ".t", there is a function call to list's _overloadBracketsRightSide, which just returns a value (the structure). Since it is a value (not a variable), no values can be assigned to it. To put this another way, the line above is the same as trying to do

```
a=0
a+9=5
```

One way to sort of get around this would be to put in the list pointers to the structures:

```
IDL> c=list(ptr_new({t:0}))
IDL> print,(*c[0]).t
      0
IDL> (*c[0]).t=9
IDL> print,(*c[0]).t
      9
```

Or one could make a derived list class that stored the elements by pointers, and returned the pointers, so that one would not need to keep doing ptr_new() everytime something is added to the list.

It could even return either the value or a pointer to it, depending on how the brackets are used: If it gets an integer index, it returns the element, as usual; if it gets a floating point index, it returns the pointer to the element. Then it could be used like:

```
IDL> c=listbypointers({t:0})
IDL> print,c[0].t
      0
IDL> print,(*c[0.]).t
      0
IDL> (*c[0.]).t=9
IDL> print,c[0].t
      9
```

To make things like

c[0].t=1

valid, the IDL interpreter would have to change, so that when an
object with brackets shows up in the left side of the assignment but
it is qualified (with the .t, in this case) the object's
_overloadBracketsRightSide would be called, to return a variable, then
whatever that assignment does to the variable is performed, then at
the end the variable is passed back to the object's
_overloadBracketsLeftSide.

On Mar 15, 11:02 am, fawltylangu...@gmail.com wrote:

```
> On Friday, March 15, 2013 9:10:45 AM UTC+1, Tom Grydeland wrote:
>
>> IDL> c = List({ t: 0})
>> IDL> print, c[0].t
>>           0
>> IDL> c[0].t = 1
>> % Attempt to store into an expression: Structure reference.
>> % Execution halted at: $MAIN$
>
> This is a bug IMHO. This construct should work as a structure array works (LIST is a pointer array in disguise):
>
> IDL> a=replicate({t:0}, 1)
> IDL> a[0].t=1
> IDL> print, a[0].t
>        1
>
> regards,
> Lajos
```