Subject: Re: Modifying Arrays and Structures in HASH's (hint: you can't)
Posted by penteado on Thu, 22 Mar 2012 23:24:08 GMT
View Forum Message <> Reply to Message

It is not an intrinsic limitation of their objectyness. It is all about how the overloadbracketsleftside method is defined. Once someone pointed out that problem to me (in the case of arrays), I had several discussions with the developers (some in this newsgroup), and eventually the methods were changed (in 8.1, I think) so that through multiple indices it can be done with arrays:

```
IDL> h=hash('a',[0,3,9])
IDL> print,h['a']
       0       3       9
IDL> print,h['a',1]
       3
IDL> h['a',1]=-1
IDL> print,h['a',1]
      -1
```

The problems shows up in structures and array indices if they are used in the way written above:

```
IDL> print,(h['a'])[1]
      -1
IDL> (h['a'])[1]=99
% Expression must be named variable in this context: <INT       Array[3]>.
% Execution halted at: $MAIN$
```

And that is all fault of the parser's weird use of values instead of references on qualified names. Which, for that reason, and for breaking the least surprise principle, should be changed in a future version (a compile_opt, or a new file extension, to keep compatibility).

But I think that the multiple indices for arrays approach could easily be applied to structures, just making a small change to the overloadbracketsleftside method. I may write a small derived class for that purpose. In that way, it would be possible to do

```
h['a','b']=2
```

On Thursday, March 22, 2012 6:58:19 PM UTC-3, JDS wrote:
> HASH's are nice in that they can contain any variable type, and make iterating over deeply nested data structure more humane. On the other hand, their limitations as separate standalone objects, and not as a deeper feature of the language, are very apparent, for example, when you'd like to alter something inside of them:
>
> IDL> h=hash('a',{b:1,c:2})
> IDL> print,h['a'].b
> 1
> IDL> h['a'].b=2
> % Attempt to store into an expression: Structure reference.

> % Execution halted at: $MAIN$ 84
>
> The same happens for arrays stored in hashes. You're required to copy the entire array or structure out, modify it, then copy it back into the hash (remind anyone of the old days and widget state?). If it were possible for HASH dereferencing to return true IDL variable references, they would be far more useful.
>
> Any workarounds to this "write-only" HASH behavior (other than, say, "use a pointer")?
>
> JD

Subject: Re: Modifying Arrays and Structures in HASH's (hint: you can't)
Posted by JDS on Fri, 23 Mar 2012 23:07:00 GMT
View Forum Message <> Reply to Message

> The problems shows up in structures and array indices if they are used in the way written above:
>
> IDL> print,(h['a'])[1]
> -1
> IDL> (h['a'])[1]=99
> % Expression must be named variable in this context: <INT Array[3]>.
> % Execution halted at: $MAIN$
>
> And that is all fault of the parser's weird use of values instead of references on qualified names. Which, for that reason, and for breaking the least surprise principle, should be changed in a future version (a compile_opt, or a new file extension, to keep compatibility).
>
> But I think that the multiple indices for arrays approach could easily be applied to structures, just making a small change to the overloadbracketsleftside method. I may write a small derived class for that purpose. In that way, it would be possible to do
>
> h['a','b']=2

Interesting. I wonder if this would this be fully as flexible as the structure dereference, which can retrieve and assign arrays:

```
IDL> a=replicate({b:randomu(sd)},100)
IDL> help,a.b
<Expression>    FLOAT     = Array[100]
IDL> a.b=findgen(100)
```

It would need to do so for the POLS. It certainly loses points for lack of semantic consistency. And it creates another more extreme category of, e.g. the "structure member variables cannot be passed by reference" rule.

Thanks,

JD