
Subject: Re: confusion around a pointer to an array of structures
Posted by [JD Smith](#) on Tue, 09 Aug 2005 21:14:28 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 09 Aug 2005 16:07:14 -0400, Paul Van Delst wrote:

```
> Edd wrote:
>> Henry <henrygroee@yahoo.com> wrote:
>>
>>> But, I can see no other way around this, aside from redesigning how I'm
>>> storing the information. Does anyone see how to do the equivalent of
>>> "(*a)[0].x = 99d" in the above example? (without the awkward three
>>> line dumb hack I've shown)
>>> I'm sure I'm just not seeing something simple....
>>
>>
>> Is there anything incorrect about saying (*a)[0].x=99d ?
>>
>
> I tried the OP's method and got his error,
>
> IDL> ((*a)[0]).x = 99d
> % Attempt to store into an expression: Structure reference.
> % Execution halted at: $MAIN$
>
> and thought that the "correct" syntax would be something like ((*a).x)[0]
>
> This is what I get,
>
> IDL> a = ptr_new( replicate( {x:0d}, 5 ) )
> IDL> (*a).x = dindgen(5)
> IDL> help, ((*a).x)[0]
> <Expression>  DOUBLE   =      0.0000000
> IDL> ((*a).x)[0] = 99d
> % Internal error: The Interpreter stack is not empty on exit.
> IDL> help, ((*a).x)[0]
> <Expression>  DOUBLE   =      99.0000000
>
> So that works too..... sort of.
>
> Veird.
```

Actually, it's pretty much as expected. What you wanted is:

```
IDL> (*a)[0].x=99D
```

Doing it your way first creates a temporary array ((*a).x) and then indexes and assigns a member, which is slow, and leads to the reported

error. The original error was even more severe: you can't (usually) include temporary expressions on the LHS of the assignment.

Since '[' and '.' are at the same level of precedence, all you need to remember is to use parentheses to group '*' with all the pointers in the expression. No other parentheses are needed, and in fact using any other parentheses will likely create temporary variables, which is not usually what you want (and can lead to errors of the type mentioned).

Imagine a bizzarely deeply nested data structure like this:

```
IDL> a=ptr_new(replicate({b:ptr_new([ptr_new(replicate(1,5)),  
    ptr_new(replicate({c:ptr_new(14.}),3))}]},  
    4))
```

Suppose we want the "c" field of the last of that final array of structures. How to approach this? One way is first to pretend there is no such thing as operator precedence, and just write it out without thinking:

```
val=***a[1].b[1][2].c
```

This of course, won't work. Now go through and group, using (), all pointers together with their leading dereference operator '*', starting with the innermost. There are a total of 4 pointers we need to consider. Here is the result:

```
val=*((*(*a)[1].b)[1])[2].c
```

See how 'a' is a pointer, so '(*a)' is the group, '(*a)[1].b' is a pointer so '(*(*a)[1].b)' is the group, etc.? You can leave off any "outer" parentheses that are not followed by anything (and probably should if you are assigning to the element, just to get in the habit).

JD
