Subject: Re: nested structures
Posted by Phillip Bitzer on Mon, 27 May 2013 15:03:01 GMT
View Forum Message <> Reply to Message

You can certainly do what you're after. In fact, I do this sort of thing when building arrays of radar data, which may have different lengths, sizes, etc.

First, some basic pointer stuff:

Consider:

```
IDL> s1 = {tag1:0L, tag2:PTR_NEW(/ALLOCATE)}
```

Then,

```
IDL> help, s1
** Structure <314b91d8>, 2 tags, length=8, data length=8, refs=1:
   TAG1            LONG                 0
   TAG2            POINTER   <PtrHeapVar14>
```

So, we see tag2 is a pointer. Fine, let's assign the pointer to a (new) structure:

```
IDL> *s1.tag2 = {ntag1:0L, nTag2:0L}
```

Okey doke. So, s1.tag2 is the pointer, and when we dereference this:

```
IDL> help, *s1.tag2
** Structure <1dc84338>, 2 tags, length=8, data length=8, refs=1:
   NTAG1           LONG                 0
   NTAG2           LONG                 0
```

we see our (new) structure.

What about getting to one of these tags? Notice this doesn't work:

```
IDL> help, *s1.tag2.ntag2
% Expression must be a structure in this context: <No name>.
% Execution halted at: $MAIN$
```

But this does:

```
IDL> help, (*s1.tag2).ntag2
<Expression>    LONG      =           0
```

Remember, *s1.tag2 is the pointer, and that's what what we want to dereference. That's why the parentheses are where they are.

Arrays of structures with pointers can be a little more tricky, because you'll be throwing brackets in there too. Just keep in mind where the pointer is.

Further, you'll want to take a look a this for the initialization:

http://www.idlcoyote.com/code_tips/structptrinit.html