
Subject: structures, arrays of pointers and assignment
Posted by [Ingo von Borstel](#) on Wed, 02 May 2007 13:22:13 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi there,

with one of my routines I found an issue that I cannot explain myself.
Maybe someone might put me on the right track as to where things go
wrong. Probably it's an issue with passing variables by reference and by
value, but I don't quite understand the difference in the two things I
do. Assume the two variables:

```
IDL> help, sep213.trackvalues
<Expression>  STRUCT  = -> TRACKVALUES Array[1]
IDL> help, myvalues
MYVALUES      STRUCT  = -> TRACKVALUES Array[1]
```

where trackvalues is a struct:

```
trackvalues = {trackvalues, $
               value:PTRARR(4), $
               used: BYTARR(4), $
               description:STRARR(4), $
               timebase:" $
               }
```

and sep213 is a struct, too, that contains trackvalues as one item.

I have written a routine to manipulate the contents of trackvalues:

```
PRO do_something, in_trackvalues
; create just another struct, details of it unimportant
newentry = {values:dblarr(300), used:bytarr(300), descr:"}
; create a pointer to this struct
ptrnew = ptr_new(newentry)
; ensure to get a valid index in the array in_trackvalues.value so that
I can assign the above created pointer to this index
wherefree = <some code to ensure I get a valid index so that the next
line is a valid assignment>
in_trackvalues.value[wherefree] = ptrnew
END
```

Now I call this routine from the IDL prompt:

```
IDL>do_something, sep213.trackvalues
or
IDL>do_something, myvalues
```

and get different results:

```
IDL> print, sep213.trackvalues.value[0:3]
```

```
<PtrHeapVar77176><PtrHeapVar77177><PtrHeapVar77178><NullPointer >  
IDL> print, myvalues.value[0:3]  
<PtrHeapVar77176><PtrHeapVar77177><PtrHeapVar77178><PtrHeapVar77973 >
```

The latter case is what I expect as I create an additional, valid entry in the ptrarr array. Why is this entry forgotten in the first case? I checked whether the routine works in the first case. It does up to the last line exactly the same, only there is no modified data returned to the calling routine.

Any clue as to why? My guess it's to do with passing elements of a struct prohibits its modification by a subroutine. If so, is there a way to circumvent this behaviour?

Best regards,
Ingo

--

Ingo von Borstel <newsgroups@planetmaker.de>
Public Key: <http://www.planetmaker.de/ingo.asc>

If you need an urgent reply, replace newsgroups by vgap.
