Subject: Re: structures, arrays of pointers and assignment
Posted by Ingo von Borstel on Wed, 02 May 2007 15:50:11 GMT
View Forum Message <> Reply to Message

Hi,

>> Any clue as to why? My guess it's to do with passing elements of a
>> struct prohibits its modification by a subroutine.
>
> I think you already know the answer. When you pass a de-referenced
> structure variable, you pass it by value, and when you pass
> the structure itself, you pass it by reference. When something
> is passed by value, a copy of the thing is made and worked on.

I feared this to be the case. Thanks for sort-of confirming my guess.

>> If so, is there a way to circumvent this behaviour?

The point is the following:

I have written a bunch of routines which work well on a single set of data; their result are a number of different structs. Now I want to compare them, plot one vs. the other, filter them propperly etc. That's fine as long as I do it with one set of data at a time. But I want to do that not only for one set of data, but I want to compare also different data sets with eachother.
Then the amount of variables starts to get long... so creating a struct that contains all essential data for a single set was my idea to keep order. Any other idea how to keep together those things that belong together (e.g.makes the use of SAVE much easier, if I want to preserve some results for future reference).

> If you mean circumvent this behavior within the confines
> of pass by value or pass by reference, the answer is no.
> But this is software. Anything is possible! You could,
> for example, have your program return a value that you then
> assigned to the proper field in the structure. But the

Hm... Let's see. Let's create a wrapper function of the same name, and with the same arguments:
var.struct = do_something(var.struct,...)
That will work. Probably it'll work in most other cases as I mostly only modify one variable.

But now I wonder why deleting an entry works:

```
PRO delete_entry, trackvalues, number
 IF NOT PTR_VALID(trackvalues.value[number]) THEN BEGIN
```

```
    MESSAGE, "Trackvalues number is not a valid record. Nothing done",
/CONTINUE
    RETURN
 ENDIF
 PTR_FREE, trackvalues.value[number]
 trackvalues.used[number] = 0
 trackvalues.description[number] = ''
END
```

IDL>delete_entry, sep213.trackvalues, 3
will result in the deletion of the 3rd (or 4th, depends upon where you start counting) entry... Why the hell does this work then? Shouldn't work, if this was also passed by value...

> easiest thing to do, of course, since you have the
> structure hanging around or you wouldn't be able to pass
> a *field* of the structure, is to just pass the whole darn
> thing. That seems simple enough to me.

Not so easy. As mentioned above, all the routines are written such that they accept variable types as used for dealing with a single set of data. Wouldn't want to change their basic structure.

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.