Subject: Re: Structures and arrays of structures
Posted by penteado on Fri, 28 Oct 2016 15:43:00 GMT
View Forum Message <> Reply to Message

On Friday, October 28, 2016 at 6:54:42 AM UTC-7, rj...@le.ac.uk wrote:
> That's what I tried initially but didn't seem to work. It came up with an error that structure_name__Define was not defined.

That sounds like you were trying to use a named structure, which had not been defined. Note that structures can be anonymous, as is the case of all the structures that showed up above, or named. I am not talking about the names of the variables holding the structures, I am talking about the names of the structure types (think of it like a type name such as float, integer, complex, etc).

This is a way to obtain that error:

```
IDL> a=replicate({structure_name},10)
% Attempt to call undefined procedure: 'STRUCTURE_NAME__DEFINE'.
% Execution halted at: $MAIN$
```

This is because there was not a type structure_named already defined, so IDL looked for a procedure called struct_name__define.pro which would define it, but it also did not find such a procedure. Note that doing

```
IDL> structure_name={a:0,b:1.0}
IDL> a=replicate({structure_name},10)
% Attempt to call undefined procedure: 'STRUCTURE_NAME__DEFINE'.
% Execution halted at: $MAIN$
```

Results in the same error. That is because the first line is just creating an anonymous structure, and assigning it to a variable that happens to be called structure_name. The structure type is not named.

This works:

```
IDL> structure_name={a:0,b:1.0}
IDL> a=replicate(structure_name,10)
IDL> help,a
A               STRUCT    = -> <Anonymous> Array[10]
```

The call to replicate is asking for a structure array that has 10 copies of the structure in the variable structure_name. Note there were no {} around structure_name in the call to replicate.

If what you wanted really was a named structure, it would be defined as:

```
IDL> c={structure_name,a:0,b:1.0}
IDL> help,c
** Structure STRUCTURE_NAME, 2 tags, length=8, data length=6:
   A               INT            0
   B               FLOAT          1.00000
IDL> a=replicate({structure_name},10)
IDL> help,a
A               STRUCT    = -> STRUCTURE_NAME Array[10]
```

Note that both a and c are not anonymous, they are of the type structure_name. The same result would be obtained with

```
IDL> a=replicate(c,10)
IDL> help,a
A               STRUCT    = -> STRUCTURE_NAME Array[10]
```

That said, you cannot use replicate with your global_struct to create an array that can also hold your aus_struct, because the types global_struct and aus_struct are not compatible: in global_struct the fields have 2392 elements, in aus_struct they have 2341 elements:

```
IDL> s1={a:fltarr(2392),b:fltarr(2392)}
IDL> s2={a:fltarr(2341),b:fltarr(2341)}
IDL> s3=[s1,s2]
% Conflicting data structures: S1,S2.
% Execution halted at: $MAIN$
IDL> s3=replicate(s1,2)
IDL> s3[0]=s1
IDL> s3[1]=s2
% Conflicting data structures: S3,S2.
% Execution halted at: $MAIN$
```

But they can be nested structures, as discussed above. Or, since you have strings to identify them, I would find it more neat to use a hash (or maybe an orderedhash, if it is IDL 8.3 or newer):

```
IDL> ss=['s1','s2'] ;names of the variables to put in the hash
IDL> h=hash()
IDL> for i=0,n_elements(ss)-1 do h[ss[i]]=scope_varfetch(ss[i])
IDL> help,h
H               HASH  <ID=155135  NELEMENTS=2>
IDL> print,h.keys()
s1
s2
IDL> help,h['s2']
** Structure <1abaed30>, 2 tags, length=18728, data length=18728, refs=3:
   A               FLOAT     Array[2341]
   B               FLOAT     Array[2341]
```