Subject: Re: Q: Array of Structures
Posted by J.D. Smith on Fri, 12 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Saeid.Zoonematkermani@sunysb.edu writes:
>
>> Thanks for the response. I think there is more to this problem than is
>> apparent at first. I believe that my structures are created identically.
>> It is the output of a complicated procedure. The structure is defined
>> inside the procedure and they should be the same each time. And just to
>> check this, I ran your example from above and here is the result:
>>
>> IDL> a = { data:FltArr(10), name:'First Structure' }
>> IDL> b = { data:FltArr(10), name:'Second Structure' }
>> IDL> c = [a, b]
>> % Conflicting data structures: B,concatenation.
>> % Execution halted at: $MAIN$
>>
>> Rather curious. I wonder if this is a MacOS version problem!
>
> Oh, oh. Another one of those days. I *tested* this. I have the
> file to prove it! But you are right, I can't reproduce it. :-(
>
> I guess string lengths in structures have to be exactly the
> same, too. Which surprises me and goes contrary to what happens,
> for example, when you replicate named structures with the
> REPLICATE command.
>
> But, look. I just tried this:
>
>   a = { data:FltArr(10), name:'' }
>   b = { data:FltArr(10), name:'' }
>   c = [a, b]
>   % Conflicting data structures: B,concatenation.
>   % Execution halted at: $MAIN$
>
> But now, I try this:
>
>   a = { data:FltArr(10), name:'' }
>   b = a
>   c = [a, b]
>
> No problem!
>
> Very, very weird. I'll see if RSI can explain this. :-)
>

> Cheers,
>
> David

As far as I know, you cannot, *in general*, concatenate anonymous structures (I mean into a list, not into a larger structure which is their superset, which can be accomplished with create_struct). I have a possible guess as to what may be happening.

For concatenation of structures, IDL probably tests the *types* of structures to see whether the concatenation is allowed, rather than the fields and field types contained in those structures (and much less the actual data members).

E.g.

IDL> a={S1, name:'name'}
IDL> b={S1, name:'alongname'}
IDL> c=[a,b]

...no problem, no matter what data lengths are involved, as long as those types are the same. On the other hand, when you create an anonymous structure, IDL likely gives it some random internal structure type -- not for public consumption, but merely as a convenience. For replicate, as for your second example above, the exact same structure (of some mysterious "internal anonymous type") is being *copied*, and no new "internal anonymous type" is created. This type of anonymous structure concatenation succeeds, since the structure types (albeit unknown to us) are the same, being mere copies of a original anonymous structure. On the other hand, two successive anonymous structure creations yield two *different* internal types, which, according to the rules, cannot be concatenated.

Why do anonymous structures need this internal type? Likely because if they didn't have it, replicate and other such functions couldn't work with the same code for both named and anonymous structures. Anonymous structures clearly cannot all have the *same* internal type, since then something like:

IDL> a=[{test:5},{test:[1,2,3,4,5]}]

would be allowed. The only options are either to treat anonymous structures specially, or to use fields and data member types (not structure type) to determine when structure concatenation is allowed. The former might be a pain in the neck. The latter would significantly slow down structure operations, increasing the burden of structure type-checking from one time only (at the time of creation) to possibly very many times.

There may not be any other compelling reasons, and a new way of dealing with anonymous structures might even get implemented, if we make enough noise. As a reminder, this may be a totally uninformed guess of the way IDL works (seems

plausible to me, though :)).

For a solution to your problem, just use a named structure when you need dynamically modified lists of them. They are more inconvenient, but also more reliable in these situations. If you do object oriented programming, an easy place to define them is in the same class__define procedure used to prototype your object class.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Q: Array of Structures
Posted by davidf on Fri, 12 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Saeid.Zoonematkermani@sunysb.edu writes:

> Thanks for the response. I think there is more to this problem than is
> apparent at first. I believe that my structures are created identically.
> It is the output of a complicated procedure. The structure is defined
> inside the procedure and they should be the same each time. And just to
> check this, I ran your example from above and here is the result:
>
> IDL> a = { data:FltArr(10), name:'First Structure' }
> IDL> b = { data:FltArr(10), name:'Second Structure' }
> IDL> c = [a, b]
> % Conflicting data structures: B,concatenation.
> % Execution halted at: $MAIN$
>
> Rather curious. I wonder if this is a MacOS version problem!

Oh, oh. Another one of those days. I *tested* this. I have the file to prove it! But you are right, I can't reproduce it. :-(

I guess string lengths in structures have to be exactly the same, too. Which surprises me and goes contrary to what happens, for example, when you replicate named structures with the REPLICATE command.

But, look. I just tried this:

```
a = { data:FltArr(10), name:'' }
b = { data:FltArr(10), name:'' }
```

```
c = [a, b]
% Conflicting data structures: B,concatenation.
% Execution halted at:  $MAIN$
```

But now, I try this:

```
a = { data:FltArr(10), name:'' }
b = a
c = [a, b]
```

No problem!

Very, very weird. I'll see if RSI can explain this. :-)

Cheers,

David

Subject: Re: Q: Array of Structures
Posted by Saeid.Zoonematkermani on Fri, 12 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

> An array must be a collection of identical "things". For
> example, if your anonymous structures were identical you
> would have no problem. (By "identical" I mean that they
> have the same fields and that those fields are defined
> in exactly the same way.) For example:
>
> a = { data:FltArr(10), name:'First Structure' }
> b = { data:FltArr(10), name:'Second Structure' }
> c = [a, b]
>

Hi David,

Thanks for the response. I think there is more to this problem than is apparent at first. I believe that my structures are created identically. It is the output of a complicated procedure. The structure is defined inside the procedure and they should be the same each time. And just to check this, I ran your example from above and here is the result:

```
IDL> a = { data:FltArr(10), name:'First Structure' }
IDL> b = { data:FltArr(10), name:'Second Structure' }
IDL> c = [a, b]
% Conflicting data structures: B,concatenation.
% Execution halted at:  $MAIN$
```

Rather curious. I wonder if this is a MacOS version problem!

Cheers,

- Saeid

--
Saeid Zoonematkermani
Dept. of Physics & Astronomy
State University of New York
Stony Brook, NY 11794-3800

EMail: Saeid.Zoonematkermani@sunysb.edu
Voice: (+1) (516) 632-8237 Fax: (+1) (516) 632-8742

Subject: Re: Q: Array of Structures
Posted by davidf on Fri, 12 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Saeid Zoonematkermani (Saeid.Zoonematkermani@sunysb.edu) writes:

> The output of one of my applications is an anonymous structure. After a
> long session, I would like to create an array ofthese structures. I have
> tried the following unsuccessfully.
>
> IDL> tmp = [str1, str2]
> % Conflicting data structures: STR2,concatenation.
> % Execution halted at: $MAIN$
>
> I have also tried:
>
> IDL> tmp=replicate(str1,2)
> IDL> tmp(1)=str2
> % Conflicting data structures: STR2,TMP.
> % Execution halted at: $MAIN$
>
> I have string tags in the structure and my first thought was that the
> unequal string length from one structure to another is causing the
> problem. This does not seem to be the case since I picked two structures
> with identical lengths and the same error message popped up.
>
> Since these structures are created inside the same application, all the
> tags are the same and I had hoped there would be no problem with creating
> the arrays after the processing. My work around has been to write a simple
> procedure that assigns the tags individually for each array element.
>

> I am sure I am missing some thing very simple but with the world cup going
> on, my brain seems to be on vacation. Any help would be greatly
> appreciated. I am using IDL Version 5.1 (MacOS PowerMac). Thanks in
> advance,

It is odd how the universe works. I've never been asked this question before and this week I encounter it twice! Weird. :-)

An array must be a collection of identical "things". For example, if your anonymous structures were identical you would have no problem. (By "identical" I mean that they have the same fields and that those fields are defined in exactly the same way.) For example:

```
a = { data:FltArr(10), name:'First Structure' }
b = { data:FltArr(10), name:'Second Structure' }
c = [a, b]
```

Notice that this works even though the name fields have strings in them of different lengths. The length of a string name is not important. It is the definition of the field as a string that is important.

Where you have problems, and where you will get the error you report above, is when the fields have different definitions. For example, this doesn't work:

```
a = { data:FltArr(10), name:'First Structure' }
b = { data:FltArr(15), name:'Second Structure' }
c = [a, b]
```

It doesn't work because even though the data field in B is defined as a float array, it has 15 elements in it as opposed to the 10 in the data field of A. Thus, these structures are not "identical".

A way around this, of course, is to create an array of pointers to structures that are defined differently. For example:

```
a = { data:FltArr(10), name:'First Structure' }
b = { data:FltArr(15), name:'Second Structure' }
c = [Ptr_New(a), Ptr_New(b)]
```

But now you have to do structure de-referencing when you refer to the structures:

```
Plot, (*c[1]).data
```

You also have to be sure to destroy your pointers when
you are finished with them. Otherwise, they will persist
in memory.

   For j=0,N_Elements(c)-1 Do Ptr_Free, c[j]

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Q: Array of Structures
Posted by korpela on Sun, 14 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.feb2857db08f77a989681@news.frii.com>,
David Fanning <davidf@dfanning.com> wrote:
> But, look. I just tried this:
>
>   a = { data:FltArr(10), name:'' }
>   b = { data:FltArr(10), name:'' }
>   c = [a, b]
>   % Conflicting data structures: B,concatenation.
>   % Execution halted at: $MAIN$
>
> But now, I try this:
>
>   a = { data:FltArr(10), name:'' }
>   b = a
>   c = [a, b]
>
> No problem!

I think that the problems it that the first example creates anonymous
structures for each variable. Because the structures won't have the
same tag (anonymous structures are still tagged invisibly), they are
treated as different structure types. The problem goes away for tagged
 structures. The following works just fine.

a = {StruTag, data:FltArr(10), name:'' }

b = {StruTag, data:FltArr(10), name:'' }
c = [a, b]

In the second example, the assignment gives b the same structure tag
as a, so the concatenation works just fine.

Personally, I'd like to see IDL check for identical anonymous structures
and set the structure tags appropriately.

Eric
--
Eric Korpela | An object at rest can never be
korpela@ssl.berkeley.edu | stopped.
Click for home page.

Subject: Re: Q: Array of Structures
Posted by Vap User on Wed, 17 Jun 1998 07:00:00 GMT
View Forum Message <> Reply to Message

korpela@islay.ssl.berkeley.edu (Eric J. Korpela) writes:

One may also use anonymous structures, as Dave did in after 'but
now, I try this'. I concatenate anonymous structures frequently, so
when I saw Dave' assertion that you can't do it, I shook my head in
wonder. However, I found out, only too quickly, that all of you are
right. So, how do I do it? The secret is that at the beginning of
whatever program I'm doing the concatenation I define the anonymous
structure and a blank copy of it. I keep filling the blank copy with
data and then concatenating it to create the larger array of
structures. Like this.

Start program

```
 a={some anonymous structure definition} & b=a
 start your loop construct here
do your processing
fill in b structure
      if first iteration
        output_array_of_structures=b
else
  output_array_of_structures = [output_array_of_structures,b]
 end loop
end program
```

No conflicts that I can see.

You could use named structures, with the additional hassle, perhaps small in your case, of having to exit IDL everytime you make a change to the structure.

I suspect it has to do with an underlying 'naming' convention for anonymous structures, (do a help,/structure. See the '** Structure <long hex number>' The long hex number is the 'type' I believe) so that deep down in IDLs code there is some comparison of the structure 'types' going on, not comparison of the actual contents of the structure.

I would point out that if you know how big your output will be, you should just go ahead and create the array in the beginning of the processing. Concatenation is a *slow* process. Alternately, you can create the largest array you think you'll need then enlarge it if your processing produces more output than expected.

>
> In article <MPG.feb2857db08f77a989681@news.frii.com>,
> David Fanning <davidf@dfanning.com> wrote:
>> But, look. I just tried this:
>>
>> a = { data:FltArr(10), name:'' }
>> b = { data:FltArr(10), name:'' }
>> c = [a, b]
>> % Conflicting data structures: B,concatenation.
>> % Execution halted at: $MAIN$
>>
>> But now, I try this:
>>
>> a = { data:FltArr(10), name:'' }
>> b = a
>> c = [a, b]
>>
>> No problem!
>
> I think that the problems it that the first example creates anonymous
> structures for each variable. Because the structures won't have the
> same tag (anonymous structures are still tagged invisibly), they are
> treated as different structure types. The problem goes away for tagged
> structures. The following works just fine.
>
> a = {StruTag, data:FltArr(10), name:'' }
> b = {StruTag, data:FltArr(10), name:'' }
> c = [a, b]
>
> In the second example, the assignment gives b the same structure tag

> as a, so the concatenation works just fine.
>
> Personally, I'd like to see IDL check for identical anonymous structures
> and set the structure tags appropriately.
>
> Eric
> --
> Eric Korpela | An object at rest can never be
> korpela@ssl.berkeley.edu | stopped.
> Click for home page.

--
I don't speak for JPL, it doesn't speak for me.
Well, not all the time, at least.
William Daffer <vapuser@haifung.jpl.nasa.gov>
