Subject: Re: memory allocation for structure arrays
Posted by Ian Sprod on Fri, 01 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Eric and Peter,

Thanks for your detailed replies. I have plenty to work on now!

I should have been a little more specific - IDL fails with an out-of-memory error during the REPLICATE call. Not the READU call. So the lazy-malloc makes sense to me.

Just to recap I have a structure describing a 44 byte record. Some of the fields are strings, but I initialize the structure with strings the correct length. With smaller files I can read in the data without a problem. So I don't think strings are the problem (but it could be). Perhaps IDL is allocating a large amount of memory to each string field, just in case in needs to grow? This is not allowed with structures according to the manual, as once a structure's fields are defined they cannot be changed.

On a related point, when you do a :

help,data_structure,/structure

There is a field called "length". Is this the amount of memory in bytes allocated to the structure? I can't find this field defined in the hardcopy or online documentation.

Lets assume it is the structure memory allocation in bytes. Then we get odd results!

```
test = {byte1:0b,byte2:0b,byte3:0b}
help,test,/structure
** Structure <819499c>, 3 tags, length=3, refs=1:
   BYTE1           BYTE       0
   BYTE2           BYTE       0
   BYTE3           BYTE       0
```

>> Note the length is 3 - this looks OK for 3 bytes. What about for long-integers?

```
test = {long1:0l,long2:0l,long3:0l}
help,test,/structure
** Structure <8194ab4>, 3 tags, length=12, refs=1:
```

>> OK - this looks like what we would expect too 4 bytes per long integer. BUT try this one :

```
test = {long1:0l,byte2:0b,byte3:0b}
```

help,test,/structure
** Structure <8194e5c>, 3 tags, length=8, refs=1:

>> The length is 8 - I would expect 6 (4 + 1 + 1).

>> Now for strings it gets even worse :

test = {str1:'a'}
help,test,/structure
** Structure <8194ba4>, 1 tags, length=8, refs=1:

test = {str1:'aa',str2:'b'}
help,test,/structure
** Structure <8194e0c>, 2 tags, length=16, refs=1:

>> IDL seems to allocate 8 bytes per string field - no matter how long the string is initialized. I guess this is the "padding byte" problem Peter talks about in his email.

So for my supposedly 44 byte-long record I get a "length" of 104. If this is truly the length in bytes then REPLICATE would be looking for 104/44 or ~2.5 times more memory than I was expecting. This is pretty huge memory allocation when the file in question in 40Mb to begin with!

But then again maybe length is NOT the memory allocation in bytes :-)

I am still investigating this problem - please let me know if you can shed any more light on this. I'm going to take Peters suggestion and recode with bytarr instead of strings and hope that makes IDL behave a little better. Again thanks for your help already.

Ian

> In article <Pine.SUN.3.91.980430131547.324A-100000@demsyd.syd.dem.csiro.au>,
> Peter Mason <peterm@demsyd.syd.dem.CSIRO.AU> wrote:
>
>> Do you get the error during the READU call? (i.e., Not during the REPLICATE
>> call?) If so then there's something very odd going on here.
>> Can you read the first few (say 10) records correctly? If so, then I'd say
>> that there might be a bug in IDL on Linux.
>
> I seem to recall that IDL uses a lazy-malloc() which allocates memory
> without making sure there is enough available swap space. The swap
> space is allocated as a portion of memory is paged out for the first
> time. Therefore the out of memory error could during the READU call.
>
> Eric
> --
> Eric Korpela | An object at rest can never be

> korpela@ssl.berkeley.edu | stopped.
> Click for home page.

--

Ian E. Sprod
CIRES ian@ngdc.noaa.gov
NOAA/NGDC E/GC1 http://swat.ngdc.noaa.gov/~ian
325 Broadway 303-497-6284 (voice)
Boulder, CO 80303 303-497-6513 (fax)
