
Subject: Re: read a C written binary file with IDL

Posted by manodeep@gmail.com on Tue, 21 Feb 2012 03:43:50 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Feb 20, 7:49 pm, bing999 <thibaultga...@gmail.com> wrote:

> Hi,

>

> sorry to reply so late, but I was away recently.

>

> First, as the issue seems to be more complicated than i first thought,

> i have to say that the structure i have to read is bigger than what I

> wrote previously just to make an example.

>

> It actually contains (in the right order):

> int x 1

> long long x 1

> int x 5

> float x 9

> int x 1

> float x 24

>

> that is, 7 int, 33 float and 1 long long (written in C).

>

> When I look at the IDL commands you suggest, i get:

>

> print,n_tags(mystruct,/length) 176

>

> print,n_tags(mystruct,/data_length) 172

>

> So you were right.

>

> Now, the question is "How to handle that?? " :)

>

> I inserted an extra int (0L) in the structure of my IDL reading

> routine in second position of the structure (just before the long

> long), and it gives results coherent with what one gets with a correct

> Python routine. However, I am not quite satisfied with this

> solution...

>

> Does anyone can think of a better way to proceed?

>

> Thanks !!

>

>

>

>

>

>

>

```
>
>
>
>> Bill Nel wrote:
>>> On Feb 9, 11:56 pm, Manodeep Sinha <manod...@gmail.com> wrote:
>>>> Hi,
>
>>>> This is because C pads the structure to produce alignments. Under
>>>> 'normal' operations, you would expect MyStruct to be 20 bytes,
>>>> however, if you do a sizeof(struct MyStruct), you will probably see
>>>> that the size is 24. (And you can enable the warning for gcc by using
>>>> the compile time option -Wpadded).
>
>>> The N_TAGS() function has LENGTH and DATA_LENGTH keywords:
>
>>> IDL> print, n_tags(mystruct, /length)
>>>      24
>>> IDL> print, n_tags(mystruct, /data_length)
>>>      20
>
>>> I mention it because one might not think to look at the N_TAGS
>>> function
>>> for this sort of information.
>
>>> --Wayne
```

As Paul suggested, you should check out the assoc function. I can read the data in just fine through IDL:

```
-----
openu, 1, 'data.bin', /SWAP_IF_BIG_ENDIAN
a = assoc(1, {a:0L, b:0LL, c:lonarr(5),d:fltarr(9),e:0L,f:fltarr(24)})
print,a[0]
-----
```

Cheers,
Manodeep
