Subject: Re: Bug? yea or nay.
Posted by William Daffer on Fri, 16 Feb 2001 04:21:25 GMT
View Forum Message <> Reply to Message

by the way, I'm 'vapuser'

Paul van Delst <pvandelst@ncep.noaa.gov> writes:

```
> Vapuser wrote:
>>
>> I greatly simplified the structure in question to elucidate the point.
>>
>> IDL> print,!version
>> { mipseb IRIX unix 5.3 Nov 11 1999}
>>
>> IDL> r={a:0.d, c:0L} & print,n_tags(r,/length),totsize(r)
>> 16        12
>>
>
> <snip>
>
>> What say you? Bug or not?
>
> From the IDL online manual (I know you already know this, Vapuser. It's here for
> completeness):
>
> NTAGS
>   KEYWORDS
>     LENGTH
>
>     Set this keyword to return the length of the structure, in bytes.
>
>     Note - The length of a structure is machine dependent. The length
>     of a given structure will vary depending upon the host machine.
>     IDL pads and aligns structures in a manner consistent with the host
>     machine's C compiler.
>
```

I understand this, although, as I pointed out, I thought that if you always went from largest to smallest you wouldn't get into trouble since each indivudual quantity couldn't help but be on a border that was a multiple of its type.

What's strange is that the lower level I/O routines don't behave this way. How is this possilbe if it's the 'machine dependent' implementation (i.e. the padding) that determines the answer N_Tags gives, i.e how do the I/O routines do something which is *not* 'machine dependent?'

And if the lower level I/O routines don't care (and clearly they don't) how is it that n_tags isn't as smart as them?

That's the point of the 'bug,' the obvious disagreement between the two.

> This sounds like a similar scenario I have encountered with COMMON
> blocks in F77 and structures in F90 (to use or not to use the
> SEQUENCE statement? :o). If I want to read in a series of numbers of
> different types, although my data structure consists of variables
> adding up to N bytes, the actual amount of memory used _could_ be
> more depending on what system I was on. I always presumed this was
> done because various OS's were "optimised" to deal with data
> (i.e. store, retrieve) lined up on either 8-byte or 4-byte
> boundaries in actual memory. The user shouldn't have to worry about
> the number of bytes (with padding) in memory (unless you have
> *humungous* data structures).
>

Again, it isn't the fact that there's padding, but the fact of the disagreement between what N_Tags thinks is happening and what the lower lever I/O routines do.

> I wouldn't consider it a bug, but it does seem rather, uh, lazy to
> return the _actual_ memory used rather than the sum size of the
> components when the latter is what is required to define record
> sizes to read data - producing side effects such as you have
> found. I though the one of the strengths of IDL was its system
> independence?
>

Again, aside from the disagreement between N_tags and the actual I/O that's going on, I was hoping someone could explain to me *how* a structure that starts with the largest type and works downward could *possibly* have any padding.

Clearly I'm missing something, I just don't know what.

> Maybe you can sell Kodak/RSI TOTSIZE() for IDL 5.4.1? :o)
>
> BTW, on my linux box with IDL 5.4, print,n_tags(r,/length) = 12
>

Interesting!

whd
--
Outside of a dog, a book is man's best friend
Inside of a dog it's too dark to read
  -- Groucho Marx
