Subject: Re: Bug? yea or nay.
Posted by Nigel Wade on Fri, 16 Feb 2001 10:43:48 GMT
View Forum Message <> Reply to Message

>>>> >>>>>>>>>>>>>> Original Message <<<<<<<<<<<<<<<<<<

On 15/02/01, 19:42:22, Vapuser <vapuser@catspaw.jpl.nasa.gov> wrote regarding Bug? yea or nay.:

> 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
```

> Totsize is routine I wrote to calculate the size of things, primarily
> structures, by recursing through the thing and adding up the sizes of
> the various components. Lots of the structures I use here at work are
> not laid out very well, so I had to come up with some better method
> than hand counting to determine the size of structures.

```
>  IDL> help,r,/st
>  ** Structure <1007e308>, 2 tags, length=16, refs=1:
>     A               DOUBLE           3.1415927
>     C               LONG                0
```

> My understanding of the way C padded structures was that you could
> always avoid the problem of padding if you started with the largest
> quantities first and then worked your way down to the smallest. And
> that's the way the structure that alerted me to this oddity was laid
> out, first doubles, then longs and floats.

> Does it also pad the structure so that it's an integral number of
> whatever is the largest item? That's what seems to be happening.

It's slightly more complicated than you think. A C structure can have any amount of padding placed between elements, and at the end of the structure.
The only guarantee is that the address of the structure and the address of
the first element of the structure will be the same i.e. no padding at the start of the structure. An implementation is free to add what padding

it wants within the structure, but I doubt this freedom is ever
exercised.
The "extra" padding you are seeing is padding at the end of the structure
to maintain alignment of the elements in arrays of the structure.

> The strange thing is that the real structure, for which n_tags was
> reporting the wrong size,

n_tags is not reporting the wrong size, it's reporting the actual size

> was then used to correctly read data from a
> binary file.

at a guess, IDL reads structures element-by-element.

> It didn't read all the data, of course, because the
> calculation of the number of records in the file was incorrect due to
> the incorrect record size reported by n_tags.

again, n_tags is not incorrect.

> And we verified, using fstat, that the read had read
> nrecs*totsize(r) bytes of data from the file, so each record was being
> filled with totsize(r) bytes and all the quantities looked correct. If
> it had been reading and filling each record with n_tags(r) bytes,
> there should have been a problem starting with the second record,
> since it would've been off by n_tags(r)-totsize(r) bytes.

> So the low level I/O routines were correctly filling the
> structure, treating each as a structure of totsize(r) bytes, when
> n_tags was returning the size of some padded variant.

> What say you? Bug or not?

No bug - operator error.