Subject: Re: Bug? yea or nay.
Posted by Paul van Delst on Thu, 15 Feb 2001 22:34:40 GMT
View Forum Message <> Reply to Message

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.

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).

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?

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

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
pvandelst@ncep.noaa.gov Alexander Pope.

Subject: Re: Bug? yea or nay.
Posted by Todd Clements on Thu, 15 Feb 2001 23:57:31 GMT
View Forum Message <> Reply to Message

> Maybe you can sell Kodak/RSI TOTSIZE() for IDL 5.4.1? :o)

It would have to be IDL 5.4.2... 5.4.1 already exists =>
(it fixes the multiple sessions problem in UNIXes, and NO OTHER
CHANGES...they just released it a couple of days ago, I guess only to
the people who complained about it.)

Todd

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

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.

Subject: Re: Bug? yea or nay.
Posted by Paul van Delst on Fri, 16 Feb 2001 14:16:47 GMT
View Forum Message <> Reply to Message

Todd Clements wrote:
>
>> Maybe you can sell Kodak/RSI TOTSIZE() for IDL 5.4.1? :o)
>
> It would have to be IDL 5.4.2... 5.4.1 already exists =>
> (it fixes the multiple sessions problem in UNIXes, and NO OTHER
> CHANGES...they just released it a couple of days ago, I guess only to
> the people who complained about it.)

I complained quite loudly and had a nice talk with an RSI tech feller to solve my problem way back.

I didn't get no darn upgrade notice.

Nuts!

Still-working-in-evaluation-mode-ly yours

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
pvandelst@ncep.noaa.gov Alexander Pope.

Subject: Re: Bug? yea or nay.
Posted by Paul van Delst on Fri, 16 Feb 2001 14:35:09 GMT
View Forum Message <> Reply to Message

William Daffer wrote:
>
> 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.

If one of the requirements for efficient data storage and retrieval in memory is to align data boundaries defined by the largest type in the structure? (I'm postulating... not stating).

```
IDL> print,!version
{ mipseb IRIX unix 5.3 Nov 11 1999}
IDL> r={a:0.d, c:0L} & print,n_tags(r,/length)
        16
```

== Everything lined up on 8-byte boundary

```
IDL> r={c:0L,a:0.d} & print,n_tags(r,/length)
        16
```

== Still on 8-byte boundary...even though 4-byte is first.

```
IDL> r={a:0L, c:0L, d:0L} & print,n_tags(r,/length)
        12
```

== Whoa... 4-byte boundaries now. How 'bout that (I'm not being sarcastic, I did not expect this result)

The last result suggests to me that how IDL/the OS (insert hand-waving here) decides to allocate space in memory is dependent on the structure types.

> Clearly I'm missing something.

I don't think so, apart from the fact that memory allocation for structures seems to be (sort of) dynamic depending on what is in the structure.

Also, FWIW, I disagree with Nigel Wade that this is "operator error". I think N_TAGS should return the sum total of the sizes of the structure elements, NOT the amount of memory said structure occupies. If one was dealing with accessing system memory for OS purposes, fine. But IDL is a simple tool for playing with data - why should the user care whether the structure is padded in memory or not? I think your definition of TOTSIZE(r) is the "correct" one IMO.

paulv

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
pvandelst@ncep.noaa.gov Alexander Pope.

Subject: Re: Bug? yea or nay.
Posted by Craig Markwardt on Fri, 16 Feb 2001 15:22:16 GMT
View Forum Message <> Reply to Message

Vapuser <vapuser@catspaw.jpl.nasa.gov> writes:

> 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
>

Nigel got it right. The 16 includes the padding at the end of the structure, which is needed to align the *next* structure. The Alpha version of IDL does the same thing.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

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

William Daffer <whdaffer@earthlink.net> wrote:
>
> 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.

Data alignment is a performance issue, of course, but it's not quite as
simple as just having each quantity aligned on its "natural" boundary.
Hardware, especially modern RISC hardware, tends to be optimized for

certain sized operations, for example 32 bits. That is, memory ops will be done as 32-bit load/stores, arithmetic ops as 32-bit adds/subtracts, etc. If you want to add 1 to a 16-bit number, that machine is going to actually read 32 bits out of memory, save the "other" 16 bits somewhere, possibly shift the desired 16 bits into the low order half of the register, possibly mask out the other 16 bits, add 1, possibly shift the result into the other half of the register, replace the "other" 16 bits and store the 32-bit result back in memory. On such hardware it's much more efficient to ensure that frequently used 16-bit quantities are stored as 32-bit values and just "waste" the other 16 bits. Many, if not most, modern C compilers do that for you (or *to* you, depending on your point of view).

Another way of saying the above is that it's more efficient to align data on the *hardware's* natural boundary than on the *data's* natural boundary. Compilers know about that and align data accordingly.

Dave

Dave Greenwood Email: Greenwoodde@ORNL.GOV
Oak Ridge National Lab %STD-W-DISCLAIMER, I only speak for myself

---

Subject: Re: Bug? yea or nay.
Posted by Nigel Wade on Mon, 19 Feb 2001 12:21:38 GMT
View Forum Message <> Reply to Message

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

On 16/02/01, 14:35:09, Paul van Delst <pvandelst@ncep.noaa.gov> wrote regarding Re: Bug? yea or nay.:

> Also, FWIW, I disagree with Nigel Wade that this is "operator error". I think N_TAGS
> should return the sum total of the sizes of the structure elements, NOT the amount of
> memory said structure occupies.

But it doesn't, and the documentation quite clearly says it doesn't. That's why I said "operator error" and not "bug".

> If one was dealing with accessing system memory for OS
> purposes, fine. But IDL is a simple tool for playing with data - why should the user care
> whether the structure is padded in memory or not?

The user may want to find out how much memory would be used by a large
array of structures.
You can actually work out the sum of the size of the individual elements,
but you cannot
work out the amount of memory required to hold the structure.

--

Nigel Wade, System Administrator, Space Plasma Physics Group,
University of Leicester, Leicester, LE1 7RH, UK
E-mail : nmw@ion.le.ac.uk
Phone : +44 (0)116 2523568, Fax : +44 (0)116 2523555