Subject: Re: memory allocation for structure arrays
Posted by mallors on Sat, 02 May 1998 07:00:00 GMT
View Forum Message <> Reply to Message

In article <3549F646.51FF035E@ngdc.noaa.gov>,
 Ian Sprod <ian@ngdc.noaa.gov> writes:
>
> 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.
> .
> .
> .
> 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.

I was looking at the IDL Advanced Development Guide, and under the section
"Mapping of Basic Types", an IDL_STRING is defined as follows:

```
typedef struct {
   unsigned short slen; /* Length of string */
   short stype;         /* Type of string */
   char *s;             /* Pointer to string */
} IDL_STRING;
```

I would guess this is how IDL defines its strings internally,
so your structure example above would make sense, that is, each
string would take 8 bytes: two for the length "slen", two for type
"stype", and then the 4-byte pointer to the string (on most machines,
but these are all machine-dependent).

For the other case you mention

> 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).

I think this has to do with how the structure is aligned in memory, as mentioned by Peter. Some hint of this can be seen as follows:

1. test = {long: 0L, b1: 0B, b2: 0B} => length = 8
2. test = {long: 0B, b1: 0B, b2: 0L} => length = 8
3. test = {long: 0B, b1: 0L, b2: 0B} => length = 12

On a 32-bit machine (4-byte words), case 1 uses one word to hold the long, and then another word to hold both of the byte variables. Similarly, in case 2, first word for the two bytes, another word for the long. But for case 3, the first word holds the first byte. Then we need another word to hold the long, because there is not enough room left in the first word. Then a third word is needed for the last byte, for a total of 3 words = 12 bytes.

-bob mallozzi

--
Robert S. Mallozzi
http://cspar.uah.edu/~mallozzir/