Subject: Re: problem with ASSOC
Posted by David Theil on Thu, 03 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

> If you change your structure from
> {Test, hh:0, mm:0, ss:0, ut:0.0}
> to {Test, hh:0, mm:0, ss:0, junk:0, ut:0.0}

Another safer way would be to force one of the fields
to be a long.
Perhaps,

  {Test, hh:0, mm:0, ss:0l, ut:0.0}

I think integers in IDL are *always* shorts, so this
will ensure that each record is 12 bytes long.
David Theil

Subject: Re: problem with ASSOC
Posted by Peter Mason on Thu, 03 Jul 1997 07:00:00 GMT
View Forum Message <> Reply to Message

On Wed, 2 Jul 1997, Brian Jackel wrote:
> I've run across some strange behaviour with ASSOC (used for easy
> random file access), and I was wondering if
>
> 1) anyone has seen this before
> 2) anyone would like to test my sample script on other platforms
>
> Basically, I make a small file with a number of records, and can read
> it back in with READU. However, trying to ASSOC gives some very
> strange results. Any comments would be greatly appreciated. The
> sample script is included below.
<cut>

I think that your code is showing up an inconsistency in IDL's handling
of structure padding.

Although your structure has 10 bytes worth of data (3 INTs followed by a
FLOAT), IDL stores it *in memory* with some padding (2 bytes on my platforms -
Win95 and Digital Unix) before the float, to align the float on a longword (or
higher?) boundary. (The structure size is reported as 12 bytes by
HELP,/str,record in your program.)
Ordinary READU and WRITEU appear to be smart enough to insert these padding
bytes when reading from disk to memory, and to strip them out when writing
from memory to disk.

However, reading by means of ASSOC seems to work at a lower level. I think that the ASSOC read is reading 12 bytes at a time (i.e., it's expecting the padding bytes to be saved on disk), and this doesn't match the way in which the data was written.

If you change your structure from
{Test, hh:0, mm:0, ss:0, ut:0.0}
to {Test, hh:0, mm:0, ss:0, junk:0, ut:0.0}
Then this will naturally align UT on a longword.
Your prog. runs correctly on my platforms with this mod.
However, I think that this is still a bit risky because IDL's structure padding may not be the same on all platforms. And it can be a bit strange. e.g., If you try the structure {Test, ut:0.0, hh:0, mm:0, ss:0}, IDL *still* inserts 2 padding bytes somewhere.

You could also try writing with ASSOC, to match the reading. (I haven't tried this out.)

BTW, your original program crashed out my Win95 IDL session - in IDL 4 and 5. (And I'm talking about a "This program performed an illegal operation and will be shut down" sort of crash.)

Peter Mason
