Subject: Re: reading unformatted data into a structure
Posted by David Fanning on Tue, 19 Mar 2002 02:52:42 GMT
View Forum Message <> Reply to Message

Rick Towler (rtowler@u.washington.edu) writes:

```
>  I have an unformatted data file which I am trying to read and I am having
>  only limited success.  The data file was created on a Sun machine and it is
>  being read on a win32 machine (I have set the swap_if_little_endian keyword
>  on the open statement).  As for the file format, all I have to go on is the
>  C header file that describes the records in the file.
>
>  Early on in the header I find this:
>
>  //  Variable types:
>  //     int   is 32 bit integer.
>  //     short is 16 bit integer.
>
>
>  Here are the first few lines of the header (all of the data types found in
>  the header are in these lines):
>
>  typedef struct
>  {
>     int fileFormatVersion;     /* The file format version number. */
>     int points;                /* Number of xyz points on the file. */
>     int time;                 /* UNIX time of file generation,
>                               seconds since 1/1 1970. */
>     unsigned short coordSys;   /* The coordinate system used:
>       0 = Geographical lat/long.
>       1 = Projection */
>     char projection[32];       /* Projection name (see above)   */
>     int projParam1;           /* 1. projection parameter (see above). */
>     int projParam2;           /* 2. projection parameter (see above). */
>     int projParam3;           /* 3. projection parameter (see above). */
>     int projParam4;           /* 4. projection parameter (see above). */
>     int projParam5;           /* 5. projection parameter (see above). */
>     char datum[32];            /* Name of the datum (see above).   */
>     char ellLargeHalfAxis[16]; /* Large half axis of the ellipsoide. */
>     ....
>
>
>  From the comments about variable types and the structure definition I
>  created an IDL structure that I think matches the C struct (again, I'll list
>  the first few lines):
>
>  header = {  fileFormatVersion:0L, $
>              points:0L, $
```

>   time:0L, $
>   coordSys:0, $
>   projection:bytarr(32), $
>   projParam1:0L, $
>   projParam2:0L, $
>   projParam3:0L, $
>   projParam4:0L, $
>   projParam5:0L, $
>   datum:bytarr(32), $
>   ellLargeHalfAxis:bytarr(16), $
>   ....
>
>
> I open the data file and read the header like so:
>
>   openr, lun, 'J:\hydrographic\1.xyz', /get_lun, $
>   /swap_if_little_endian
>   readu, lun, header
>
>
>
> Am I doing this correctly? I know I am close because I get what seems like
> a valid time and I know I get a valid datum. But my data doesn't seem
> right. I just want to check that I am using the correct types in my IDL
> struct and that I am reading in the data correctly.

It certainly appears correct to me, at first glance,
although it is possible to run into a lot of weird
things with data. It always helps to have a good
example of what the numbers are *suppose* to be.
That's about the only way that you can find out
the documentation forgot to mention that those
integers were *unsigned* integers, for example. :-)

> Also, I know I have seen this but how do I convert the decimal ASCII chars
> in my bytarr's to a string?

Use the STRING function:

```
  Print, String(header.projection)
```

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting

Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: reading unformatted data into a structure
Posted by ignore_this_adress on Tue, 19 Mar 2002 08:38:09 GMT
View Forum Message <> Reply to Message

Rick Towler wrote:

> I have an unformatted data file which I am trying to read and I am having
> only limited success. The data file was created on a Sun machine and it is
> being read on a win32 machine (I have set the swap_if_little_endian keyword
> on the open statement). As for the file format, all I have to go on is the
> C header file that describes the records in the file.

>
> Am I doing this correctly? I know I am close because I get what seems like
> a valid time and I know I get a valid datum. But my data doesn't seem
> right. I just want to check that I am using the correct types in my IDL
> struct and that I am reading in the data correctly.

Most compilers have switches to set memory alignment. This has of course to match the way the IDL structure is aligned in memory for IDL to read in the struct properly. Without other tools, I would try the following :
count the total number of allocated bytes in the header, and compare it to the raw data filesize. If the sizes don't match, you'll know you have run into
problems. Alternatively, you can count the number of elements in the IDL struct :-) If the size match, you'll probably have the right data.

Using a hexeditor may be informative when looking at your raw data format.

Subject: Re: reading unformatted data into a structure
Posted by Nigel Wade on Tue, 19 Mar 2002 09:42:54 GMT
View Forum Message <> Reply to Message

Rick Towler wrote:

> I have an unformatted data file which I am trying to read and I am having
> only limited success. The data file was created on a Sun machine and it
> is being read on a win32 machine (I have set the swap_if_little_endian
> keyword
> on the open statement). As for the file format, all I have to go on is
> the C header file that describes the records in the file.

```
>
>  Early on in the header I find this:
>
>  //  Variable types:
>  //     int   is 32 bit integer.
>  //     short is 16 bit integer.
>
>
>  Here are the first few lines of the header (all of the data types found in
>  the header are in these lines):
>
>  typedef struct
>  {
>     int fileFormatVersion;     /* The file format version number. */
>     int points;                /* Number of xyz points on the file. */
>     int time;                  /* UNIX time of file generation,
>                                seconds since 1/1 1970. */
>     unsigned short coordSys;   /* The coordinate system used:
>       0 = Geographical lat/long.
>       1 = Projection */
>     char projection[32];       /* Projection name (see above)   */
>     int projParam1;            /* 1. projection parameter (see above). */
>     int projParam2;            /* 2. projection parameter (see above). */
>     int projParam3;            /* 3. projection parameter (see above). */
>     int projParam4;            /* 4. projection parameter (see above). */
>     int projParam5;            /* 5. projection parameter (see above). */
>     char datum[32];            /* Name of the datum (see above).   */
>     char ellLargeHalfAxis[16]; /* Large half axis of the ellipsoide. */
>     ....
>
>
>  From the comments about variable types and the structure definition I
>  created an IDL structure that I think matches the C struct (again, I'll
>  list the first few lines):
>
>  header = {  fileFormatVersion:0L, $
>              points:0L, $
>              time:0L, $
>              coordSys:0, $
>              projection:bytarr(32), $
>              projParam1:0L, $
>              projParam2:0L, $
>              projParam3:0L, $
>              projParam4:0L, $
>              projParam5:0L, $
>              datum:bytarr(32), $
>              ellLargeHalfAxis:bytarr(16), $
>              ....
```

>
>
> I open the data file and read the header like so:
>
> openr, lun, 'J:\hydrographic\1.xyz', /get_lun, $
> /swap_if_little_endian
> readu, lun, header
>
>
>
> Am I doing this correctly? I know I am close because I get what seems
> like
> a valid time and I know I get a valid datum. But my data doesn't seem
> right. I just want to check that I am using the correct types in my IDL
> struct and that I am reading in the data correctly.

The types appear to be correct at first glance. Do you know how the data were written out from the C program? Was it in a single write, or were they written as individual elements? If it was a single write there *may* be packing problems in the struct, although the definition above wouldn't appear to require any packing, the C compiler could have decided otherwise.

My favourite tool for debugging these sorts of problems is a binary/hex editor or dump tool such as od in UNIX. I don't know of similar tools for Windows as I don't use that platform. But there's no substitute for getting your hands dirty, delving into the data file and looking at the contents byte by byte to ensure you know *exactly* what its contents are. You really need at least one record converted to human readable form by some alternate means so you can compare that to the contents of the file and be sure you can identify the components in the binary structure.

I know from past experience that relying on the documentation can be an exercise in futility - it's all too easy for some well meaning individual to make an alteration to the code and not bother to change the documentation.

--

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

Subject: Re: reading unformatted data into a structure
Posted by Nigel Wade on Tue, 19 Mar 2002 10:26:38 GMT

Nigel Wade wrote:

> If it was a single write there *may*
> be packing problems in the struct, although the definition above wouldn't
> appear to require any packing, the C compiler could have decided
> otherwise.
>

Sorry to follow up on my own post, but I've just noticed that there is a
potential alignment problem, here:

```
  unsigned short coordSys;   /* The coordinate system used:
    0 = Geographical lat/long.
    1 = Projection */
  char projection[32];       /* Projection name (see above)   */
  int projParam1;            /* 1. projection parameter (see above). */
```

this int may well need to be aligned on a word boundary. If so 16 bits of
padding will have been inserted into the struct by the C compiler to align
it. If the struct is output by a single write this padding will have been
output with the rest of the data.

On the PC, IDL may have been built to allow structs without padding, so the
same aligment problem might not appear in IDL, so reading the struct as a
single item in IDL will store those 16 bits of padding into projParam1, and
all other items in the struct will be similarly offset.

If the struct was part of an array, then a further 16 bits of padding may
have been added at the end of the struct to ensure that the int which is
the first element of the second struct in the array was properly aligned.

--

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

Subject: Re: reading unformatted data into a structure
Posted by Rick Towler on Tue, 19 Mar 2002 16:49:39 GMT

>> Also, I know I have seen this but how do I convert the decimal ASCII

chars
>> in my bytarr's to a string?
>
> Use the STRING function:
>
>   Print, String(header.projection)
>

I actually want the ASCII characters that the char array refers to.
Interestingly, when I do use the string function I get a blank line. I have
tried different format statements to no avail.

IDL> print,header.datum
  0   0 87 71 83 56 52   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0
IDL> print,string(header.datum)

IDL>

-Rick

Subject: Re: reading unformatted data into a structure
Posted by David Fanning on Tue, 19 Mar 2002 17:23:02 GMT
View Forum Message <> Reply to Message

Rick Towler (tsehai@attbi.com) writes:

> I actually want the ASCII characters that the char array refers to.
> Interestingly, when I do use the string function I get a blank line. I have
> tried different format statements to no avail.
>
> IDL> print,header.datum
>   0   0 87 71 83 56 52   0   0   0   0   0   0   0   0   0   0   0
> 0   0   0   0   0   0   0   0   0   0   0   0   0
> IDL> print,string(header.datum)
>

It's hard to know what this should be. I agree with
previous posters that you may have structure alignment
problems. But, typically, if a string were set to bytes
then blank characters would have the value 32, not 0.
(I don't know what the ASCII character is for 0B.)

But, if you do this:

   IDL> indices = Where(header.datum EQ 0, count)

```
IDL> IF count GT 0 THEN header.datum[indices] = 32B
IDL> Print, String(header.datum)
    WGS84
```

I don't know why this works, exactly. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: reading unformatted data into a structure
Posted by Rick Towler on Tue, 19 Mar 2002 17:54:42 GMT
View Forum Message <> Reply to Message

```
>
>    unsigned short coordSys;   /* The coordinate system used:
>      0 = Geographical lat/long.
>      1 = Projection */
>    char projection[32];       /* Projection name (see above)   */
>    int projParam1;           /* 1. projection parameter (see above). */
>
> this int may well need to be aligned on a word boundary. If so 16 bits of
> padding will have been inserted into the struct by the C compiler to align
> it. If the struct is output by a single write this padding will have been
> output with the rest of the data.
>
> On the PC, IDL may have been built to allow structs without padding, so
the
> same aligment problem might not appear in IDL, so reading the struct as a
> single item in IDL will store those 16 bits of padding into projParam1,
and
> all other items in the struct will be similarly offset.
>
```

This seems to be the problem. I was wondering why I had 2 null bytes at the beginning of my next bytarr. I had actually stumbled on the data by padding the entire header but this meant that most of my header data was wacked.

> If the struct was part of an array, then a further 16 bits of padding may
> have been added at the end of the struct to ensure that the int which is
> the first element of the second struct in the array was properly aligned.

>

The struct was not part of an array because I think that I am reading the
correct data.

Thanks!

-Rick

Subject: Re: reading unformatted data into a structure
Posted by Nigel Wade on Wed, 20 Mar 2002 10:20:30 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Rick Towler (tsehai@attbi.com) writes:
>
>> I actually want the ASCII characters that the char array refers to.
>> Interestingly, when I do use the string function I get a blank line. I
>> have tried different format statements to no avail.
>>
>> IDL> print,header.datum
>> 0 0 87 71 83 56 52 0 0 0 0 0 0 0 0 0 0 0
>> 0
>> 0 0 0 0 0 0 0 0 0 0 0 0 0
>> IDL> print,string(header.datum)
>>
>
> It's hard to know what this should be. I agree with
> previous posters that you may have structure alignment
> problems. But, typically, if a string were set to bytes
> then blank characters would have the value 32, not 0.
> (I don't know what the ASCII character is for 0B.)
>
> But, if you do this:
>
> IDL> indices = Where(header.datum EQ 0, count)
> IDL> IF count GT 0 THEN header.datum[indices] = 32B
> IDL> Print, String(header.datum)
> WGS84
>
> I don't know why this works, exactly. :-)
>
> Cheers,
>
> David

I think the problem is to do with the way C handles strings. I presume, from working with DLMs that internally in IDL strings are handled in C.

A string in C is terminated by a null (0) character. So, the above sequence of bytes would be interpreted as an empty string because of the initial 0, regardless of what followed it. I am guessing that the actual string begins in the third element, and the first two bytes are padding. The 8th element, the next 0, is the string terminator put in by C, and the rest are unitialised elements in the char array which just happen to be 0s.

If the padding is removed, so the byte array doesn't begin with a 0, then the converstion to a string should be ok, eg.

```
IDL> print,a
  0   0  87  71  83  56  52   0   0   0   0   0   0   0   0   0   0   0
0   0
IDL> print,string(a(2:*))
WGS84
```

--

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