Subject: Re: writing a structure with pointers
Posted by davidf on Wed, 15 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Lisa Bryan (lbryan@arete-az.com) writes:

> Could someone tell me the easiest way to do the following.
>
> state = {.....big long huge structure with many substructures and lots
>
> of pointers all over the place....}
> writeu,unit,state
>
> I'm trying to write my state structure into an unformatted file and
> get the error:
>
> WRITEU: Expression containing pointers not allowed in this context:
> STATE.

Oh, oh. Looks like someone will have to write a recursive
procedure to example the fields of structures and dereference
the pointer fields. It is a shame there isn't a DEREFERENCE
keyword that could be used with the WRITEU command.

The alternative, of course, is to SAVE the structure. SAVE
will store the pointers along with the data they point to, so
that all can be RESTORED later. This is not always my preferred
solution, since it is not guaranteed to work with IDL upgrades
and variables that are not named carefully can be a problem.
I can imagine that pointer variables might be even more
difficult to handle than normal variables.

Absent RSI adding some capability to WRITEU, how about you
let us know when you have this program written, Lisa. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: writing a structure with pointers
Posted by davidf on Thu, 16 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

I wrote yesterday, responding to a question by Lisa Bryan:

> The alternative, of course, is to SAVE the structure. SAVE
> will store the pointers along with the data they point to, so
> that all can be RESTORED later. This is not always my preferred
> solution, since it is not guaranteed to work with IDL upgrades.

It has been pointed out to me by someone who certainly knows
how this works that this is not accurate. RSI does not guarantee
that *code* in IDL SAVE files (i.e., compiled procedures and
functions) will be upwards compatible, but that *data* stored
in such files will *always* be available in future upgrades.

Sorry for the confusion.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: writing a structure with pointers
Posted by lbryanNOSPAM on Thu, 16 Jul 1998 07:00:00 GMT
View Forum Message <> Reply to Message

On Wed, 15 Jul 1998 20:35:13 -0600, davidf@dfanning.com (David
Fanning) wrote:

> Lisa Bryan (lbryan@arete-az.com) writes:
>
>> Could someone tell me the easiest way to do the following.
>>
>> state = {.....big long huge structure with many substructures and lots
>>
>> of pointers all over the place....}
>> writeu,unit,state
>>

>
> The alternative, of course, is to SAVE the structure. SAVE
> will store the pointers along with the data they point to, so
> that all can be RESTORED later. This is not always my preferred
> solution, since it is not guaranteed to work with IDL upgrades
> and variables that are not named carefully can be a problem.
> I can imagine that pointer variables might be even more
> difficult to handle than normal variables.
>
> Absent RSI adding some capability to WRITEU, how about you
> let us know when you have this program written, Lisa. :-)
>
> Cheers,
>
> David

Thanks David and Martin,

I might use the save/restore option for the time being and save the
recursive procedure for another day (Not that it didn't sound like a
fun challenge!). Thanks for giving me a good direction. Since my
most immediate desire is to be able to have a user of my GUI be able
to revert the data entry fields to a given point in time, it looks
like save/restore will do that (given the cautions that David
mentioned). Am I correct in understanding that since I have no
external information on the format of these files, they cannot be
accesed except through the IDL RESTORE command nor ammended in any
way?

Lisa

Arete Associates
Tucson, Arizona

Subject: Re: writing a structure with pointers
Posted by Martin Schultz on Thu, 16 Jul 1998 07:00:00 GMT

View Forum Message <> Reply to Message

David Fanning wrote:
>
> Lisa Bryan (lbryan@arete-az.com) writes:
>
>> Could someone tell me the easiest way to do the following.
>>
>> state = {.....big long huge structure with many substructures and lots
>>
>> of pointers all over the place....}

>> writeu,unit,state
>>
>> I'm trying to write my state structure into an unformatted file and
>> get the error:
>>
>> WRITEU: Expression containing pointers not allowed in this context:
>> STATE.
>
> Oh, oh. Looks like someone will have to write a recursive
> procedure to example the fields of structures and dereference
> the pointer fields. It is a shame there isn't a DEREFERENCE
> keyword that could be used with the WRITEU command.
>
Hi Lisa, David,

David, didn't you yourself have this recursive program that extracts all structure tags on your webpage? I think, this could be a good starting point for a general routine to write structures with pointers and structures with pointers ...

But first, let's step back a little: I don't think there is an easy "generic" solution to this problem, because a pointer is only a long word, and you don't "know" the size of the stuff it points to off-hand. Hence, you can't simply de-reference all pointers and store the "real" data into a file, because you won't be able to read it again. So you actually have to come up with a proprietary file format that contains the tag names *and* the data/pointers you want to store (which is somewhat similar to what SAVE will do, except that you can have more control over it).

Now here is what I would do:
Start your file with some unique file type identifier (just a string of fixed length, e.g. BYTE('binary IDL structure file'), which makes it easier to analyze the file if you forgot what it is, and you try a more, vi, edit, etc.) Then write out the structure tagnames [fixed string length, i.e. BYTE() !(*)] recursively, thereby adding a type and size information (e.g. just print out the size array, although it would be easier to handle for input if you had a fixed length type information field, e.g. type [note that structures have 8, pointers have 10 when you adopt the "size" types], dim1, dim2, dim3,dim4 [should all be LONG !!]). Finally take a second recursive turn through the structure and write out all values, de-referencing your pointers on the fly.

(*) In a way that is going "back" to e.g. FORTRAN, where you have to deal with fixed string lengths, but the advantage is that you know the size of your "header", which makes it much easier to read. What I do to get a fixed length string is:

```
   flstring = ([ byte(str), replicate(32B,MAXLEN) ])[0:MAXLEN-1]
```

When you read a structure back in, you may want to check out my CHKSTRU function (attached below), which allows you to test (a) whether a variable is a structure, and (b) whether it contains the tags that you need {does not operate recursively though}.

Hope, this helps a little,
Martin.

-- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
109 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
Internet-homepage: http://www-as.harvard.edu/people/staff/mgs/
-- -------

```
;------------------------------------------------------------ --
;+
; NAME:
;        CHKSTRU  (function)
;
; PURPOSE:
;        check validity of a structure and test if necessary
;        fields are contained
;
; CATEGORY:
;        tools
;
; CALLING SEQUENCE:
;        res=CHKSTRU(STRUCTURE,FIELDS [,/VERBOSE])
;
; INPUTS:
;        STRUCTURE --> the structure to be tested. If STRUCTURE is
;             not of type structure, the function will return 0
;
;        FIELDS --> a string or string array with field names to
;             be contained in STRUCTURE. CHKSTRU returns 1 (true)
;             only if all field names are contained in STRUCTURE.
;             The entries of FIELDS may be upper or lowercase.
;
; KEYWORD PARAMETERS:
;        INDEX --> a named variable that will contain the indices of
;             the required field names in the structure. They can then
;             be assessed through structure.(index(i)) . Index will
;             contain -1 for all fields entries that are not in the
;             structure.
;
;         /VERBOSE --> set this keyword to return an error message
;             in case of an error.
;
; OUTPUTS:
;         CHKSTRU returns 1 if successful, otherwise 0.
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;
; EXAMPLE:
;         test = { a:1, b:2, c:3 }
;         required = ['a','c']
;         if CHKSTRU(test,required) then print,'found a and c.'
;
; MODIFICATION HISTORY:
;         mgs, 02 Mar 1998: VERSION 1.00
;         mgs, 07 Apr 1998: - second parameter (FIELDS) now optional
;
;-
; Copyright (C) 1998, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine chkstru"
 ;------------------------------------------------------------- --


function chkstru,structure,fields,index=index,verbose=verbose


     ; default index
     index = -1

     ; first check number of parameters (must be at least 1)
     if (n_params() lt 1) then begin
         if(keyword_set(verbose)) then $
```

```
        print,'CHKSTRU: ** invalid number of parameters ! **'
        return,0
    endif


; check if the user really passed a structure

s = size(structure)
if (s(1+s(0)) ne 8) then begin
    if(keyword_set(verbose)) then $
        print,'CHKSTRU: ** No structure passed ! **'
    return,0
endif

; only one parameter: then we are finished
if (n_params() eq 1) then return,1



; see if required field names are contained in the structure
; and return indices of these fields

names = tag_names(structure)
index = intarr(n_elements(fields)) - 1   ; default index to 'not found'

for i=0,n_elements(fields)-1 do begin
    ind = where(names eq strupcase(fields(i)))
    if (ind(0) lt 0) then begin
        if(keyword_set(verbose)) then $
           print,'CHKSTRU: ** Cannot find field '+fields(i)+' ! **'
    endif else index(i) = ind(0)
endfor


; check minimum value of index field: -1 indicates error
return,(min(index) ge 0)

end
```

File Attachments

1) chkstru.pro, downloaded 131 times