
Subject: Re: undefining structures ?

Posted by [rudu](#) on Fri, 17 Dec 1993 16:24:31 GMT

[View Forum Message](#) <> [Reply to Message](#)

> I am trying to read in files of a priori undefined organization within IDL.
>
> The files are "tabular" binary files (an a priori unknown number of columns
> of an a priori unknown type with an a priori unknown number of rows).
> "a priori unknown" means documented in a file header.
... and so on

I had a similar problem with some data and I solved it with a .pro I've included at the end of this message. It creates the structure by actually writing a .pro in the current directory and compiling this .pro; so you must have rw privileges. It has a switch which allows you to delete the .pro, but after running this create_struct.pro many times, you get quite a few structures in your IDL session! I've never had a problem with this, although I don't think I ever got past 100 structures in one IDL session (never tried). I got it from IDL Astronomy User's library. I deserve no credit for this .pro, aside from the fact that I found it while net.surfing. I hope it proves useful.

Don

donald.rudy@mve.aero.org

```
pro create_struct, struct, strname, tagnames, tag_descript, DIMEN = dimen, $  
    CHATTER = chatter, NODELETE = nodelete
```

```
;  
;+  
; NAME:  
;   CREATE_STRUCT  
; PURPOSE:  
;   Dynamically create an IDL structure variable from list of tag names  
;   and data types of arbitrary dimensions. Useful when the type of  
;   structure needed is not known until run time.  
; CALLING SEQUENCE:  
; CREATE_STRUCT, STRUCT, strname, tagnames, tag_descript, DIMEN = ,  
;   CHATTER= , NODELETE =  
; INPUTS:  
; STRNAME - name to be associated with structure (string)  
;           Must be unique for each structure created. Set  
;           STRNAME = " to create an anonymous structure  
; TAGNAMES - tag names for structure elements  
;           (string or string array)  
; TAG_DESCRIPTOR - String descriptor for the structure, containing the  
;                 tag type and dimensions. (e.g., 'A(2),F(3),I', will be  
;                 the descriptor for a structure with 3 tags, strarr(2),
```

```

;         fltarr(3) and Integer scalar, respectively.
;         Allowed types are 'A' for strings, 'B' for unsigned byte
;         integers, 'I' for integers, 'L' for longword integers,
;         'F' for floating point, 'D' for double precision
;
; OPTIONAL INPUTS:
; DIMEN - number of dimensions of structure array (default is 1)
; CHATTER - If /CHATTER is set, then CREATE_STRUCT will display
;           the dimensions of the structure to be created, and prompt
;           the user whether to continue. Default is no prompt.
; NODELETE - If /NODELETE is set, then the temporary file created
;           CREATE_STRUCT will not be deleted upon exiting. See below
; OUTPUTS:
; STRUCT - IDL structure, created according to specifications
;
; EXAMPLES:
; IDL> create_struct, new, 'name', ['tag1', 'tag2', 'tag3'], 'D(2),F,A(1)'
;
; will create a structure variable new, with structure NAME
;
; To see the structure of new:
;
; IDL> help,new,/struc
; ** Structure NAME, 3 tags, 20 length:
; TAG1      DOUBLE      Array(2)
; TAG2      FLOAT       0.0
; TAG3      STRING      Array(1)
;
; PROCEDURE:
; Generates a temporary procedure file using input information with
; the desired structure data types and dimensions hard-coded.
; This file is then executed with CALL_PROCEDURE.
;
; NOTES:
; A temporary .pro file is created to define structure in the default
; directory, so writing privileges are required. CREATE_STRUCT will
; abort with an information messages if a file already exists with the
; name temp_"strname".pro
;
; At present, can fail if a tag_name cannot be used as a proper
; structure component definition, e.g., '0.10' will not
; work, but a typical string like 'RA' or 'DEC' will.
; A partial workaround checks for characters '\' and '/'
; and '.' and converts them to '_'. in a tag_name.
;
; For more information about structures, see IDL Users' Guide,
; Chapter 8
; RESTRICTIONS:

```

```

; *** The name of the structure must be unique, for each structure created.
; Otherwise, the new variable will have the same structure as the
; previous definition (because the temporary procedure will not be
; recompiled). No error message will be generated ***
;
; SUBROUTINES CALLED:
;   function gettok, function repchr
;
; MODIFICATION HISTORY:
; Version 1.0 RAS January 1992
;   Modified 26 Feb 1992 for Rosat IDL Library (GAR)
;   Modified Jun 1992 to accept arrays for tag elements -- KLV, Hughes STX
;   Accept anonymous structures W. Landsman HSTX Sep. 92
;-
;-----
npar = N_params()

if (npar LT 4) then begin
  print,'Syntax - CREATE_STRUCT, STRUCT, strname, tagnames, tag_descript,
  print,'          DIMEN = dimen, chatter=chatter (0)'
  return
endif

if (N_elements(chatter) eq 0) then chatter = 1      ;default is 1
if (N_elements(dimen) eq 0) then dimen = 1         ;default is 1

if (dimen lt 1) then begin
  print,' Number of dimensions must be >= 1. Returning.'
  return
endif

; For anonymous structure, strname = "
  anonymous = 0b
  if (strlen( strtrim(strname,2)) EQ 0 ) then anonymous = 1b

; --- Determine if a file already exists with same name as temporary file

tempfile = 'temp_' + strname
list = findfile( tempfile + '.pro', COUNT = Nfile)
if (Nfile GT 0) then begin
  print,' Temporary file '+tempfile+' found. Returning.'
  return
endif
;
;
;
good_fmts = [ 'A', 'B', 'I', 'L', 'F', 'D' ]
fmts = [ "" , 'OB', 'O', 'OL', '0.0', '0.0D0' ]
arrs = [ 'strarr', 'bytarr', 'intarr', 'lonarr', 'fltarr', 'dblarr' ]

```

```

ngoodf = N_elements(good_fmfs)
;
; If tagname is a scalar string separated by commas, convert to a string array

tagname = tagnames
sz_name = size( tagnames )
if ( sz_name(0) EQ 0 ) then begin
    tempname = tagnames
    tagname = gettok(tempname,',')
    while (tempname NE "") do tagname = [ tagname, gettok(tempname,',') ]
endif else tagname = tagnames

Ntags = N_elements(tagname)

; Replace any illegal characters in the tag names with an underscore

bad_chars = [ '\', '/', ']'
for k = 0, N_elements( bad_chars) -1 do $
    tagname = repchr( tagname, bad_chars(k), '_' )

; If user supplied a scalar string descriptor then we want to break it up
; into individual items. This is somewhat complicated because the string
; delimiter is not always a comma, e.g. if 'F,F(2,2),I(2)', so we need
; to check positions of parenthesis also.
;
sz = size(tag_descript)
if sz(0) EQ 0 then begin
    tagvar = strarr( Ntags)
    temptag = tag_descript
    for i = 0, Ntags - 1 do begin
        comma = strpos( temptag, ',' )
        lparen = strpos( temptag, '(' )
        rparen = strpos( temptag, ')' )
        if ( comma GT lparen ) and (comma LT Rparen) then pos = Rparen+1 $
            else pos = comma

        if pos EQ -1 then begin
            if i NE Ntags-1 then message, $
            'WARNING - could only parse' + strtrim(i+1,2) + 'string descriptors'
            tagvar(i) = temptag
            goto, DONE
        endif else begin
            tagvar(i) = strmid( temptag, 0, pos )
            temptag = strmid( temptag, pos+1, 120)
        endelse
    endfor
    DONE:

endif else tagvar = tag_descript

```

; Break up the tag descriptor into a format and a dimension

```
tagfmts = strmid( tagvar, 0, 1)
tagdim = strtrim( strmid( tagvar, 1, 80),2)
;
; create string array for IDL statements, to be written into
; 'temp_'+strname+'.pro'
;
pro_string = strarr (ntags + 2)
;
if (dimen EQ 1) then begin

    pro_string(0) = "struct = { " + strname + " $"
    pro_string(ntags+1) = " } "

endif else begin

    dimen = fix(dimen)
    pro_string(0) = "struct " + " = replicate ( { " + strname + " $"
    pro_string(ntags+1) = " } , " + string(dimen) + ")"

endelse

for i = 0, ntags-1 do begin

    goodpos = -1
    try = strupcase( tagfmts(i) )
    for j = 0, ngoodf-1 do $
        if ( strpos( try, good_fmts(j)) ge 0 ) then goodpos = j

    if ( goodpos LT 0) then begin
        print, ' Format not recognized: ' + tagfmts(i)
        print, ' Allowed formats are :', good_fmts
        stop, ' Redefine tag format ( ' + string(i) + ' ) or quit now'
    endif else $

    if (tagdim(i) EQ "") then fmt = fmts(goodpos) else $
        fmt = arrs(goodpos) + tagdim(i)
    if anonymous and ( i EQ 0 ) then comma = " else comma = " , "

    pro_string(i+1) = comma + tagname(i) + ": " + fmt + " $"
endfor
```

; Check that this structure definition is OK (if chatter set to 1)

```
if keyword_set ( Chatter ) then begin
```

```

ans = "
print,' Structure ',strname,' will be defined according to the following:'
temp = repchr( pro_string, '$', "")
print, temp
read,' OK to continue? (Y or N) ',ans
if strmid(strupcase(ans),0,1) eq 'N' then begin
    print,' Returning at user request.'
    return
endif
endif
;
; ---- open temp file and create procedure
;
openw, unit, tempfile + '.pro', /get_lun
printf, unit, 'pro ' + tempfile + ', struct'
for j = 0,N_elements(pro_string)-1 do $
printf, unit, strtrim( pro_string(j) )
printf, unit, 'return'
printf, unit, 'end'
free_lun, unit
;
call_procedure, tempfile, struct

if keyword_set( NODELETE ) then begin
    message,'Created temporary file ' + tempfile + '.pro',/INF
    return
endif

if ( !version.os EQ 'vms' ) then spawn, 'delete '+ tempfile + '.pro;' else $
    spawn, 'rm ' + tempfile + '.pro'

return
end      ;pro create_struct

```
