Subject: Re: Reading files with unknown amount of dat
Posted by oet on Fri, 05 Nov 1993 08:28:00 GMT
View Forum Message <> Reply to Message

In Perl exists a function 'push' to dynamically grow up arrays. Some times
ago I wrote a similiar function in IDL. The function requires the function
datatype.pro from the JHUAPL-IDL-Library.
Both functions are included below.

--Thomas

```
; ****************************** CUT HERE ******************************

FUNCTION PUSH, oldarr, newvals
;
;+
; NAME:
; PUSH
;
; PURPOSE:
;       Pushes new values onto the end of a 1d array. The length of the
;       new array increases by the length of the appended array.
;       Works like the push - function in PERL. Structure arrays are supported.
;
; CATEGORY:
;       Array
;
;
; CALLING SEQUENCE:
;      tmparr=push(tmparr,newarr)
;
; INPUTS:
;      old array, new array to append
;
; OUTPUTS:
;      expanded array
;
;
; RESTRICTIONS:
;      Type structure only with named structures, anonymous structures
;      will terminate on incompatible type message.
;      Requires procedure datatype from the JHUAPL (or idlmeteo) library
;
;
; PROCEDURE:
;
; EXAMPLE:
```

```
;     IDL> tmparr=['Hanna','Berta','Fritz']
;     IDL> print, tmparr
;     Hanna Berta Fritz
;     IDL> tmparr=push(tmparr,'Heinz')
;     IDL> print, tmparr
;      Hanna Berta Fritz Heinz
;
;     - - - - - - - - - - - - - - - - - - -
;     IDL> tmparr=['Hanna','Berta','Fritz']
;     IDL> newarr=['Gustav','Friedrich','Max']
;     IDL> tmparr=push(tmparr,newarr)
;     IDL> print, tmparr
;     Hanna Berta Fritz Gustav Friedrich Max
;
;     - - - - - - - - - - - - - - - - - - -
;     IDL> tmparr=[1,2,3]
;     IDL> tmparr=push(tmparr,[4,5,6])
;     IDL> print, tmparr
;          1   2   3   4   5   6
;     - - - - - - - - - - - - - - - - - - -
;     Example for a structure array (database table):
;
;     IDL> tmp_attr={name:'', nr:0}                  ; define attribute
;     IDL> tmp_table=replicate(tmp_attr, 100)          ; define table
;     IDL> tmp_table(10).name='Friedrich'              ; insert values
;     IDL> tmp_table(10).nr=10
;     IDL> new_attr=tmp_table(0)                     ; define new attr
;                                          ; getting structure
;                                          ; from tmp_table
;     IDL> new_attr(0).name='Hans'                     ; insert values
;     IDL> new_attr(0).nr=39
;     IDL> tmp_table=push(tmp_table, new_attr)
;     IDL> print, 'tmp_table(10): ', tmp_table(10)
;     IDL> print, 'tmp_table(100): ', tmp_table(100)
;
;     Note: It is required to get the structure for a new attribute from
;           the table to which we want to append a new attribute. Just
;           defining a new attribute with the same structure wouldn't work
;           correctly!
;
; SEE ALSO:
;     make_array,boost_array, embed, store_array, reform, mean, makenlog
;
; MODIFICATION HISTORY:
;
;      IDLMETEO-Library  Swiss Meteorological Institute
;
;  Written by: Thomas@Oettli@sma.ch   21-Dec-1992
```

```
;
;       04-jan-1993     Th. Oettli      - added handling of structure arrays
;                                         - improved performance using array
;                                           operations instead of for loops
;       11-june-1993    Th. Oettli      - added note to RESTRICTIONS in
;                                           help section 'anonymous structures'
;
;
;-

 type = datatype(oldarr,3)
 newdim=n_elements(oldarr)+n_elements(newvals)
 old_last=n_elements(oldarr)-1
 new_last=newdim-1

 CASE type of
    'UND':     print, 'Could not evaluate datatype!'
    'BYT':     newarr=make_array(newdim,1, /BYTE)
    'INT':     newarr=make_array(newdim,1, /INT)
    'LON':     newarr=make_array(newdim,1, /LON)
    'FLT':     newarr=make_array(newdim,1, /FLOAT)
    'DBL':     newarr=make_array(newdim,1, /DOUBLE)
    'COMPLEX': newarr=make_array(newdim,1, /COMPLEX)
    'STR':     newarr=make_array(newdim,1, /STRING)
    'STC':     newarr=replicate(oldarr(0),newdim)
 ENDCASE

   newarr(0:old_last)=oldarr(*)

   newarr(old_last+1:new_last) = newvals(*)

  return, newarr
END




; ********************************** CUT HERE ***********************************


 ;-------------------------------------------------------- --
;+
; NAME:
;       DATATYPE
;
; PURPOSE:
;       Datatype of variable as a string (3 char or spelled out).
;
```

```
; CATEGORY:
;       Type Conversion
;
; CALLING SEQUENCE:
;       typ = datatype(var, [flag])
;
; INPUTS:
;       var = variable to examine.         in
;       flag = output format flag (def=0). in
;
; KEYWORD PARAMETERS:
; OUTPUTS:
;       typ = datatype string or number.   out
;          flag = 0       flag = 1          flag = 2       flag = 3
;          UND            Undefined          0              UND
;          BYT            Byte               1              BYT
;          INT            Integer            2              INT
;          LON            Long               3              LON
;          FLO            Float              4              FLT
;          DOU            Double             5              DBL
;          COM            Complex            6              COMPLEX
;          STR            String             7              STR
;          STC            Structure          8              STC
; COMMON BLOCKS:
; NOTES:
; MODIFICATION HISTORY:
;       Written by R. Sterner, 24 Oct, 1985.
;       RES 29 June, 1988 --- added spelled out TYPE.
;       R. Sterner, 13 Dec 1990 --- Added strings and structures.
; R. Sterner, 19 Jun, 1991 --- Added format 3.
;       Johns Hopkins University Applied Physics Laboratory.
;       Added to idlmeteo from the JHU/APL-Library   Nov-1992  oet@sma.ch
;
; Copyright (C) 1985, Johns Hopkins University/Applied Physics Laboratory
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made.  This
; routine is provided as is without any express or implied warranties
; whatsoever.  Other limitations apply as described in the file disclaimer.txt.
;-
 ;------------------------------------------------------------ --

 FUNCTION DATATYPE,VAR, FLAG, help=hlp

 if (n_params(0) lt 1) or keyword_set(hlp) then begin
   print,' Datatype of variable as a string (3 char or spelled out).'
   print,' typ = datatype(var, [flag])'
   print,'   var = variable to examine.         in'
   print,'   flag = output format flag (def=0). in'
```

```
  print,'   typ = datatype string or number.   out'
  print,'      flag=0    flag=1      flag=2    flag=3'
  print,'      UND       Undefined   0         UND'
  print,'      BYT       Byte        1         BYT'
  print,'      INT       Integer     2         INT'
  print,'      LON       Long        3         LON'
  print,'      FLO       Float       4         FLT'
  print,'      DOU       Double      5         DBL'
  print,'      COM       Complex     6         COMPLEX'
  print,'      STR       String      7         STR'
  print,'      STC       Structure   8         STC'
  return, -1
 endif

 IF N_PARAMS(0) LT 2 THEN FLAG = 0 ; Default flag.

 if n_elements(var) eq 0 then begin
   s = [0,0]
 endif else begin
   S = SIZE(VAR)
 endelse

 if flag eq 2 then return, s(s(0)+1)

 IF FLAG EQ 0 THEN BEGIN
   CASE S(S(0)+1) OF
   0:     TYP = 'UND'
   7:       TYP = 'STR'
   1:       TYP = 'BYT'
   2:       TYP = 'INT'
   4:       TYP = 'FLO'
   3:       TYP = 'LON'
   5:       TYP = 'DOU'
   6:       TYP = 'COM'
   7:       TYP = 'STR'
   8:       TYP = 'STC'
ELSE:       PRINT,'Error in DATATYPE'
   ENDCASE
 ENDIF ELSE if flag eq 1 then BEGIN
   CASE S(S(0)+1) OF
   0:     TYP = 'Undefined'
   7:       TYP = 'String'
   1:       TYP = 'Byte'
   2:       TYP = 'Integer'
   4:       TYP = 'Float'
   3:       TYP = 'Long'
   5:       TYP = 'Double'
   6:       TYP = 'Complex'
```

```
   7:       TYP = 'String'
   8:       TYP = 'Structure'
ELSE:       PRINT,'Error in DATATYPE'
   ENDCASE
 ENDif else IF FLAG EQ 3 THEN BEGIN
   CASE S(S(0)+1) OF
   0:     TYP = 'UND'
   7:       TYP = 'STR'
   1:       TYP = 'BYT'
   2:       TYP = 'INT'
   4:       TYP = 'FLT'
   3:       TYP = 'LON'
   5:       TYP = 'DBL'
   6:       TYP = 'COMPLEX'
   7:       TYP = 'STR'
   8:       TYP = 'STC'
ELSE:       PRINT,'Error in DATATYPE'
   ENDCASE
 endif

 RETURN, TYP

 END
```