Subject: Re: Reading files with unknown amount of data
Posted by pat on Thu, 30 Sep 1993 17:32:15 GMT
View Forum Message <> Reply to Message

Robert Davis (rdavis@aerospace.aero.org) wrote:
> I am trying to read a data file containing an unknown amount of data into
> arrays in IDL. I know the format of the data in the file, but not the number
> of pieces of data in the file. Currently, I read the file twice; once to
> determine the amount of data in the file and then a second time to actually
> read the data into an array (now that I know the size of array needed).
> Is there a better way to do this (without having to read the file twice)?

I think you want the built-in routine FSTAT.

--
"Now about those pictures..."
"I can explain! I was young! I needed the money!.."

patrick m. ryan
nasa / goddard space flight center / oceans and ice branch / hughes stx
pat@jaameri.gsfc.nasa.gov / patrick.m.ryan@gsfc.nasa.gov

Subject: Re: Reading files with unknown amount of data
Posted by jim on Thu, 30 Sep 1993 17:47:33 GMT
View Forum Message <> Reply to Message

In article <28csjc$qde@news.aero.org> rdavis@aerospace.aero.org
(Robert Davis) writes:
> I am trying to read a data file containing an unknown amount of data into
> arrays in IDL. I know the format of the data in the file, but not the number
> of pieces of data in the file. Currently, I read the file twice; once to
> determine the amount of data in the file and then a second time to actually
> read the data into an array (now that I know the size of array needed).
> Is there a better way to do this (without having to read the file twice)?
>
> Robert Davis
> Member of Technical Staff
> The Aerospace Corporation
> rdavis@aero.org
>
I assume your data file does not have a header
which contains the information you need. If the file has fixed-length
records, you could probably deduce the number of entries from the file
size. You can get the file size using FSTAT. On a Unix box you'll get
the size in bytes, which should get you to the number of records fairly
accurately. If you are running under VMS, the result might only be
good to the nearest multiple of 512 bytes.

Subject: Re: Reading files with unknown amount of data
Posted by zawodny on Thu, 30 Sep 1993 19:27:29 GMT
View Forum Message <> Reply to Message

> Robert Davis (rdavis@aerospace.aero.org) wrote:
> I am trying to read a data file containing an unknown amount of data into
> arrays in IDL. I know the format of the data in the file, but not the number
> of pieces of data in the file. Currently, I read the file twice; once to
> determine the amount of data in the file and then a second time to actually
> read the data into an array (now that I know the size of array needed).
> Is there a better way to do this (without having to read the file twice)?

Use the EOF function in IDL. Try something like this:

```
a = fltarr(n) ; or whatever is appropriate for your records

openr,1,filename

while not eof(1) do begin

 readu,1,a
 .
 . ; Process it
 .

endwhile

close,1

.
. ; More processing or plotting or whatever.
.
```

This should work as I use this all the time.

--
Joseph M. Zawodny (KO4LW) NASA Langley Research Center
Internet: zawodny@arbd0.larc.nasa.gov MS-475, Hampton VA, 23681-0001
Packet: ko4lw@wb0tax.va.usa

Subject: Re: Reading files with unknown amount of data
Posted by sterne on Thu, 30 Sep 1993 23:38:37 GMT
View Forum Message <> Reply to Message

>>>> > "Robert" == Robert Davis <rdavis@aerospace.aero.org> writes:

Robert> I am trying to read a data file containing an unknown amount of
Robert> data into arrays in IDL. I know the format of the data in the
Robert> file, but not the number of pieces of data in the file.
Robert> Currently, I read the file twice; once to determine the amount
Robert> of data in the file and then a second time to actually read the
Robert> data into an array (now that I know the size of array needed).
Robert> Is there a better way to do this (without having to read the
Robert> file twice)?

Here are a couple more variations on the theme:

1. Check the size of the file on the system. In unix, if your file
is a simple formatted file, you can use the IDL command spawn:

```
spawn,'wc -l'+filename, result
```

and result will contain the number of lines in the file.

For a binary with fixed record length, you could read the size of
the file in bytes and work out the number of records from that.

2. Building on Joseph Zawodny's suggestion of using the EOF function:
sometimes you want to keep all the data around and manipulate stuff
after you have read it all in. You can use a variation like this:

```
a = fltarr(n,50) ; or whatever is appropriate.
                 ; Initial guess of 50 records in the file.
 openr,1,filename

count = 0

while not eof(1) do begin

    if count eq (size(a))(2) then a = [[a],[a]] ; double size
                                                ; of 2nd index
    readf,1,a(*,count)

    count = count + 1

endwhile

close,1

a = a(*,0:ncount-1)  ; trim to correct size
```

I've used both methods in the past, but, assuming I have control over
the software generating the numbers, I now much prefer to change the
software to make it write the number and format of the records at the
top of the file.

Hope this helps,
  Phil
--
 -- ---------
Philip Sterne | sterne@dublin.llnl.gov
Lawrence Livermore National Laboratory | Phone (510) 422-2510
Livermore, CA 94550 | Fax (510) 422-7300

Subject: Re: Reading files with unknown amount of data
Posted by brau1231 on Thu, 04 Nov 1993 11:12:27 GMT
View Forum Message <> Reply to Message

In article <28fbv1$qg6@reznor.larc.nasa.gov>, zawodny@arbd0.larc.nasa.gov (Dr Joseph M Zawodny) writes:
|> >Robert Davis (rdavis@aerospace.aero.org) wrote:
|> >I am trying to read a data file containing an unknown amount of data into
|> >arrays in IDL. I know the format of the data in the file, but not the number
|> >of pieces of data in the file. Currently, I read the file twice; once to
|> >determine the amount of data in the file and then a second time to actually
|> >read the data into an array (now that I know the size of array needed).
|> >Is there a better way to do this (without having to read the file twice)?
|>
|> Use the EOF function in IDL. Try something like this:
|>
|>

That's good, but you can also add entry's to a array.
IDL let grow the array automatically, too.

```
 array=0.       ; to initialize a variable called array


 openr,lun,filename, /get_lun    ; no interest of 'lun's name
       point_lun,lun,0                ;  but 'lun' know what I mean
                                ;  (logical unit number)

 while not eof(lun) do begin
```

;I use mostly

```
        readf,lun,a       ; reading the entry of 'lun'

        array=[array,a]   ; idl set the variable 'array' (if existing)
                          ;     to the right dimensions / append one entry
                          ;     so-called 'a'
 endwhile
 close,lun
      free_lun,lun


;cut the first entry of the array (array=0.)

       array=(1:*)
```

from now on, the array 'array' has all entries and the right dimension.

--
Rainer Brauckhoff TU-Berlin Germany

E-Mail: brau1231@camillo.fb12.tu-berlin.de

Subject: Re: Reading files with unknown amount of data
Posted by zawodny on Thu, 04 Nov 1993 12:44:42 GMT

View Forum Message <> Reply to Message

In article <2bao2r$qi9@mailgzrz.TU-Berlin.DE> brau1231@camillo.fb12.tu-berlin.de writes:
> In article <28fbv1$qg6@reznor.larc.nasa.gov>, zawodny@arbd0.larc.nasa.gov (Dr Joseph M Zawodny) writes:
> |> >Robert Davis (rdavis@aerospace.aero.org) wrote:
> |> >I am trying to read a data file containing an unknown amount of data into
> |> >arrays in IDL. I know the format of the data in the file, but not the number
> |> >of pieces of data in the file. Currently, I read the file twice; once to
> |> >determine the amount of data in the file and then a second time to actually
> |> >read the data into an array (now that I know the size of array needed).
> |> >Is there a better way to do this (without having to read the file twice)?
> |>
> |> Use the EOF function in IDL. Try something like this:
> |>
> |>
> >
> >
> That's good, but you can also add entry's to a array.
> IDL let grow the array automatically, too.
> >
> >
> >
> array=0.   ; to initialize a variable called array

```
>
>
>  openr,lun,filename, /get_lun    ; no interest of 'lun's name
>        point_lun,lun,0               ;  but 'lun' know what I mean
>                                  ;  (logical unit number)
>
>  while not eof(lun) do begin
>
> ;I use mostly
>           readf,lun,a       ; reading the entry of 'lun'
>
>           array=[array,a]   ; idl set the variable 'array' (if existing)
>                             ;     to the right dimensions / append one entry
>                             ;     so-called 'a'
>   endwhile
>   close,lun
>        free_lun,lun
>
>
> ;cut the first entry of the array (array=0.)
>
>        array=(1:*)
>
> from now on, the array 'array' has all entries and the right dimension.
>
> --
> Rainer Brauckhoff    TU-Berlin    Germany
>
> E-Mail:    brau1231@camillo.fb12.tu-berlin.de
```

If I understand the way IDL uses memory, this "growing" of the array could be a real memory hog. As I understand it, IDL will go off and try to find a contiguous piece of memory for the new array. The areas where the old ARRAYs will not be big enough for the new version of ARRAY. So if the final size of ARRAY is 10000 points, then this method will consume (n*(n+1)/2) or 50 million points worth of space (200 MB if we are talking floating point numbers) in the process. Obviously this is a worse case scenario and is dependant upon what else you may be doing in your program. In my mind it is clearly more efficient to make an ARRAY which is way too big (like [file size in bytes]/4 elements).

--
Joseph M. Zawodny (KO4LW) NASA Langley Research Center
Internet: zawodny@arbd0.larc.nasa.gov MS-475, Hampton VA, 23681-0001
Packet: ko4lw@n4hog.va.usa

Subject: Re: Reading files with unknown amount of data
Posted by alans on Thu, 04 Nov 1993 16:09:34 GMT
View Forum Message <> Reply to Message

I always thought the best approach to the "growing array" problem was to "cache" the data in an array of a KByte or so. When the "cache" is full, append to array. So, rewriting the previously posted example program this way yields:

```
function read_vl_file, filename

  ; yes, I *did* test this, but didn't benchmark it...
  a = 0.0
  i = 0
  csize = 1024
  cache = fltarr (csize)
  openr, lun, filename, /get_lun
  while (not eof (lun)) do begin
    readf, lun, a
    cache(i) = a
    i = (i + 1) mod csize
    if (i eq 0) then $
      if ((size (out))(0) gt 0) then $
        out = [temporary (out),cache] else $
        out = cache
  endwhile

  close, lun
  free_lun, lun

  ; grab the rest of the cache.

  if (i gt 0) then $
     if ((size (out))(0) gt 0) then $
       out = [temporary (out),cache(0:i-1)] else $
       out = cache(0:i-1)
  return, out
end
```

Anyway, if there are better ways to do this, I'd sure love to hear about them.
--
Alan J.Stein MIT/Lincoln Laboratory alans@LL.mit.edu

Subject: Re: Reading files with unknown amount of data
Posted by pendleton on Thu, 04 Nov 1993 17:46:56 GMT
View Forum Message <> Reply to Message

In article <ALANS.93Nov4110936@fallout.juliet.ll.mit.edu>, alans@ll.mit.edu (A.J.Stein) writes:
> I always thought the best approach to the "growing array" problem was to
> "cache" the data in an array of a KByte or so. When the "cache" is full,
> append to array. So, rewriting the previously posted example program this
> way yields:
>
> [example function deleted]
>
> Anyway, if there are better ways to do this, I'd sure love to hear about
> them.
> --
> Alan J.Stein MIT/Lincoln Laboratory alans@LL.mit.edu

"Better" depends a lot on what your needs are, of course. For the particular data sets we deal with, reading to EOF while counting records, then creating exact-size arrays and re-reading the file makes the most sense. (Actually, it's a little more complicated since we use indexed files, but you get the idea.)

The excess I/O time we incur is siginificantly less than the time it takes the pager to go out and find more continguous memory for each append. This, after all, is more I/O unless your data set is small enough to fit in physical memory.

This method also leaves around a lot more contiguous memory that will still be available later on.

You should try it both ways, but as your data set size increases, the two-pass method will, I predict, increase your efficiency in both speed and memory utilization.

For most of our analysis tasks, we've tried to avoid array appends completely, even with 400K block (VMS) pagefiles. Appends are easy to code, but they can become real hogs in just a few passes through a WHILE loop.

Jim Pendleton, Programmer Analyst/Technical Services Specialist
GRO/OSSE, Dept. Physics & Astronomy
Northwestern University
j-pendleton@nwu.edu (708) 491-2748 (708) 491-3135 [FAX]

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
```