Subject: Re: Reading in text data
Posted by Craig Markwardt on Tue, 08 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

reardonb@my-deja.com writes:

> Hi. I am reading in text data (columns and rows of numbers) and I would
> like to know if there is a more elegant way of doing it. Currently, the
> user must specify how many columns there are. In my case the number of
> columns is manually inserted into the first line of the file like this:
>
> 3
> 0 1 2
> 1 2 3
> 2 3 4
> 3 4 5
> 4 5 6
> 5 6 7
> 6 7 8
> 7 8 9
> 8 9 10
> 9 10 11

You've already had some pretty good responses. You're really asking two questions: (1) What if I don't know how many columns there are? and (2) What if I don't know how many rows there are?

Question 1: how many columns? Answer: count them! If you can read the first line, then with judicious application of STRTRIM and STRCOMPRESS you can do this quite readily:

```
str = '' & readf, unit, str              ;; Read string
str = byte(strcompress(strtrim(str,2)))   ;; Remove spaces, convert to bytes
wh = where(str EQ 32B, n_columns)         ;; Count number of remaining spaces
n_columns = n_columns + 1
```

Then you will have to rewind the file pointer to actually read the data.

Question 2: how many rows? Answer: either count them, or use a dynamic resizing technique.

You've seen some suggestions already for counting rows, which are good. The "wc" trick works only on Unix.

The dynamic resizing technique is to grow your array as needed. I have found that growing the array with each line is too slow and memory-wasting. What I normally do is grow the size of the array by a

factor of two, up to a certain limit, beyond which the arrays grows
linearly.  This has the benefit that you do a minimum number of growth
operations for small-to-mid sized arrays.

To use your terminology, it would be something like this:

```
max_rows = 0L
counter = 0L
while NOT EOF(lun) do begin
  .... read data ....
  if count GE max_rows then begin
    if max_rows EQ 0 then max_rows = 128L     ;; Minimum array size
    max_rows = max_rows + (max_rows < 4096L)  ;; Maximum growth is 4k
    newdata = make_array(n_columns, max_rows, value=tp)  ;; Make new array

    if n_elements(data) GT 0 then $
      newdata(0,0) = data                     ;; Copy old data into newdata
    data = 0 & data = temporary(newdata)      ;; Now "data" contains new data
  endif

  data(*,counter) = temporary_data            ;; Insert one row
  counter = counter + 1
endwhile
data = data(*,0:counter-1)                    ;; Trim the array
```

Meditate on that for awhile. :-)

Good luck,
Craig

--

 ---------------------------------------------------------------- --------------
Craig B. Markwardt, Ph.D.        EMAIL:    craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 ---------------------------------------------------------------- --------------


---

Subject: Re: Reading in text data
Posted by Andy Loughe on Tue, 08 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

reardonb@my-deja.com wrote:
>
> Hi.  I am reading in text data (columns and rows of numbers) and I would
> like to know if there is a more elegant way of doing it.  Currently, the
> user must specify how many columns there are.  In my case the number of
> columns is manually inserted into the first line of the file like this:

>
> 3
> 0 1 2
> 1 2 3
> 2 3 4
> 3 4 5
> 4 5 6
> 5 6 7
> 6 7 8
> 7 8 9
> 8 9 10
> 9 10 11

There are lots of ways to attack this problem depending upon
whether you need to operate on the data one-row-at-a-time, or
if you need all the data stored into an array. Some potentially
useful ideas are illustrated in this simple procedure...

```
infile = 'column.dat'                ;** ASCII input filename.
spawn, 'wc -l < ' + infile, num_recs  ;** Number of rows in ASCII file
num_recs = long(num_recs(0))          ;** Convert to num_recs to LONG
data = strarr(num_recs)               ;** Declare output array
openr, iu, infile, /get_lun           ;** Open file
readf, iu, data                       ;** Real ALL the data as a string
free_lun, iu                          ;** Close the file

;**  Use help, to check size of arrays.  Notice record #1.
for i = 0, num_recs-1 do help, long(str_sep(data(i),' ',/rem))

;**  Or use print, to check on value of array elements.
for i = 0, num_recs-1 do print, long(str_sep(data(i),' ',/rem))

end
```

--
Andrew Loughe ==========
NOAA/OAR/FSL/AD R/FS5 | email: loughe@fsl.noaa.gov
325 Broadway | wwweb: www-ad.fsl.noaa.gov/users/loughe
Boulder, CO 80305-3328 | phone: 303-497-6211 fax: 303-497-6301

---

Subject: Re: Reading in text data
Posted by davidf on Tue, 08 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Brian Reardon (reardonb@my-deja.com) writes:

> I am reading in text data (columns and rows of numbers) and I would

> like to know if there is a more elegant way of doing it. Currently, the
> user must specify how many columns there are. In my case the number of
> columns is manually inserted into the first line of the file like this:
>
> 3
> 0 1 2
> 1 2 3
> 2 3 4
> 3 4 5
> 4 5 6
> 5 6 7
> 6 7 8
> 7 8 9
> 8 9 10
> 9 10 11
>
> The attached procedure reads in the data. Is there a way to read in the
> data such that the user does not have to a priori know how many columns
> there are and such that IDL does not have to reserve a large amount of
> memory for the number of rows?

There are any number of ways to count columns and rows,
as I'm afraid you are about to find out.

But if you data is well-behaved and your taste
runs to the totally unsophisticated, you might
try the Count_Columns and Count_Rows functions
on my anonymous ftp site. For someone who is
suppose to be a professional programmer these
are laughable. But they seem to work for me
more often than not, so I continue to use them. :-)

Count_Columns uses the first line in the file as an
example of what follows. Count_Rows just counts until
it reaches the end of the file. On most of the files
I deal with, this is still orders of magnitude faster
than reading the data in a loop.

ftp://www.dfanning.com/pub/dfanning/outgoing/idl_course/coun t_columns.pro
ftp://www.dfanning.com/pub/dfanning/outgoing/idl_course/coun t_rows.pro

Cheers,

David

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

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

Subject: Re: Reading in text data
Posted by reardonb on Wed, 09 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Thanks everyone. I will be up late tonight making adjustments!
I wish there was a decent late night coffee place in Santa Fe.

Sent via Deja.com http://www.deja.com/
Before you buy.

Subject: Re: Reading in text data
Posted by Paul van Delst on Wed, 09 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

> Brian Reardon (reardonb@my-deja.com) writes:
>
>> I am reading in text data (columns and rows of numbers) and I would
>> like to know if there is a more elegant way of doing it. Currently, the
>> user must specify how many columns there are. In my case the number of
>> columns is manually inserted into the first line of the file like this:
>>
>> 3
>> 0 1 2
>> 1 2 3
>> 2 3 4
>> 3 4 5
>> 4 5 6
>> 5 6 7
>> 6 7 8
>> 7 8 9
>> 8 9 10
>> 9 10 11
>>
>> The attached procedure reads in the data. Is there a way to read in the
>> data such that the user does not have to a priori know how many columns
>> there are and such that IDL does not have to reserve a large amount of
>> memory for the number of rows?
>

Wot about DDREAD.PRO (and associated routines) by F.K.Knight? I use it

all the time. It allows you skip row, columns so the first line being a single number shouldn't matter.

Check out

http://www.astro.washington.edu/deutsch/idl/htmlhelp/library 38.html

where you'll find:

Routine Descriptions

DDREAD

[Next Routine] [List of Routines]

Name:
ddread
Purpose:
This routine reads data in formatted (or unformatted) rows and columns.
The name stands for Data Dump Read. By default, comments are skipped and the number of columns is sensed. Many options exist, e.g., selecting rows and columns, reading binary data, and selecting non-default data type and delimiters.

Examples:

```
junk = ddread(/help)                    ; get information only
array = ddread(file)                    ; read ASCII data
array = ddread(file,/formatted)         ; ditto
array = ddread(file,object=object)      ; read binary data
array = ddread(file,columns=[0,3])      ; get only 1st & 4th columns
array = ddread(file,rows=lindgen(10)+10); get only 2nd 10 rows
array = ddread(file,offset=10,last=19)  ; get rows (10,19)
array = ddread(file,/countall)          ; count comment lines
array = ddread(file,/verbose)           ; echo comment lines
array = ddread(file,type=1)             ; return bytes, not floats or longs
array = ddread(file,range=['start text','stop text'])   ; text delimiters

; Place the detailed output from a Lowtran run in a 2-D array---wow!
output = ddread('lowtran.out',range=['(CM-1)
```

(MICRN)','0INTEGRATED ABSORPTION'])
% DDREAD: Read 69 data lines selecting 14 of 14 columns; skipped
395 comment lines.
Usage:
array = ddread([file][,options][,/help])
Optional Inputs:
file = file with data; if omitted, then call pickfile.
Keywords:
/formatted, /unformatted = flags to tell IDL whether data format
is
binary or ASCII. ddread tries to determine the type
of data but it's not foolproof.
object = a string containing the IDL declaration for one
instance
of the object in an unformatted file, e.g.,
'fltarr(4)'
or
'{struct,dwell:0.,pitch:0.,yaw:0.,roll:0.}'
rows = an array to select a subset of the rows in a formatted
file
Does not count comment lines, unless /countallrows is
set!
columns = likewise for columns
type = data type of the output D=float (if '.' appears) or long
delimiter = column separater, D=whitespace
/help = flag to print header
range = start and stop row or strings,
e.g. range = ['substring in 1st line','substring in last
line']
offset = start row (read to end of file, unless last set)
last = stop row (read from start of file, unless offset set)
/countallrows = flag to count comment rows as well as data rows
(D=0)
/verbose = flag to echo comments to screen
Outputs:
array = array of data from the lines (ASCII) or objects (binary)
Common blocks:
none
Procedure:
After deciding on ASCII or binary, read file and return array.

Restrictions:
- Comments can be either an entire line or else an end of a
line, e.g.,
/* C comment. */
; IDL comment
Arbitrary text as a comment
Comment in Fortran

The next line establishes # of columns (4) & data type
(float):
6. 7 8 9
This line and the next are both considered comments.
6 comment because only one of 4 columns appears
1 2 3 4 but this line has valid data and will be read as
data

- Even if a range of lines is selected with offset, range or
last, all
lines are read. This could be avoided.

- Other routines needed:
pickfile.pro - to choose file if none is given
nlines.pro - to count lines in a file
nbytes.pro - to count bytes in a variable
replicas.pro - to replicate arrays (not scalars as in
replicate.pro)
typeof.pro - to obtain the type of a variable

Modification history:
write, 22-26 Feb 92, F.K.Knight (knight@ll.mit.edu)
allow reading with arbitrary delimiter using reads, 23 Mar 92,
FKK
add countallrows keyword and modify loop to read as little
data as possible, 20 May 92, FKK
correct bug if /formatted set, 6 Jul 92, FKK
add verbose keyword to print comments, 6 July 92, FKK
correct bug if /rows=...,/countall set, 6 July 92, FKK & EJA
add a guard against a blank line being converted to a
number, 21 Aug 92, FKK
allow parital line just before the EOF. Possibly this isn't the
right thing to do, but I decided to allow it. If the final
line
is incomplete, the values are still read and the remainder of
the line is filled with zeroes. 26 Oct 92, FKK
allow range keyword to be a string array, 2 Dec 92, FKK
make default for countallrows be true if range is present, 2 Dec
92, FKK
add new function (typeof); called in a few places, 2 Dec 92, FKK

--
Paul van Delst Ph: (301) 763-8000 x7274
CIMSS @ NOAA/NCEP Fax: (301) 763-8545
Rm.202, 5200 Auth Rd. Email: pvandelst@ncep.noaa.gov
Camp Springs MD 20746

Subject: Re: Reading in text data
Posted by R.Bauer on Wed, 09 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

reardonb@my-deja.com wrote:
>
> Hi. I am reading in text data (columns and rows of numbers) and I would
> like to know if there is a more elegant way of doing it. Currently, the
> user must specify how many columns there are. In my case the number of
> columns is manually inserted into the first line of the file like this:
>
> 3
> 0 1 2
> 1 2 3
> 2 3 4
> 3 4 5
> 4 5 6
> 5 6 7
> 6 7 8
> 7 8 9
> 8 9 10
> 9 10 11
>
> The attached procedure reads in the data. Is there a way to read in the
> data such that the user does not have to a priori know how many columns
> there are and such that IDL does not have to reserve a large amount of
> memory for the number of rows?
> Thanks.
> -Brian
>
>

Dear Brian,

this is our solution.
For further routines and copyright and licence.
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

The routine itselfs.
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_
html/dbase/download/read_data_file.tar.gz

The routine as loadable module.
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_ html/dbase/download/read_data_file.sav

It's not necessary to insert the line 3 command. This routine finds
itselfs start and end of the arrayed
data block. If you don't believe you can use the /assistant key.

You are able to read quite fast all ascii files data having
header - array table - trailer

e.g.

1)

TEST
time hno3 no2
2  2  5
5  8  9

---
2)

2  2  5
5  8  9

---

3)
HEADER
2  2  5
5  8  9
TRAILER

; Copyright (c) 1998, Forschungszentrum Juelich GmbH ICG-1
; All rights reserved.
; Unauthorized reproduction prohibited.
; 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.
;
;+
; USERLEVEL:
;   TOPLEVEL
;
; NAME:
;   read_data_file
;
; PURPOSE:

```
;   This function reads different ASCII files into a structure (array
orientated)
;
; CATEGORY:
;   DATAFILES
;
; CALLING SEQUENCE:
;   result=read_data_file(file_name)
;
; INPUTS:
;   file_name:  the filename to read in
;
; KEYWORD PARAMETERS:
;  integer: if set read type will be integer
;  long:    if set read type will be long
;  float:   if set read type will be float
;  double:  if set read type will be double
;  string:  if set read type will be string
; if no keyword_set read type is double
;  assistant: if set x_get_file is used to show the data file
;
; OUTPUTS:
; The result is if not keyword_set(string) a sstructure with this
tagnames:
;   FILE:          the file name of the ASCII file
;   separator      the used separator sign
;   DATA           the data array
;   COMMENTS       the number of comments
;   HEADER         the stringarray of comment lines before the data
;   TRAILER        the stringarray of comment lines behind the data
; If set type string a stringarray of the file is returned
;
; RESTRICTIONS:
; datasets which uses ';' as separator without a space behind like '; '
will not readed correctly
; possible separators: ' ', ',', ';'
;
; EXAMPLE:
;  result=read_data_file('dat.txt')
;  help,result,/str
;** Structure <135d458>, 5 tags, length=104, refs=1:
;   FILE           STRING    'dat.txt'
;   separator      STRING    ','
;   DATA           DOUBLE    Array[3, 3]
;   COMMENTS       LONG                 1
;   HEADER         STRING    Array[1]
;
; result=read_data_file('popjgr96.cs')
```

```
; help,result,/str
;  ** Structure <1358528>, 5 tags, length=28992, refs=1:
;   FILE            STRING    'popjgr96.cs'
;   separator       STRING    ' '
;   DATA            DOUBLE    Array[9, 401]
;   COMMENTS        LONG                 12
;   HEADER          STRING    Array[12]
;
; result=read_data_file('ghost.nas')
; help,result,/str
; ** Structure <15f32c8>, 5 tags, length=1696, refs=1:
;   FILE            STRING    'ghost.nas'
;   separator       STRING    ' '
;   DATA            DOUBLE    Array[3, 63]
;   COMMENTS        LONG                 20
;   HEADER          STRING    Array[20]
;
; result=read_data_file('bi21.txt')
; help,result,/str
; ** Structure <13584b8>, 5 tags, length=1776, refs=1:
;    FILE           STRING    'bi21.txt'
;    separator      STRING    ' '
;    DATA           DOUBLE    Array[16, 13]
;    COMMENTS       LONG                 11
;    HEADER         STRING    Array[11]
;
;
```