Subject: Re: How to read file to fill an array "partially" ?
Posted by ben.bighair on Wed, 17 Oct 2007 01:25:29 GMT
View Forum Message <> Reply to Message

On Oct 16, 5:19 pm, mystea <idllear...@gmail.com> wrote:
> Hello All,
>
> I am trying to read data from an ASCII file. The format of the file is
> as follows.
>
> "mydata.txt":
>
> 5
>
> 7.7
> 8.1
> 9.0
> 1.1
> 3.0
>
> 3
>
> 2.2
> 1.0
> 2.2
>
> Namely, the first line tells you how many entries are there, and the
> data follows. I plan to save them in a 2*5 array "myarray." So the
> following is what I am doing:
>
> openr, lun, 'mydata.txt', /get_lun ;open file
> myarray=dblarr(2,5) ;declare array to
> store data
> n_entries=0S ;declare variable
> to store number of entries
> for i=0, 1 do begin ;try to fill
> "myarray" by a for loop
> readf, lun, n_entries ;read in # of
> entries
> n_entries=fix(n_entries-1S) ;the final index of
> valid entry is # of entries minus one.
> readf, lun, myarray[i,0:n_entries] ;read data into
> myarray[i,0:# of entries -1]
> endfor
>
> However, it does not work! All I get is 0.00000 for all the entries I
> have. (data does not seem to be read)
> I tried to use format code, it doesn't help either. However, the

> terminal replies:
> % Attempt to store into an expression: <DOUBLE Array[5]>.
> so I guess IDL does not allow data to be read to stuff like
> myarray[0,0:4] because it is an "expression" instead of a real array.
>
> I have three simple questions:
> 1. How to read data and stored partially to an array?
> 2. Why the response "% Attempt to store into an expression" does not
> show up when I wasn't offering formats?
> 3. (might be the answer to the first question) How can I find the
> reference address of a certain portion of an array?
>
> P.S.Of course my data is much larger, I modified them to 2*5 in order
> to make the case clearer.

Hi,

Yes, you are right that IDL needs to read into a variable and not into an expression. David Fanning has a very good article on this - see ...

http://dfanning.com/tips/read_subscripted_array.html

But while you are at it, you mention that you have a lot of these. This might be a great chance to store each array as an object, and then store the collection of these array objects in an container. I have pasted below code that will do just that plus it will read in your file automatically.

Here's what a session in IDL might look like using these objects which i call "Bucket" which will hold an array and "MyBigBucket" will will hold a bunch of the little buckets.

```
IDL> o = obj_new("mybigbucket", "myarray.txt")
% Compiled module: MYBIGBUCKET__DEFINE.
Reading a 5 element array
0=7.7
1=8.1
2=9.0
3=1.1
4=3.0
Reading a 3 element array
0=2.2
1=1.0
2=2.2
IDL> x = o->Get(1)
IDL> help, x
```

```
X               OBJREF    = <ObjHeapVar62(BUCKET)>
IDL> print, x->get()
      2.20000      1.00000      2.20000
IDL> print, x->get(2)
      2.20000
```

And here's the code which you mst save in your search path as "mybigbucket__define.pro"

**BEGIN

```
;******
;   BUCKET is simply a pointer manager - acts as a bucket to place
anything
;******
; returns the number of elements in data
FUNCTION Bucket::Count
   return, self->Size(/N_ELEMENTS)
END
;returns results of SIZE call on data
FUNCTION Bucket::Size, $
   _EXTRA = extra
   IF PTR_VALID(self.pData) Then $
      RETURN, SIZE(*self.pData, _EXTRA = extra) Else $
      RETURN, SIZE(dummy, _EXTRA = extra)
END
;gets the pointer to the data (be careful)
FUNCTION Bucket::GetPointer
   Return, self.pData
END;GetPointer
;returns the data or the ith elements of data
FUNCTION Bucket::Get, i, COUNT = count
   count = PTR_VALID(self.pData)
   if count EQ 0 then Return, -1
   count = n_elements(*self.pData)
   if count EQ 0 then return, -1
   if n_elements(i) NE 0 then Begin
      d = (*Self.pData)[i]
      count = n_elements(d)
      return, d
   EndIf Else Begin
      return, *self.pData
   EndElse
END ;get
;sets the data
PRO Bucket::Set, data
   if PTR_VALID(self.Pdata) EQ 0 then $
```

```
        self.pData = PTR_NEW(data) Else $
        *self.pData = data
END
;init
FUNCTION Bucket::Init, data
    if n_elements(data) NE 0 then self->Set, data
    Return, 1
END
;cleanup
PRO Bucket::Cleanup
    PTR_FREE, self.pData
END
;definition
PRO Bucket__Define, struct
    struct = {Bucket, $
        pData: PTR_NEW()}
END

;*****************
; MyBigBucket - a container for the smallerbuckets
;*****************

PRO MyBigBucket::ReadFile, file

  if n_elements(file) EQ 0 then file = self.file

  testfile = FILE_SEARCH(file[0], COUNT = nFile)
  if nFile EQ 0 then message, 'File not found: ' + file[0]
  lines = (FILE_LINES(testFile[0]))[0]
  OPENR, LUN, testFile[0], /GET_LUN
  N = '0'
  DUMMY = ''
  Repeat Begin
    ReadF, LUN, N, format = '(A1)'
    READF, LUN, dummy, format = '(A1)'  ;this is blank line
    PRINT, 'Reading a ' + N + ' element array'
    arr = FLTARR(LONG(N))
    for i = 0, LONG(n)-1 do begin
      READF, LUN, dummy
      print, STRTRIM(i,2) + "="+dummy
      arr[i] = FLOAT(dummy)
    endfor
    self->Add, OBJ_NEW('Bucket', arr)
    if(eof(LUN) EQ 0) then $
      READF, LUN, dummy, format = '(A1)' else $ ;this is blank line
      BREAK
  EndRep Until (eof(LUN) EQ 1)
```

```
  FREE_LUN, LUN
  self.file = testFile[0]
END ;ReadFile

FUNCTION MyBigBucket::Get, pos, _REF_EXTRA = extra
  return, self->IDL_CONTAINER::Get(position = pos, _EXTRA = extra)
END
FUNCTION MyBigBucket::Init, file
  if self->IDL_Container::Init() EQ 0 then return, 0
  if n_elements(file) NE 0 then self->ReadFile, file
  return, 1
END

PRO MyBigBucket::Cleanup
  self->IDL_CONTAINER::Cleanup
END

PRO MyBigBucket__Define, struct
  struct = {MyBigBucket, $
    INHERITS IDL_CONTAINER, $
    FILE: ""}
END

**END
```

Cheers,
Ben

Subject: Re: How to read file to fill an array "partially" ?
Posted by mystea on Wed, 17 Oct 2007 18:26:39 GMT
View Forum Message <> Reply to Message

Hi Ben,

Thanks a lot for providing the link and your code.
I have two observations when I tried out David's ASCII reading tips.

1. the format code '(I0)' means integer of any length while '(A0)'
literally means string of length zero.
2. A string can't start with a number. for example, if you type
junk="3", IDL returns syntax error. The best one can do seems to be
junk=" 3".

Although I can swallow it and take them as simple facts, I don't feel
very comfortable about it. Why is there such inconsistency in the
design of format codes? And what can you do if you really want your
string starts with a number?

As to your bucket code, I need a little more time to digest. Do you
recommend any reference in IDL object programming?

Sincerely,

Gene

On Oct 16, 6:25 pm, "ben.bighair" <ben.bigh...@gmail.com> wrote:
> On Oct 16, 5:19 pm, mystea <idllear...@gmail.com> wrote:
>
>
>
>> Hello All,
>
>> I am trying to read data from an ASCII file. The format of the file is
>> as follows.
>
>> "mydata.txt":
>
>> 5
>
>> 7.7
>> 8.1
>> 9.0
>> 1.1
>> 3.0
>
>> 3
>
>> 2.2
>> 1.0
>> 2.2
>
>> Namely, the first line tells you how many entries are there, and the
>> data follows. I plan to save them in a 2*5 array "myarray." So the
>> following is what I am doing:
>
>> openr, lun, 'mydata.txt', /get_lun ;open file
>> myarray=dblarr(2,5) ;declare array to
>> store data
>> n_entries=0S ;declare variable
>> to store number of entries
>> for i=0, 1 do begin ;try to fill
>> "myarray" by a for loop
>> readf, lun, n_entries ;read in # of
>> entries
>> n_entries=fix(n_entries-1S) ;the final index of

>> valid entry is # of entries minus one.
>> readf, lun, myarray[i,0:n_entries] ;read data into
>> myarray[i,0:# of entries -1]
>> endfor
>
>> However, it does not work! All I get is 0.00000 for all the entries I
>> have. (data does not seem to be read)
>> I tried to use format code, it doesn't help either. However, the
>> terminal replies:
>> % Attempt to store into an expression: <DOUBLE Array[5]>.
>> so I guess IDL does not allow data to be read to stuff like
>> myarray[0,0:4] because it is an "expression" instead of a real array.
>
>> I have three simple questions:
>> 1. How to read data and stored partially to an array?
>> 2. Why the response "% Attempt to store into an expression" does not
>> show up when I wasn't offering formats?
>> 3. (might be the answer to the first question) How can I find the
>> reference address of a certain portion of an array?
>
>> P.S.Of course my data is much larger, I modified them to 2*5 in order
>> to make the case clearer.
>
> Hi,
>
> Yes, you are right that IDL needs to read into a variable and not into
> an expression. David Fanning has a very good article on this -
> see ...
>
> http://dfanning.com/tips/read_subscripted_array.html
>
> But while you are at it, you mention that you have a lot of these.
> This might be a great chance to store each array as an object, and
> then store the collection of these array objects in an container. I
> have pasted below code that will do just that plus it will read in
> your file automatically.
>
> Here's what a session in IDL might look like using these objects which
> i call "Bucket" which will hold an array and "MyBigBucket" will will
> hold a bunch of the little buckets.
>
> IDL> o = obj_new("mybigbucket", "myarray.txt")
> % Compiled module: MYBIGBUCKET__DEFINE.
> Reading a 5 element array
> 0=7.7
> 1=8.1
> 2=9.0
> 3=1.1

```
>  4=3.0
>  Reading a 3 element array
>  0=2.2
>  1=1.0
>  2=2.2
>  IDL> x = o->Get(1)
>  IDL> help, x
>  X               OBJREF    = <ObjHeapVar62(BUCKET)>
>  IDL> print, x->get()
>       2.20000      1.00000      2.20000
>  IDL> print, x->get(2)
>       2.20000
>
>  And here's the code which you mst save in your search path as
>  "mybigbucket__define.pro"
>
>  **BEGIN
>
>  ;******
>  ;   BUCKET is simply a pointer manager - acts as a bucket to place
>  anything
>  ;******
>  ; returns the number of elements in data
>  FUNCTION Bucket::Count
>     return, self->Size(/N_ELEMENTS)
>  END
>  ;returns results of SIZE call on data
>  FUNCTION Bucket::Size, $
>     _EXTRA = extra
>     IF PTR_VALID(self.pData) Then $
>        RETURN, SIZE(*self.pData, _EXTRA = extra) Else $
>        RETURN, SIZE(dummy, _EXTRA = extra)
>  END
>  ;gets the pointer to the data (be careful)
>  FUNCTION Bucket::GetPointer
>     Return, self.pData
>  END;GetPointer
>  ;returns the data or the ith elements of data
>  FUNCTION Bucket::Get, i, COUNT = count
>     count = PTR_VALID(self.pData)
>     if count EQ 0 then Return, -1
>     count = n_elements(*self.pData)
>     if count EQ 0 then return, -1
>     if n_elements(i) NE 0 then Begin
>        d = (*Self.pData)[i]
>        count = n_elements(d)
>        return, d
>     EndIf Else Begin
```

```
>         return, *self.pData
>     EndElse
>  END ;get
>  ;sets the data
>  PRO Bucket::Set, data
>     if PTR_VALID(self.Pdata) EQ 0 then $
>         self.pData = PTR_NEW(data) Else $
>         *self.pData = data
>  END
>  ;init
>  FUNCTION Bucket::Init, data
>     if n_elements(data) NE 0 then self->Set, data
>     Return, 1
>  END
>  ;cleanup
>  PRO Bucket::Cleanup
>     PTR_FREE, self.pData
>  END
>  ;definition
>  PRO Bucket__Define, struct
>     struct = {Bucket, $
>         pData: PTR_NEW()}
>  END
>
>  ;*****************
>  ; MyBigBucket - a container for the smallerbuckets
>  ;*****************
>
>  PRO MyBigBucket::ReadFile, file
>
>    if n_elements(file) EQ 0 then file = self.file
>
>    testfile = FILE_SEARCH(file[0], COUNT = nFile)
>    if nFile EQ 0 then message, 'File not found: ' + file[0]
>    lines = (FILE_LINES(testFile[0]))[0]
>    OPENR, LUN, testFile[0], /GET_LUN
>    N = '0'
>    DUMMY = ''
>    Repeat Begin
>      ReadF, LUN, N, format = '(A1)'
>      READF, LUN, dummy, format = '(A1)'  ;this is blank line
>      PRINT, 'Reading a ' + N + ' element array'
>      arr = FLTARR(LONG(N))
>      for i = 0, LONG(n)-1 do begin
>        READF, LUN, dummy
>        print, STRTRIM(i,2) + "="+dummy
>        arr[i] = FLOAT(dummy)
>      endfor
```

```
>      self->Add, OBJ_NEW('Bucket', arr)
>      if(eof(LUN) EQ 0) then $
>        READF, LUN, dummy, format = '(A1)' else $ ;this is blank line
>        BREAK
>    EndRep Until (eof(LUN) EQ 1)
>
>    FREE_LUN, LUN
>    self.file = testFile[0]
>  END ;ReadFile
>
>  FUNCTION MyBigBucket::Get, pos, _REF_EXTRA = extra
>    return, self->IDL_CONTAINER::Get(position = pos, _EXTRA = extra)
>  END
>  FUNCTION MyBigBucket::Init, file
>    if self->IDL_Container::Init() EQ 0 then return, 0
>    if n_elements(file) NE 0 then self->ReadFile, file
>    return, 1
>  END
>
>  PRO MyBigBucket::Cleanup
>    self->IDL_CONTAINER::Cleanup
>  END
>
>  PRO MyBigBucket__Define, struct
>    struct = {MyBigBucket, $
>      INHERITS IDL_CONTAINER, $
>      FILE: ""}
>  END
>
>  **END
>
>  Cheers,
>  Ben
```