Subject: Read an ASCII file in a table based on delimeter line string
Posted by burkina on Thu, 21 Jul 2016 15:36:53 GMT
View Forum Message <> Reply to Message

Hi everyone,

suppose I have a file like this one (there will be no regularity in the real ones!):

a 1
b 2
c 3
####
d 4
e 5
f 6
g 7
####
h 8
####

I want to read in IDL in separated arrays:

x[0]=[a,b,c]
x[1]=[d,e,f,g]
x[2]=[h]

y[0]=[1,2,3]
y[1]=[4,5,6,7]
y[2]=[8]

or a table t[2,8], or something equivalent.
In practice, I would like to use something like readcol but with IDL understanding the reading should stop every time it encounters a delimiter (#### in this case), and then start again separating the arrays for each set.

How can I do that?

Thanks,

Stefano

Subject: Re: Read an ASCII file in a table based on delimeter line string
Posted by Craig Markwardt on Thu, 21 Jul 2016 18:17:47 GMT
View Forum Message <> Reply to Message

On Thursday, July 21, 2016 at 11:36:56 AM UTC-4, burkina wrote:
> Hi everyone,

>
> suppose I have a file like this one (there will be no regularity in the real ones!):
...
> I want to read in IDL in separated arrays:
>
> x[0]=[a,b,c]
> x[1]=[d,e,f,g]
> x[2]=[h]
>
> y[0]=[1,2,3]
> y[1]=[4,5,6,7]
> y[2]=[8]
>
> or a table t[2,8], or something equivalent.
> In practice, I would like to use something like readcol but with IDL understanding the reading should stop every time it encounters a delimiter (#### in this case), and then start again separating the arrays for each set.

It's not going to be really easy, or efficient. It's not hard for someone who has programming experience, though.

Step one is deciding on what kind of data structure you want. The IDL arrays you selected do not support varying dimensions so you can't use that type of data structure. IDL lists might be a better approach, if you want to keep similar semantics.

Step two is deciding how to parse the data from an open file unit. Think of how you would do that as a human. The following script may help. I didn't test it in any way, so who knows if it will work.

```
openr, unit, 'myfile.dat', /get_lun
output = []  ;; Initialize output
while eof(unit) EQ 0 do begin
  ;; Initialize X and Y to empty arrays
  x = []
  y = []
  s = ''         ;; Initialize dummy string
  readf, unit, s ;; Read one line from input file

  ;; s is now one line from the input file

  ;; Begin parsing
  if s EQ '###' then begin  ;; It is a delimiter
    output = [output, {x: x, y: y}

  endif else begin ;; Or else parse it into data
    x1 = '' & y1 = 0L ;; Initialize dummy variables

    ;; NOTE: this will change based on the actual format of your data file
    reads, s, x1, y1, format='(A1,I)'
```

```
    ;; Append to existing list
    x = [x, x1]
    y = [y, y1]
  endelse

endwhile
free_lun, unit
```

Subject: Re: Read an ASCII file in a table based on delimeter line string
Posted by burkina on Fri, 22 Jul 2016 10:09:26 GMT
View Forum Message <> Reply to Message

On Thursday, 21 July 2016 20:17:50 UTC+2, Craig Markwardt wrote:
> On Thursday, July 21, 2016 at 11:36:56 AM UTC-4, burkina wrote:
>> Hi everyone,
>>
>> suppose I have a file like this one (there will be no regularity in the real ones!):
> ...
>> I want to read in IDL in separated arrays:
>>
>> x[0]=[a,b,c]
>> x[1]=[d,e,f,g]
>> x[2]=[h]
>>
>> y[0]=[1,2,3]
>> y[1]=[4,5,6,7]
>> y[2]=[8]
>>
>> or a table t[2,8], or something equivalent.
>> In practice, I would like to use something like readcol but with IDL understanding the reading should stop every time it encounters a delimiter (#### in this case), and then start again separating the arrays for each set.
>
> It's not going to be really easy, or efficient. It's not hard for someone who has programming experience, though.
>
> Step one is deciding on what kind of data structure you want. The IDL arrays you selected do not support varying dimensions so you can't use that type of data structure. IDL lists might be a better approach, if you want to keep similar semantics.
>
> Step two is deciding how to parse the data from an open file unit. Think of how you would do that as a human. The following script may help. I didn't test it in any way, so who knows if it will work.
>
> openr, unit, 'myfile.dat', /get_lun
> output = [] ;; Initialize output

```
>  while eof(unit) EQ 0 do begin
>    ;; Initialize X and Y to empty arrays
>    x = []
>    y = []
>    s = ''         ;; Initialize dummy string
>    readf, unit, s ;; Read one line from input file
>
>    ;; s is now one line from the input file
>
>    ;; Begin parsing
>    if s EQ '###' then begin  ;; It is a delimiter
>      output = [output, {x: x, y: y}
>
>    endif else begin ;; Or else parse it into data
>      x1 = '' & y1 = 0L ;; Initialize dummy variables
>
>      ;; NOTE: this will change based on the actual format of your data file
>      reads, s, x1, y1, format='(A1,I)'
>
>      ;; Append to existing list
>      x = [x, x1]
>      y = [y, y1]
>    endelse
>
>  endwhile
>  free_lun, unit
```

Great!
The key information was the suggestion to use IDL lists.
I modified your script, introducing lists for x and y, and now I have exactly what I wanted: each listx element can be easily plotted against its respective listy element, without knowing anything a priori on the array lengths.

Thanks!

Stefano
