
Subject: Unsolved indexing problem 2 weeks ago.
Posted by [kim20026](#) on Mon, 10 Sep 2007 02:00:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

G'day, Everyone!

I posted this question about 2 weeks ago, but I couldn't make it at that time. I was so urgent and I just finished with the way of the Old Stone Age. I spent several hours to finish up this things with MS Excel... T.T. However, I got a chance to do the same thing, I want to make it with IDL this time. Please give me any idea.

What I am trying to do now is to read in 16 lines at one time and compare the values of the second column for all 16 lines. Then, I either extract the one good value for output or I set the output to be unchanged. Once I've figured out what I want to output, I output all 16 lines at once and output the same value to the third column for each line.

This exmaple data file is simplified for test simulation. With this file, I am testing with 5 lines instead of 16.

```
aaa.txt
01 -999.9
02 -999.9
03 -999.9
04 0.13
05 -999.9
06 -999.9
07 0.17
08 -999.9
09 -999.9
10 -999.9
11 -999.9
12 -999.9
13 32.77
14 -999.9
15 -999.9
```

This is the array that I want to make.

```
01 -999.9 0.13
02 -999.9 0.13
03 -999.9 0.13
04 0.13 0.13
05 -999.9 0.13
06 -999.9 0.17
07 0.17 0.17
```

```
08 -999.9 0.17
09 -999.9 0.17
10 -999.9 0.17
11 -999.9 -999.9
12 -999.9 -999.9
13 32.77 32.77
14 -999.9 -999.9
15 -999.9 -999.9
```

I coded as shown below. However, As Conor pointed out 2 weeks ago, I am doing something different. I have changed several part of this code, but my trial has not been successful so far. Please give me any idea, recommendable functions, indexing tips, etc... Thanks.

Harry

```
-----
pro albedo_final
close, /all
data1 = 'D:\MODIS_ALL\aaa.txt'
num_data = file_lines(data1)
albedo_arr = fltarr(2, num_data)
albedo_fin = fltarr(3, num_data)
albedo_OK = 0.0

openr, 2, data1
readf, 2, albedo_arr
close, 2
c1 = 0

openw, 1, 'bbb.txt'
for i= 0, num_data-1 do begin

    dd = 5*(c1+1) +1
    if albedo_arr[0, i] lt DD then begin
        if (albedo_arr[1,i] gt 0 and albedo_arr[1,i] lt 1) then
begin
            albedo_OK = albedo_arr[1,i]
            print, albedo_OK
        endif
        albedo_fin[0:1, i] = albedo_arr[0:1, i]
        albedo_fin[2, i] = albedo_OK
    endif
    c1 = c1+1
endfor
print, albedo_fin
;printf, 1, albedo_fin
;close, 1
```

```
    print, " It's done!"
end
```

This is the last result.

1.00000	-999.900	0.000000
2.00000	-999.900	0.000000
3.00000	-999.900	0.000000
4.00000	0.130000	0.130000
5.00000	-999.900	0.130000
6.00000	-999.900	0.130000
7.00000	0.170000	0.170000
8.00000	-999.900	0.170000
9.00000	-999.900	0.170000
10.0000	-999.900	0.170000
11.0000	-999.900	0.170000
12.0000	-999.900	0.170000
13.0000	-999.900	0.170000
14.0000	-999.900	0.170000
15.0000	-999.900	0.170000

Subject: Re: Unsolved indexing problem 2 weeks ago.
Posted by [Brian Larsen](#) on Tue, 11 Sep 2007 15:14:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

As another solution, it is my style to pretty much always use `read_ascii()` and `ascii_template()` to read in data files. This has the advantage of making a structure for each column making indexing easier (at least for me). The down side of this is that you have to take the time to hard code (or save and recall) the output from `ascii_template()` but that is not too hard (I will put an example here as well).

This is a 14 column data file

```
doy.frac ut lt edge(1) edge(2) n_he Tperhe Tparhe
theta_he theta_hp n_lp effective_kp minkp[100ev] maxkp[50kev]
107.75080 18.01830 18.58810 0.000E+00 0.000E+00 9.623E-02
6.189E+02 7.768E+02 1.714E+02 8.068E+01 3.206E+00 2.300E+00
2.469E+00 1.003E+01
107.75180 18.04220 18.61200 0.000E+00 0.000E+00 1.133E-01
5.728E+02 4.715E+02 8.667E+01 7.945E+01 5.591E+00 2.300E+00
2.445E+00 9.961E+00
107.75280 18.06610 18.63590 0.000E+00 0.000E+00 1.316E-01
5.405E+02 4.477E+02 8.490E+01 8.091E+01 2.121E+00 2.300E+00
2.428E+00 9.920E+00
107.75380 18.09000 18.65970 0.000E+00 0.000E+00 1.560E-01
```

```

4.614E+02 3.598E+02 8.758E+01 8.014E+01 1.730E+00 2.300E+00
2.414E+00 9.881E+00
 107.75474 18.11390 18.68360 0.000E+00 0.000E+00 1.566E-01
5.184E+02 3.616E+02 8.700E+01 7.947E+01 1.704E+00 2.300E+00
2.403E+00 9.842E+00
 107.75573 18.13770 18.70750 2.941E+04 0.000E+00 1.572E-01
4.514E+02 3.012E+02 8.971E+01 8.268E+01 2.751E+00 8.317E+00
2.394E+00 9.803E+00

```

and the hard coded `ascii_template()` output is:

```

template = create_struct('version', 1.0, $
                        'datastart', 1l, $
                        'delimiter', 32b, $
                        'missingvalue', !values.f_nan, $
                        'commentsymbol', "", $
                        'fieldcount', 14l, $
                        'fieldtypes', lonarr(14), $
                        'fieldnames', strarr(14), $
                        'fieldlocations', lonarr(14), $
                        'fieldgroups', lonarr(14) )
template.fieldnames = ['doy', 'ut', 'lt1', 'edge1', 'edge2', $
                      'n_he', 'tperp_he', 'tpar_he', 'theta_he', $
                      'theta_hp', 'n_lp', 'eff_kp', 'minkp_100ev', $
                      'maxkp_50kev']
template.fieldtypes[*] = 4
template.fieldlocations =
[2,14,23,33,44,55,66,77,88,99,110,121,132,143]
template.fieldgroups = lindgen(14)

```

While annoying, you only have to do it once.

Cheers,

Brian

 Brian Larsen
 Boston University
 Center for Space Physics

Subject: Re: Unsolved indexing problem 2 weeks ago.
Posted by [Brian Larsen](#) on Mon, 01 Oct 2007 19:37:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

> While annoying, you only have to do it once.

All, I was writing a bunch of read_ascii wrappers today and got so annoyed at having to hard code in the ascii_template() output each time I wrote a code to do it for me...

It might be useful for you, and might not. You just run the code and then paste in the stuff between the SNIPS and that is your template for read_ascii().

[[Example]]

```
IDL> ascii_template_code
% Compiled module: ASCII_TEMPLATE_CODE.
% Compiled module: ASCII_TEMPLATE.
% Compiled module: QUERY_ASCII.
% Compiled module: XMANAGER.
;-----SNIP-----
template = create_struct('version',    1.00000, $
'datastart',    21l, $
'delimiter',    32b, $
'missingvalue', !values.f_nan, $
'commentsymbol', "", $
'fieldcount',    4l, $
'fieldtypes', lonarr(    4), $
'fieldnames', strarr(    4), $
'fieldlocations', lonarr(    4), $
'fieldgroups', lonarr(    4) )
template.fieldnames = [ $
'L', $
'PHI', $
'INTENSITY', $
'ERROR']
template.fieldtypes = [ $
    4, $
    4, $
    4, $
    4]
template.fieldlocations = [ $
    0, $
    15, $
    29, $
    43]
template.fieldgroups = [ $
    0, $
    1, $
```

```
2, $
3]
;-----SNIP-----
```

Cheers,

Brian

Brian Larsen
Boston University
Center for Space Physics

--- code ---

```
;+
; NAME:
; ascii_template_code
;
;
; PURPOSE:
; put the code needed to hard code an ascii_template() in a read
routine
;
;
; CATEGORY:
; coding help
;
; INPUTS:
; none
;
;
; OPTIONAL INPUTS:
; file - the file to make a template of (opt since it will ask)
;
;
; KEYWORD PARAMETERS:
; none
;
;
; OUTPUTS:
; code printed to the screen
;
```

```

;
;
; OPTIONAL OUTPUTS:
; none
;
;
; COMMON BLOCKS:
; none
;
;
; SIDE EFFECTS:
; none
;
;
; RESTRICTIONS:
; limited testing, but has worked in each case I have tested on IDL6.4
;
;
;
; EXAMPLE:
; I cant come up with a simple one...
; IDL> ascii_template_code
; template = create_struct('version',    1.00000, $
; 'datastart',    1l, $
; 'delimiter',    32b, $
; 'missingvalue', !values.f_nan, $
; 'commentsymbol', "", $
; 'fieldcount',    2l, $
; 'fieldtypes', lonarr(    2), $
; 'fieldnames', strarr(    2), $
; 'fieldlocations', lonarr(    2), $
; 'fieldgroups', lonarr(    2) )
; template.fieldnames = [ $
; 'PHI', $
; 'FIELD2']
; template.fieldtypes = [ $
;    4, $
;    4]
; template.fieldlocations = [ $
;    6, $
;    18]
; template.fieldgroups = [ $
;    0, $
;    1]
;
;
;
; MODIFICATION HISTORY:
;

```

```
; Mon Oct 1 15:26:51 2007, Brian Larsen
; <balarsen@bu.edu>
;
;
; written and tested
;
;-
```

pro ascii_template_code

```
temp = ascii_template()
print, ';-----SNIP-----'
print, "template = create_struct('version'," + string(temp.version) +
', $'
print, "datastart'," + string(temp.datastart)+'l, $'
print, "delimiter'," + string(fix(temp.delimiter))+ 'b, $'
if not finite(temp.missingvalue) then $
    print, "missingvalue', !values.f_nan, $" $
else $
    print, "missingvalue'," + string(temp.missingvalue) + ', $'
print, "commentsymbol', ""+ temp.commentsymbol + "", $"
print, "fieldcount'," + string(temp.fieldcount) + 'l, $'
print, "fieldtypes', lonarr(""+string(temp.fieldcount)+'), $'
print, "fieldnames', strarr(""+string(temp.fieldcount)+'), $'
print, "fieldlocations', lonarr(""+string(temp.fieldcount)+'), $'
print, "fieldgroups', lonarr(""+string(temp.fieldcount)+') )'
print, "template.fieldnames = [ $"
for i=0l, temp.fieldcount-2 do $
    print, "" + temp.fieldnames[i] + "", $"
print, "" + temp.fieldnames[i] + "]"
print, "template.fieldtypes = [ $"
for i=0l, temp.fieldcount-2 do $
    print, string(temp.fieldtypes[i]) + ", $"
print, string(temp.fieldtypes[i]) + "]"

print, "template.fieldlocations = [ $"
for i=0l, temp.fieldcount-2 do $
    print, string(temp.fieldlocations[i]) + ", $"
print, string(temp.fieldlocations[i]) + "]"

print, "template.fieldgroups = [ $"
for i=0l, temp.fieldcount-2 do $
    print, string(temp.fieldgroups[i]) + ", $"
print, string(temp.fieldgroups[i]) + "]"
print, ';-----SNIP-----'
```


return
end
