Subject: Re: Comma seperators
Posted by John-David T. Smith on Mon, 22 May 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:
>
> Ben Tupper wrote:
>>
>> Paul van Delst wrote:
>>
>>> Simon de Vet wrote:
>>>>
>>>> I am reading in data that looks like the following:
>>>>
>>>> CHATHAM ISLAND - NEW ZEALAND (DOE),,,,,,,,,,
>>>> 43.92ï¿½S,176.50ï¿½W,,,,,,,,,
>>>> 16-Sep-1983,11-Oct-1996,,,,,,,,,
>>>> Mon,Stat,Cl,NO3,SO4,Na ,SeaSalt,nssSO4,MSA,Dust,NH4
>>>> of,Param,Air,Air,Air,Air,Air,Air,Air,Air,Air
>>>> Yr,*,ï¿½g/m3,ï¿½g/m3,ï¿½g/m3,ï¿½g/m3,ï¿½g/m3,
ï¿½g/m3,ï¿½g/m3,ï¿½g/m3,ï¿½g/m3
>>>> Jan,N,58,58,58,58,58,57,0,0,58
>>>> Jan,Mean,7.330,0.120,1.572,4.233,13.766,0.508,#N/A,#N/A,0.10 3
>>>> Jan,StdDev,2.788,0.055,0.412,1.479,4.811,0.249,#N/A,#N/A,0.0 51
>>>>
>>>> Which continues untill the end of the year, and then another observation
>>>> station follows the fame general format.
>>>>
>>>> I want to be able to read in the data into an array. I can already take
>>>> out the header, but I cannot read in the data.
>>>
>>> What do you consider the header?
>>>
>>>> By default, IDL is
>>>> treating each line as one entry, not recognizing the commas as entry
>>>> seperators. I've read the help extensively, but as a non-fortran user,
>>>> the input format documentation makes my brane hurt.
>>>
>>> Let's say you have:
>>>
>>> Jan,N,58,58,58,58,58,57,0,0,58
>>> Jan,Mean,7.330,0.120,1.572,4.233,13.766,0.508,#N/A,#N/A,0.10 3
>>> Jan,StdDev,2.788,0.055,0.412,1.479,4.811,0.249,#N/A,#N/A,0.0 51
>>> Feb,N,58,58,58,58,58,57,0,0,58
>>> Feb,Mean,7.330,0.120,1.572,4.233,13.766,0.508,#N/A,#N/A,0.10 3
>>> Feb,StdDev,2.788,0.055,0.412,1.479,4.811,0.249,#N/A,#N/A,0.0 51
>>> ..etc..
>>>

>>> How about:
>>>
>>> char_buffer = ' '
>>>
>>> REPEAT BEGIN
>>>   READF, lun, char_buffer
>>>
>>>   input_data = STR_SEP(char_buffer, ',')
>>>
>>>   ....here split up the data how you want by, say, testing
>>>   input_data[0] == month (Jan, Feb, Mar, ...
>>>   input_data[1] == data type (N, Mean, StdDev)
>>>   ....and checking for invalid data, e.g. the #N/A thingoes
>>>
>>> ENDREP UNTIL EOF(lun)
>>>
>>>
>>
>> Hello,
>>
>> I'ld like to add that on occasion, I have found it useful to add the /TRIM
>> keyword to the STR_SEP() function.
>> Once in a while the last element in input_data will become something
>> unexpected, such as the expected value padded with blanks.   I think
>> the problem is in how the file was written, not in how it is read by IDL.
>
> You know, the same thought occurred to me when I used this method to
> read *space*-separated data - I kept getting extra "fields" at the
> beginning of my string. I stuck the /TRIM keyword in the STRSEP call and
> nothing changed!!?? Weird.
>
> So instead of doing a
>
> result = STRSEP(string, ' ', /TRIM)
>
> I do a
>
> result = STRSEP(STRTRIM(string, 2), ' ')
>
> Mind you this was one of those cases where something didn't work
> straight up and I spent precisely 0.1seconds figuring out why not before
> going on to something else.. :o)
>
> BTW, is there some sequence of layered string function calls one can use
> to trim and "collapse" a string with multiple delimiters between items
> to a single delimiter? e.g. to convert
>
> ,,,this,,,is,,,,a,,multiple,,,,,delimited,,,,,,,,string,,,,

>
> to
>
> this,is,a,multiple,delimited,string
>
> I wrote a function to do it but it has a loop in it and a bunch of logic
> checking that looks horrendous. It does the job, but no reason why it
> can't look pretty....right?
>

```
res=strsplit(str,',',/EXTRACT)
```

will do it. The reason is null-length fields are *not* returned unless you use PRESERVE_NULL. You can also split on regular expressions. So, e.g. if you could be delimited by one or more spaces or commas, you could use:

```
res=strsplit(str,'[ ,]+',/REGEX,/EXTRACT)
```

This is mostly v5.3 specific.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-6263
304 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|