Subject: Re: Question about 'READ input error'
Posted by Richard French on Wed, 21 Mar 2001 04:24:20 GMT
View Forum Message <> Reply to Message

> My idl code is:
> ------------------------------
>
> lun=13
> aaa=fltarr(2,180,51)
> spawn,'/usr/ucb/uncompress -c'+'~anilk/Nimbus7/GriddedData/grid_EP_79134.dat.Z',unit=lu n
> readf,lun,aaa
> print, aaa
> close,lun
>

Don't you need a space after the -c?
Dick

Subject: Re: Question about 'READ input error'
Posted by akk on Wed, 21 Mar 2001 05:34:14 GMT
View Forum Message <> Reply to Message

>> readf,lun,aaa
>> print, aaa
>> close,lun
>>
>
> Don't you need a space after the -c?
> Dick
>
Sorry I doubled checked my code now, I actually have the space. I must
have mistyped when I sent the email
Thanks
Anil
>

Subject: Re: Question about 'READ input error'
Posted by Martin Schultz on Wed, 21 Mar 2001 08:52:44 GMT
View Forum Message <> Reply to Message

Anil Kochhar wrote:
>
>>> readf,lun,aaa
>>> print, aaa
>>> close,lun

>>>
>>
>> Don't you need a space after the -c?
>> Dick
>>
> Sorry I doubled checked my code now, I actually have the space. I must
> have mistyped when I sent the email
> Thanks
> Anil
>>

Anil,

    an input conversion error is - in short - an error that occurs
when IDL tries to interpret ASCII characters as numerical values. The
easiest way to demonstrate this is to use ReadS which works exactly
like ReadF but uses input from a string rather than a file:

```
IDL> s=' 1. 2. 3. 4.'
IDL> res=fltarr(4)
IDL> reads,s,res
IDL> print,res
      1.00000      2.00000      3.00000      4.00000
IDL> s=' 1. 2. 3. C'
IDL> reads,s,res
% READS: Input conversion error. Unit: 0, File: <stdin>
% Execution halted at:  $MAIN$
```

So, the error occurs, because Read expects a number but receives a
character ('C'). Now it's up to you to find out where that "format
error" in the ASCII file you are trying to read occurs. NIMBUS sounds
like a pretty standardized product - with other ASCII files I
sometimes find a ********* (indicating an overflow in a formatted
output). Different line endings (DOS/Windows vs. Unix or Mac) I think,
were also a problem until IDL 5.4.

    Following up on a recent discussion here, I think it would be nice
to have a routine that reads ASCII files efficiently but safely as
well. Probably, a 2-3 step approach would be best:
(1) try to read the whole block of data at once (this is fastest)
(2) scan the file line by line if (1) fails
Step (2) could be split into (2a): try to read each line "directly",
(2b) read each line into a string first, then use ReadS.
This routine would handle the simpler file structures where (at least
after a certain number of comment lines) you only have numerical data.
For more convoluted file structures, one needs to go with the
read-into-structure approach.

This is something that has been on my list for over a year now;
but I am happy these days if I find any time for IDL at all.

Cheers,

Martin

--
[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[