Subject: Forcing READ_ASCII output to be a set of strings...
Posted by Jonathan Greenberg on Mon, 14 Mar 2005 17:56:07 GMT
View Forum Message <> Reply to Message

I'm trying to use READ_ASCII to read in a text file which contains parameters for the algorithm I'm developing (so the user just modifies the .csv file instead of having to type all the parameters into the idl command line). It appears to be defaulting to an array of floating points in the structure, but I want the values to be read in as text (they contain things like paths to certain file, etc). How do I "force" READ_ASCII to bring those values in as an array of strings?

--j

Subject: Re: Forcing READ_ASCII output to be a set of strings...
Posted by Michael Wallace on Mon, 14 Mar 2005 20:56:22 GMT
View Forum Message <> Reply to Message

> Second, why should the default be string rather than float? Just
> because you have a need to use strings for this particular application
> doesn't mean that RSI should change their defaults. For default values,
> a floating point number is the best assumption to make. If the default
> values don't work for you, use a template. This is why templates exist...

And by the way, you want to do:

```
template = ascii_template()
data = read_ascii('myfile.csv', TEMPLATE = template)
```

If you need to use the template multiple times, just save it off somewhere and restore it in your code whenever you need it.

-Mike

Subject: Re: Forcing READ_ASCII output to be a set of strings...
Posted by Jonathan Greenberg on Mon, 14 Mar 2005 21:34:48 GMT
View Forum Message <> Reply to Message

I actually kicked a backwards compatible version of this to the IDL help, and also uploaded it to the RSI user contrib website. You are right, I probably should have renamed the file (we'll see if IDL bounces the upload and asks me to rename it), but here are the mods I made to the "official" read_ascii release:

***

Function call mod:

```
function read_ascii_string, $
   file, $            ; IN:
   RECORD_START=recordStart, $     ; IN: (opt)
   NUM_RECORDS=numRecords, $       ; IN: (opt)
   TEMPLATE=template, $        ; IN: (opt)
   DATA_START=dataStart, $     ; IN: (opt)
   DELIMITER=delimiter, $      ; IN: (opt)
   MISSING_VALUE=missingValue, $   ; IN: (opt)
   COMMENT_SYMBOL=commentSymbol, $ ; IN: (opt)
;    FIELDS=fields, $           ; IN: (opt)    [not implemented]
   VERBOSE=verbose, $          ; IN: (opt)
   HEADER=header, $            ; OUT: (opt)
   COUNT=count, $              ; OUT: (opt)
; **** JONATHAN'S MOD ****
   DATA_TYPE=data_type         ; IDL data type (opt)
```

Modifying the default data_type without needing to use a template:

```
; Keeps the default to floating point
   if n_elements(data_type) eq 0 then data_type=4
   fieldTypesUse   = REPLICATE(data_type, fieldCountUse)
```

***

This would cause zero problems, as far as I can tell, if a user just swapped this in (you'll note that I kept the default a floating point).

The reason I didn't want to use a template was I didn't see a quick way to make it apply a string format to an arbitrarily long ascii file (e.g. If I add a new column to my DB, don't I have to modify the ascii template each time?)

The reason I thought this would be better as a text default is that you can easily go from string -> number, but you can't go backwards. Considering the input is text, why would they just assume it is filled with floating point numbers? It appears to be completely arbitrary... A string format appears to be the most "generalized" form you could use.

--j

On 3/14/05 12:51 PM, in article 113bua0b4hdv2a2@corp.supernews.com, "Michael Wallace" <mwallace.no.spam@no.spam.swri.edu.invalid> wrote:

> Jonathan Greenberg wrote:
>> I kinda answered my own question -- I took the read_ascii.pro that IDL

>> distributes and simply changed the default behavior on this line:
>>
>> fieldTypesUse = REPLICATE(4L, fieldCountUse)
>>
>> To
>>
>> fieldTypesUse = REPLICATE(7L, fieldCountUse)
>>
>> This appears to work fine... I'm kicking IDL a request to modify this so
>> it'll default to a user-defined format...
>>
>
> Um, this is dangerous. First, if you ever want to distribute this
> application, or at least give it to someone else, you'll have to make
> the same modification in their IDL install. It can be *very* problematic.
>
> Second, why should the default be string rather than float? Just
> because you have a need to use strings for this particular application
> doesn't mean that RSI should change their defaults. For default values,
> a floating point number is the best assumption to make. If the default
> values don't work for you, use a template. This is why templates exist...
>
> -Mike

Subject: Re: Forcing READ_ASCII output to be a set of strings...
Posted by Michael Wallace on Mon, 14 Mar 2005 22:15:43 GMT
View Forum Message <> Reply to Message

> The reason I thought this would be better as a text default is that you can
> easily go from string -> number, but you can't go backwards. Considering
> the input is text, why would they just assume it is filled with floating
> point numbers? It appears to be completely arbitrary... A string format
> appears to be the most "generalized" form you could use.

The reason you assume that it's filled with numbers is because it most likely is. (Don't you just love circular logic? ;-)) IDL is built around *data* manipulation. We use text files all the time in my industry and aside from the header, if present, the contents are all numbers. Some of the numbers are integers and some are floating point. Obviously, a floating point type covers both cases. Text files have the nice advantage in that you can look at the contents of the file without needing to load the files into any program. Just because we happen to store data within a text file doesn't mean that it should be interpreted as text.

I'm not just talking about one particular project here, but we create ASCII representations of a lot of our data on many of our projects.

Sure, it's not the primary form we store the data in for the long term, but it's great for our scientists ad hoc work, especially since our scientists seem to be allergic to netCDF, CDF, HDF and the like.

Another issue is that it'd be terrible performance-wise to read in a bunch of values only to convert them to numeric types. And most of the time, your CSV or other files will be numeric in nature. Having strings in there is a special case. Maybe the change should be to make templates easier to modify rather than change the fundamental nature of read_ascii(). I just don't buy the explanation that the majority of folks will have ASCII files of text rather than ASCII files of numbers. You can try to convince me otherwise, but I don't know that you'll get very far. ;-)

I feel pretty dirty now that I'm actually defending something in the language rather than railing against something in the language, which is my typical. ;-)

-Mike

Subject: Re: Forcing READ_ASCII output to be a set of strings...
Posted by Karsten Rodenacker on Tue, 15 Mar 2005 09:04:37 GMT
View Forum Message <> Reply to Message

Humm, interesting debate
Why not:

```
function read_s_ascii_new, na, count=i
  openr,lun,na,/get_lun
  i=0
  s=''
  while ~eof(lun) do begin
    readf, lun,s
    r=i eq 0 ? s : [temporary(r),s]
    i++
  endwhile
  free_lun,lun
  return,i eq 0 ? -1l : r
end
```

By the way read_s_ascii was an old routine from I don't know whom.
Regards
Karsten

On Mon, 14 Mar 2005 14:56:22 -0600, Michael Wallace
<mwallace.no.spam@no.spam.swri.edu.invalid> wrote:

>> Second, why should the default be string rather than float? Just
>> because you have a need to use strings for this particular application

>> doesn't mean that RSI should change their defaults. For default
>> values, a floating point number is the best assumption to make. If the
>> default values don't work for you, use a template. This is why
>> templates exist...
>
> And by the way, you want to do:
>
> template = ascii_template()
> data = read_ascii('myfile.csv', TEMPLATE = template)
>
> If you need to use the template multiple times, just save it off
> somewhere and restore it in your code whenever you need it.
>
> -Mike

--
Karsten Rodenacker
---------- -------- :-)
GSF - Forschungszentrum Institute of Biomathematics and Biometry
D-85758 Oberschleissheim Postfach 11 29
Karsten.Rodenacker@gsf.de | http://ibb.gsf.de/
http://ibb.gsf.de/homepage/karsten.rodenacker/
Tel: +49 89 31873401 | FAX: ..3369

Subject: Re: Forcing READ_ASCII output to be a set of strings...
Posted by JD Smith on Fri, 18 Mar 2005 23:11:46 GMT
View Forum Message <> Reply to Message

On Mon, 14 Mar 2005 16:15:43 -0600, Michael Wallace wrote:

>> The reason I thought this would be better as a text default is that you can
>> easily go from string -> number, but you can't go backwards. Considering
>> the input is text, why would they just assume it is filled with floating
>> point numbers? It appears to be completely arbitrary... A string format
>> appears to be the most "generalized" form you could use.
>
> The reason you assume that it's filled with numbers is because it most
> likely is. (Don't you just love circular logic? ;-)) IDL is built
> around *data* manipulation. We use text files all the time in my
> industry and aside from the header, if present, the contents are all
> numbers. Some of the numbers are integers and some are floating point.
> Obviously, a floating point type covers both cases.

Actually, it doesn't. This is an aside, but you may have wondered how
a little old floating point number which fits in just four bytes could

manage to squeeze all those numbers from 1.17549e-38 to 3.40282e+38 in there, when a 4 byte integer (aka LONG) only holds from -2147483648 to +2147483647, and no decimals, to boot. The secret is, it doesn't. Not by a long shot. Consider the range of 1 trillion integers:

10000000000000000000ULL - 10000001000000000000ULL

How many unique floating point value do you have inside of this range? None actually. Any such value will be rounded up or down outside of this range. So if you are using integers to count the quantity of sand grains in a sandpile, this may not matter much, but if you are using them to represent an exact serial code or combination to your bank vault, then this distinction is very important.

JD
