

[View Forum Message](#) <> [Reply to Message](#)

> Dave

```
; -1.875000000 78.875000000 1.317
; -1.875000000 78.625000000 1.284
; -1.875000000 78.375000000 1.216
; -1.875000000 78.125000000 1.148
; -1.875000000 77.875000000 1.080
```

```
infile = "data-in.txt"
n = FILE_LINES(infile)
s = FLTARR(3, n)
OPENR, U, infile, /GET_LUN
READF, U, s
FREE_LUN, u
```

```
outfile = "data-out.txt"
OPENW, U, outfile, /GET_LUN
PRINTF, U, s
FREE_LUN, U
```

; data-out.txt looks like the following

```
; -1.87500 78.8750 1.31700
; -1.87500 78.6250 1.28400
; -1.87500 78.3750 1.21600
; -1.87500 78.1250 1.14800
; -1.87500 77.8750 1.08000
```

Subject: Re: Writing in text file
Posted by [d.poreh](#) on Mon, 27 Sep 2010 11:48:15 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Sep 27, 4:16 am, Ben Tupper <ben.bigh...@gmail.com> wrote:

> On 9/27/10 5:05 AM, Dave Poreh wrote:

>

>

>

>

>

>> Folks

>> Hi;

>> I read some data like this:

>> -1.875000000 78.875000000 1.317

>> -1.875000000 78.625000000 1.284

```

>> -1.8750000000 78.3750000000 1.216
>> -1.8750000000 78.1250000000 1.148
>> -1.8750000000 77.8750000000 1.080
>> .....
>> And when I want to write it,
>> openw,1,'C.dat'
>> z=transpose(reform([data[0,*],data[1,*],data[2,*]],n_elements(data[2,*]),
>> 3))
>> printf,1,z
>> close,1
>> it gives me something like this:
>>   -1.87500   0.897000   71.6250
>>    78.8750   9.87500   0.645000
>>    1.31700   70.3750   21.8750
>>   -1.87500   1.01300   71.3750
>>    78.6250   9.87500   0.538000
>>    1.28400   70.1250   21.8750
>> .....
>> Will you please help me out whit this?
>> Cheers
>> Dave
>
> Hi Dave,
>
> You haven't said what you expected to see, but clearly the transpose is
> making this come out 'funny'. Here's what I see when I simply readf
> from a file ("data-in.txt") and then printf the data to a file
> ("data-out.txt").
>
> Cheers,
> Ben
> { x86_64 darwin unix Mac OS X 7.1 Apr 21 2009    64    64}
>
> ;data-in.txt looks like the following
>
> ;-1.8750000000 78.8750000000 1.317
> ;-1.8750000000 78.6250000000 1.284
> ;-1.8750000000 78.3750000000 1.216
> ;-1.8750000000 78.1250000000 1.148
> ;-1.8750000000 77.8750000000 1.080
>
> infile = "data-in.txt"
> n = FILE_LINES(infile)
> s = FLTARR(3, n)
> OPENR, U, infile, /GET_LUN
> READF, U, s
> FREE_LUN, u
>

```

Thanks Ben. Your way is perfect. I don't know why my way does not
working?
Cheers
Dave

[illegible]

```

>>>    -1.87500    0.897000    71.6250
>>>     78.8750    9.87500    0.645000
>>>     1.31700    70.3750    21.8750
>>>    -1.87500    1.01300    71.3750
>>>     78.6250    9.87500    0.538000
>>>     1.28400    70.1250    21.8750
>>>         ĩĉ½ĩĉ½ĩĉ½.
>>> Will you please help me out whit this?
>>> Cheers
>>> Dave
>>
>> Hi Dave,
>>
>> You haven't said what you expected to see, but clearly the transpose is
>> making this come out 'funny'. Here's what I see when I simply readf
>> from a file ("data-int.txt") and then printf the data to a file
>> ("data-out.txt").
>>
>> Cheers,
>> Ben
>> { x86_64 darwin unix Mac OS X 7.1 Apr 21 2009    64    64}
>>
>> ;data-in.txt looks like the following
>>
>> ;-1.8750000000 78.8750000000 1.317
>> ;-1.8750000000 78.6250000000 1.284
>> ;-1.8750000000 78.3750000000 1.216
>> ;-1.8750000000 78.1250000000 1.148
>> ;-1.8750000000 77.8750000000 1.080
>>
>> infile = "data-in.txt"
>> n = FILE_LINES(infile)
>> s = FLTARR(3, n)
>> OPENR, U, infile, /GET_LUN
>> READF, U, s
>> FREE_LUN, u
>>
>> outfile = "data-out.txt"
>> OPENW,U, outfile,/GET_LUN
>> PRINTF, U, s
>> FREE_LUN, U
>>
>> ; data-out.txt looks like the following
>>
>> ;   -1.87500    78.8750    1.31700
>> ;   -1.87500    78.6250    1.28400
>> ;   -1.87500    78.3750    1.21600
>> ;   -1.87500    78.1250    1.14800

```

```
>> ;   -1.87500    77.8750    1.08000
```

```
>
```

```
> Thanks Ben. Your way is perfect. I don't know why my way does not
```

```
> working?
```

```
> Cheers
```

```
> Dave
```

Hi again,

I know that when I get bogged down with this kind of stuff (happens often*) using a tiny simple example helps - I like to use BINDGEN to create an ordered easy-to-read array. Below, you can see that I separate the reformed array (rs) from the transposed reformed array (trs). See how reform inserts the data into the new array in the original order? Note that the top row in rs is ordered 0,1,2,3,4. Does the rs value printed look like what you expected? Then also note that down the first column the order is 0, 5, 10. When you transpose it back to being a 3x5 array the top row becomes 0,5,10. Thus your woes begin.

```
IDL> s = bindgen(3,5)
```

```
IDL> rs = reform([s[0,*],s[1,*],s[2,*]],n_elements(s[2,*]),3)
```

```
IDL> trs = transpose(rs)
```

```
IDL> print, s
```

```
 0  1  2
 3  4  5
 6  7  8
 9 10 11
12 13 14
```

```
IDL> print, rs
```

```
 0  1  2  3  4
 5  6  7  8  9
10 11 12 13 14
```

```
IDL> print, trs
```

```
 0  5 10
 1  6 11
 2  7 12
 3  8 13
 4  9 14
```

Cheers,

Ben

* Especially now that I use R a lot. It's either row- or column- major. I can't remember, but I know it is whatever IDL isn't.

Subject: Re: Writing in text file
Posted by [rogass](#) on Tue, 28 Sep 2010 18:13:06 GMT
[View Forum Message](#) <> [Reply to Message](#)

On 27 Sep., 15:18, Ben Tupper <ben.bigh...@gmail.com> wrote:

> On 9/27/10 7:48 AM, Dave Poreh wrote:

>
>
>
>
>
>

>> On Sep 27, 4:16 am, Ben Tupper<ben.bigh...@gmail.com> wrote:

>>> On 9/27/10 5:05 AM, Dave Poreh wrote:

>

>>>> Folks

>>>> Hi;

>>>> I read some data like this:

>>>> -1.8750000000 78.8750000000 1.317

>>>> -1.8750000000 78.6250000000 1.284

>>>> -1.8750000000 78.3750000000 1.216

>>>> -1.8750000000 78.1250000000 1.148

>>>> -1.8750000000 77.8750000000 1.080

>>>>

>>>> And when I want to write it,

>>>> openw,1,'C.dat'

>>>> z=transpose(reform([data[0,*],data[1,*],data[2,*]],n_elements(data[2,*]),
>>>> 3))

>>>> printf,1,z

>>>> close,1

>>>> it gives me something like this:

>>>> -1.87500 0.897000 71.6250

>>>> 78.8750 9.87500 0.645000

>>>> 1.31700 70.3750 21.8750

>>>> -1.87500 1.01300 71.3750

>>>> 78.6250 9.87500 0.538000

>>>> 1.28400 70.1250 21.8750

>>>>

>>>> Will you please help me out whit this?

>>>> Cheers

>>>> Dave

>

>>> Hi Dave,

>

>>> You haven't said what you expected to see, but clearly the transpose is

>>> making this come out 'funny'. Here's what I see when I simply readf

>>> from a file ("data-int.txt") and then printf the data to a file

>>> ("data-out.txt").

>

>>> Cheers,

```

>>> Ben
>>> { x86_64 darwin unix Mac OS X 7.1 Apr 21 2009    64    64}
>
>>> ;data-in.txt looks like the following
>
>>> ;-1.8750000000 78.8750000000 1.317
>>> ;-1.8750000000 78.6250000000 1.284
>>> ;-1.8750000000 78.3750000000 1.216
>>> ;-1.8750000000 78.1250000000 1.148
>>> ;-1.8750000000 77.8750000000 1.080
>
>>> infile = "data-in.txt"
>>> n = FILE_LINES(infile)
>>> s = FLTARR(3, n)
>>> OPENR, U, infile, /GET_LUN
>>> READF, U, s
>>> FREE_LUN, u
>
>>> outfile = "data-out.txt"
>>> OPENW,U, outfile,/GET_LUN
>>> PRINTF, U, s
>>> FREE_LUN, U
>
>>> ; data-out.txt looks like the following
>
>>> ;   -1.87500    78.8750    1.31700
>>> ;   -1.87500    78.6250    1.28400
>>> ;   -1.87500    78.3750    1.21600
>>> ;   -1.87500    78.1250    1.14800
>>> ;   -1.87500    77.8750    1.08000
>
>> Thanks Ben. Your way is perfect. I don't know why my way does not
>> working?
>> Cheers
>> Dave
>
> Hi again,
>
> I know that when I get bogged down with this kind of stuff (happens
> often*) using a tiny simple example helps - I like to use BINDGEN to
> create an ordered easy-to-read array. Below, you can see that I
> separate the reformed array (rs) from the transposed reformed array
> (trs). See how reform inserts the data into the new array in the
> original order? Note that the top row in rs is ordered 0,1,2,3,4. Does
> the rs value printed look like what you expected? Then also note that
> down the first column the order is 0, 5, 10. When you transpose it back
> to being a 3x5 array the top row becomes 0,5,10. Thus your woes begin.
>

```

```
> IDL> s = bindgen(3,5)
> IDL> rs = reform([s[0,*],s[1,*],s[2,*]],n_elements(s[2,*]),3)
> IDL> trs = transpose(rs)
> IDL> print, s
>   0  1  2
>   3  4  5
>   6  7  8
>   9 10 11
>  12 13 14
> IDL> print, rs
>   0  1  2  3  4
>   5  6  7  8  9
>  10 11 12 13 14
> IDL> print, trs
>   0  5 10
>   1  6 11
>   2  7 12
>   3  8 13
>   4  9 14
>
> Cheers,
> Ben
>
> * Especially now that I use R a lot. It's either row- or column- major.
> I can't remember, but I know it is whatever IDL isn't.
```

Yes, reform exactly does what it shall do.

CR
