Subject: renaming a variable without making a copy
Posted by David Higgins on Tue, 08 Dec 2009 11:51:25 GMT
View Forum Message <> Reply to Message

Can I rename a variable to a more approrpriate name, instead of making
a copy of the variable

I'm using /OVERWRITE to save memory, through a number of operations on
a large multi-dimensional array. I'd like to keep my code readable and
rename the variable to reflect what the data currently are.

Thanks,
Dave

Subject: Re: renaming a variable without making a copy
Posted by lecacheux.alain on Thu, 10 Dec 2009 10:31:34 GMT
View Forum Message <> Reply to Message

On 9 déc, 18:06, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:

> read about reads or execute
>

Reimar,
I understand that 'reads' might allow a string to be read out as a
structure by using a binary format, but such a typecast method seems
to me quite a bit involved.
I do not understand the usage of 'execute' for such a thing.
alain.

Subject: Re: renaming a variable without making a copy
Posted by R.Bauer on Thu, 10 Dec 2009 15:36:34 GMT
View Forum Message <> Reply to Message

alx schrieb:
> On 9 dï¿½c, 18:06, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
>
>> read about reads or execute
>>
>
> Reimar,
> I understand that 'reads' might allow a string to be read out as a
> structure by using a binary format, but such a typecast method seems
> to me quite a bit involved.

the "s" it is not only about strings

```
IDL> a=findgen(2,2)
IDL> b=dblarr(4)
IDL> reads,a,b
IDL> print,b
      0.0000000       1.0000000       2.0000000       3.0000000
```

btw. just seen that reads, 1 seems to be valid, but what does it do?

> I do not understand the usage of 'execute' for such a thing.
> alain.

Subject: Re: renaming a variable without making a copy
Posted by lecacheux.alain on Thu, 10 Dec 2009 16:30:48 GMT
View Forum Message <> Reply to Message

On 10 déc, 16:36, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
> alx schrieb:
>
> IDL> a=findgen(2,2)
> IDL> b=dblarr(4)
> IDL> reads,a,b
> IDL> print,b
>   0.0000000   1.0000000   2.0000000   3.0000000
>
> btw. just seen that reads, 1 seems to be valid, but what does it do?
>

I did not realize so far that 'reads' could actually mean "read from
memory". Thanks for the tip !
Here is an immediate application: you want to import some structure
from a COM object, written in C or whatsoever.
But the IDL COM bridge does not support passing of structure. So you
need to import it as a typecasted (variant) array
of simple type. Then, "reads" will allow to recover and use the
structure in IDL...
alain.

Subject: Re: renaming a variable without making a copy
Posted by lecacheux.alain on Thu, 10 Dec 2009 16:56:39 GMT
View Forum Message <> Reply to Message

On 10 déc, 17:30, alx <lecacheux.al...@wanadoo.fr> wrote:

> On 10 déc, 16:36, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
>
>> alx schrieb:
>
>> IDL> a=findgen(2,2)
>> IDL> b=dblarr(4)
>> IDL> reads,a,b
>> IDL> print,b
>> 0.0000000 1.0000000 2.0000000 3.0000000
>
>> btw. just seen that reads, 1 seems to be valid, but what does it do?
>
>
Unfortunately, that does not work as you said: "reads" (cf. IDL doc)
converts
your 'a' array to an array of strings, then 'read' it to 'b' by using
implicit formatting.
That is actually not a "memory read" and, likely, does not work with
structures.

Subject: Re: renaming a variable without making a copy
Posted by R.Bauer on Fri, 11 Dec 2009 11:51:38 GMT
View Forum Message <> Reply to Message

alx schrieb:
> On 10 dï¿½c, 17:30, alx <lecacheux.al...@wanadoo.fr> wrote:
>> On 10 dï¿½c, 16:36, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:
>>
>>> alx schrieb:
>>> IDL> a=findgen(2,2)
>>> IDL> b=dblarr(4)
>>> IDL> reads,a,b
>>> IDL> print,b
>>> 0.0000000 1.0000000 2.0000000 3.0000000
>>> btw. just seen that reads, 1 seems to be valid, but what does it do?
>>
> Unfortunately, that does not work as you said: "reads" (cf. IDL doc)
> converts
> your 'a' array to an array of strings, then 'read' it to 'b' by using
> implicit formatting.
> That is actually not a "memory read" and, likely, does not work with
> structures.
>
>

what is that doing then?

```
IDL> a=bindgen(3)
IDL> b=ptr_new({s:intarr(3)})
IDL> reads,a,*b
IDL> print,*b
{       0       1       2
```

Subject: Re: renaming a variable without making a copy
Posted by lecacheux.alain on Fri, 11 Dec 2009 13:22:33 GMT
View Forum Message <> Reply to Message

On 11 déc, 12:51, Reimar Bauer <R.Ba...@fz-juelich.de> wrote:

> what is that doing then?

> IDL> a=bindgen(3)
> IDL> b=ptr_new({s:intarr(3)})
> IDL> reads,a,*b
> IDL> print,*b
> {   0   1   2

Still the same: 'a' is implicitely converted to some string array by
'reads', then read out to *b.
Look at the following:

define some structure:
```
IDL> a={n:1L,x:2.0}
```

build its binary content as a byte chain:
```
IDL> b=[byte(a.n,0,4),byte(a.x,0,4)]
IDL> print,b
  1   0   0   0   0   0   0  64
```

define a void structure to get the result:
```
IDL> aa={n:0L,x:0.0}
```

then :
```
IDL> reads,b,aa
IDL> print,aa
{          1     0.000000}
```
does not provide the right result.

Subject: Re: renaming a variable without making a copy
Posted by penteado on Fri, 11 Dec 2009 16:38:43 GMT
View Forum Message <> Reply to Message

On Dec 11, 11:22 am, alx <lecacheux.al...@wanadoo.fr> wrote:
> Look at the following:
>
> define some structure:
> IDL> a={n:1L,x:2.0}
>
> build its binary content as a byte chain:
> IDL> b=[byte(a.n,0,4),byte(a.x,0,4)]
> IDL> print,b
> 1 0 0 0 0 0 0 64
>
> define a void structure to get the result:
> IDL> aa={n:0L,x:0.0}
>
> then :
> IDL> reads,b,aa
> IDL> print,aa
> { 1 0.000000}
> does not provide the right result.

Good example. In case anybody is wondering why that happens, the reads is getting the first field of a from the string "1", and the second field from the string "0" (the string representations of the first and second elements of the byte array b).

To properly obtain aa from the bytes in b, it is necessary to know exactly how the fields are aligned, as David mentioned. Which in this particular case would be:

IDL> aa.n=long(b,0)
IDL> aa.x=float(b,4)
IDL> print,aa
{ 1 2.00000}

Somebody also talked about using (real) pointers in IDL. Ronn Kling's DLM book has a useful trick (by Nigel Wade), to use a memory copy to hold a pointer's value in an IDL byte array of the proper size.