
Subject: Re: type reform

Posted by [bill.dman](#) on Wed, 04 Jul 2007 20:25:21 GMT

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

On Jul 4, 2:07 pm, greg.a...@googlemail.com wrote:

- > Is there any way to reform an array so that its type changes? I want
- > to read a file format which contains a mixture of 2 and 4-byte
- > integers. Rather than pre-build complicated structures to read it, it
- > would be nice to read the whole thing as 2-byte values, cut it up as
- > necessary, and then 'reform' the necessary blocks into 4-byte integers
- > with no processing cost.
- >
- > Essentially a way to convert an INT = Array[100] into a LONG =
- > Array[50] ?
- >
- > Greg

Yes, look at the documentation for LONG(), FIX(), FLOAT(), and the other type conversion routines.

It may not be obvious that when a second argument is supplied, these will perform a bit pattern preserving type cast operation similar to that in the C language.

Your case for example:

```
I = indgen(100)
L = long(I, 0, 50)
```

A couple of other examples on a little-endian host (Intel X86 processor):

```
IDL> b = [0B,0B,128B, 63B]
IDL> print, float(b)
    0.00000  0.00000  128.000  63.0000
IDL> print, float(b,0)
    1.00000
IDL> print, byte(1.0,0,4)
    0  0 128 63
```

Be careful with byte order if you want your code to be portable to all platforms supported by IDL!

-Bill

Subject: Re: type reform

Posted by [greg.addr](#) on Thu, 05 Jul 2007 11:22:39 GMT

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

Thanks, Bill. Yes, I can see the byte ordering is going to be fun!

Greg

Subject: Re: type reform

Posted by [greg.addr](#) on Fri, 06 Jul 2007 09:57:53 GMT

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

On Jul 4, 10:25 pm, bill.d...@gmail.com wrote:

> On Jul 4, 2:07 pm, greg.a...@googlemail.com wrote:

>

>> Is there any way to reform an array so that its type changes? I want
>> to read a file format which contains a mixture of 2 and 4-byte
>> integers. Rather than pre-build complicated structures to read it, it
>> would be nice to read the whole thing as 2-byte values, cut it up as
>> necessary, and then 'reform' the necessary blocks into 4-byte integers
>> with no processing cost.

>

>> Essentially a way to convert an INT = Array[100] into a LONG =
>> Array[50] ?

>

>> Greg

>

> Yes, look at the documentation for LONG(), FIX(), FLOAT(), and the
> other type conversion routines.

> It may not be obvious that when a second argument is supplied, these
> will perform a bit pattern preserving type cast operation similar to
> that in the C language.

>

> Your case for example:

> I = indgen(100)

> L = long(I, 0, 50)

>

> A couple of other examples on a little-endian host (Intel X86
> processor):

> IDL> b = [0B,0B,128B, 63B]

> IDL> print, float(b)

> 0.00000 0.00000 128.000 63.0000

> IDL> print, float(b,0)

> 1.00000

> IDL> print, byte(1.0,0,4)

> 0 0 128 63

>

> Be careful with byte order if you want your code to be portable to all
> platforms supported by IDL!

> -Bill

That worked, but now I met a new problem. If I read the data as bytes
and try to do something like this to cut off a section to convert to

another type:

```
subset=data[0:n-1,*]
```

then I run out of memory (when data= \sim 1/2 GB). I suppose IDL is building a list of indices (4 bytes each?) for every element to copy. Is there any more efficient way to do this?

Greg
