Subject: Re: Coding for speed help needed
Posted by wmc on Thu, 29 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

In article r3r@gazette.bcm.tmc.edu, pford@bcm.tmc.edu (Patrick V. Ford) writes:
> I need a FAST method of decoding a series of bytes to floats. The byte pattern encodes a range of numbers
> from MIN to MAX. I am attempting to code this as an call_external routine in C, but it is buggy and has not worked yet. The
> basic C routine looks this:
>
>
...cuts...
> kappa = (byte[0] << 24) | (byte[1] << 16) | (byte[2] << 8) | byte[3];
> /* Note: endian is irrelvant to the algorithm   */
>
> range = *max - *min;
> *result = range/((float)ULONG_MAX) * (float)kappa + *min;
>
> What I would like to do, is do this in IDL where the result would go into A,
> where
>
> A = fltarr(64,64)
>
> and the byte array is
>
> B = bytarr(64*64*4)
>

so why not:

```
i=indgen(64*64)*4
a=b(i)*2l^24+b(i+1)*2l^16+b(i+2)*2l^8+b(i+3)
a=a*kappa+min    ; or some other scaling
```

its possible you might prefer your shifts to the 2l^24 etc -
I don't know which is faster.

> with a minumum of for loops.

is none few enough? ;-)

- William

William M Connolley | wmc@bas.ac.uk | http://www.nbs.ac.uk/public/icd/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself

Subject: Re: Coding for speed help needed
Posted by Liam Gumley on Fri, 30 May 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Patrick V. Ford wrote:
> pop_size = 36; this can vary greatly
> dim = 64
> B = bytarr(dim * dim * 5 * pop_size)
> There are pop_size individuals
> The first 64*64 (dim*dim) bytes code for the activity or intensity in a 64
> by 64 image. The next 4*64*64 bytes code for the attenuation coefficients
> which are
> between 0.0 and 1.0, where 1.0 is no attenuation and 0.0 is 100%.

Patrick,

Are you familiar with structures in IDL? Any time you have to read mixed datatypes from a datafile, structures are usually the easiest way to do it. For example, let's say that on disk you have (in one contiguous file),

- a byte array of size 64x64
- a float array of size 64x64

then to read it, you would do something like this:

```
openr, lun, 'input.dat', /get_lun
data = { array1 : bytarr( 64, 64 ), array2 : fltarr( 64, 64 ) }   ;
structure definition
readu, lun, data    ; read the structure from disk
free_lun, lun
print, data.array1( *, 0 )    ; print the first 64 elements of the byte
array
print, data.array2( *, 0 )    ; print the first 64 elements of the float
array
```

Then if you need to swap byte order, all you have to do is

```
data = swap_endian( data )
```

which will take care of all the data types in the structure.

Cheers,
Liam.