Subject: Re: Float procedure
Posted by eddie haskell on Wed, 02 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Charlie Solomon wrote:
>
> Can anyone shed some light on how this byte array is converted into a
> floating point number?
> two_words = bytarr(4)
> two_words = [244, 232, 165, 64]

first off, the second assignment statement makes two_words an intarr,
not a bytarr as intended.
to keep two_words as a bytarr do something like:
two_words = byte([244,232,165,64])  -or maybe-
two_words[*] = [244, 232, 165, 64]

> IDL> print, float(two_words, 0)
>  2.13062e-038

when you use the offset in the float procedure, IDL takes the first 32
bits it finds from the point
specified in the offset, in this case, the beginning of the array.

you can see this if you look at the binary representation of the numbers
(i.e. using kevin ivory's
binary program):

```
IDL> print,binary(float(two_words,0))
  0   0   0   0   0   0   0   0   1   1   1   1   0   1   0   0
  0   0   0   0   0   0   0   0   1   1   1   0   1   0   0   0
IDL> print,binary(244),binary(232)
  0   0   0   0   0   0   0   0   1   1   1   1   0   1   0   0
  0   0   0   0   0   0   0   0   1   1   1   0   1   0   0   0
```

as to the difference between machines, that could be a big/little endian
thing but i really don't
know for sure (can anybody verify this?)

cheers,
eddie

----- ---- --- ----  --- --- ---  --- -- --  -- - -  -  -
|\  A. G. Edward Haskell
|\  Center for Coastal Physical Oceanography
|\  Old Dominion University, Norfolk VA  23529
|\  Voice 757.683.4816  Fax 757.683.5550
|\  e-mail  haskell@ccpo.odu.edu

----- ---- --- ----  --- --- ---  --- -- --  -- - -  -  -

Subject: Re: Float procedure
Posted by Liam Gumley on Wed, 02 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Charlie Solomon wrote:
> Can anyone shed some light on how this byte array is converted into a
> floating point number?
> two_words = bytarr(4)
> two_words = [244, 232, 165, 64]
> IDL> print, two_words, format = '(z)'
>  f4
>  e8
>  a5
>  40
> IDL> print, float(two_words, 0)
>  2.13062e-038
>
> On my computer at work (NT4, x86) I get a different value...5.18469
> Here at home (Win98, x86) I get this really small number....
> IDL> print, !version
> { x86 Win32 Windows 5.1.1 Jul 20 1998}

In addition to Alex Schuster's comments, the only suggestion I have is
to use the HELP routine to keep track of variable types, e.g.

```
IDL> two_words = bytarr(4)
IDL> help, two_words
TWO_WORDS       BYTE      = Array[4]
IDL> two_words = [244, 232, 165, 64]
IDL> help, two_words
TWO_WORDS       INT       = Array[4]
```

One of the most important things to learn in IDL programming is the
transient nature of variable types. Frequent use of HELP both on the
command line *and* within your IDL programs is a very effective way to
help ensure your IDL code works the way you intended.

Cheers,
Liam.

---
Liam E. Gumley
Space Science and Engineering Center, UW-Madison
1225 W. Dayton St., Madison WI 53706, USA
Phone (608) 265-5358, Fax (608) 262-5974

Subject: Re: Float procedure
Posted by Alex Schuster on Wed, 02 Dec 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Charlie Solomon wrote:

> Can anyone shed some light on how this byte array is converted into a
> floating point number?
> two_words = bytarr(4)
> two_words = [244, 232, 165, 64]

Caution! This statements creates an intarr(4). You could define
two_words this way:

IDL> two_words = [244b, 232b, 165b, 64b]

> IDL> print, float(two_words, 0)
> 2.13062e-038

I get this result:

IDL> f = float( two_words, 0 )
IDL> print, f
-1.47457e+32

This still isn't the value you expect, because your PC uses another byte
ordering scheme than my workstation. Changing this with the swap_endian
routine corrects this:

IDL> print, swap_endian( f )
5.18469

Alex
--
Alex Schuster Wonko@weird.cologne.de PGP Key available
alex@pet.mpin-koeln.mpg.de