Subject: hexadecimal variables
Posted by Pavel Romashkin on Wed, 21 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi,
Is there a nice way to convert hexadecimal variables? Here's what I am
trying to do:

This is received from the outside.
    h_string = '40000FC0DA'

    h_long = long(h_string)
This, obviously, fails to recognize alphabetical hexadecimal part and
sets h_long to 40,000.

    temp = execute("h_long = long(' "+h_string+" 'x)")
This, naturally, works, setting h_long to the desired 1032410, but looks
quite ugly. Besides, it is 3 times slower than without "execute" - not
really a concern on fast machines, but inefficient code just bugs me.

I am sure I am missing something here. Can anyone suggest a neater way
to do this? Maybe, there is a way to use Z-formatted READ here?

Thank you,
Pavel

Subject: Re: hexadecimal variables
Posted by Vapuser on Thu, 22 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> writes:

> Hi,
> Is there a nice way to convert hexadecimal variables? Here's what I am
> trying to do:
>
> This is received from the outside.
>   h_string = '40000FC0DA'
>
>   h_long = long(h_string)
> This, obviously, fails to recognize alphabetical hexadecimal part and
> sets h_long to 40,000.
>
>   temp = execute("h_long = long(' "+h_string+" 'x)")
> This, naturally, works, setting h_long to the desired 1032410,

  This is not what h_long equals; 1032410 = 'fc0da'x. You've only gotten

the first 32 bits. See below.

> but looks
> quite ugly. Besides, it is 3 times slower than without "execute" - not
> really a concern on fast machines, but inefficient code just bugs me.
>
> I am sure I am missing something here. Can anyone suggest a neater way
> to do this? Maybe, there is a way to use Z-formatted READ here?
>
> Thank you,
> Pavel
>

40000FC0DA is larger than 32 bits. Unless you're using idl 5.2, having access to 64 bit integers, you're out of luck, without kluging together something having the same flavor as a 'far' pointer in old MS-DOS.

The general method, however, is to use reads, which works on scalars and arrays. Here's an example, using IDL 5.2, and 64 bit integers.

```
IDL> h='40000FC0DA'
IDL> x=0LL
IDL> help,x
X               LONG64    =                    0
IDL> reads,h,x,form='(Z)'
IDL> print,x,form='(z)'
           40000fc0da
IDL> print,x
         274878939354
IDL>
```

If you try this in idl<5.2,(with x=0L) you'll only get the right most 32 bits, i.e. X = 'fc0da'x. Witness...

```
IDL> x1=0L
IDL> reads,h,x1,form='(Z)'
IDL> print,x1
     1032410
IDL> print,x1,form='(z)'
       fc0da
```

If you really need the > 32 bits, you're going to have to use idl 5.2, or read it, split it, and recombine it as a double?

--
William Daffer

vapuser@catspaw.jpl.nasa.gov

Subject: Re: hexadecimal variables
Posted by Pavel Romashkin on Thu, 22 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Dick,
I appreciate it! It is certainly what I looked for, and I was looking for READS yesterday before I wrote a question. But, apparently quite dumb by the end of the day, I typed "READ" in the search string, naively thinking that all READ-related functions will be there. READ/READF surely are, but READS happened to be separate. And, after using IGOR and IDL at the same time, I thought I rememberd seeing READS among IGOR's functions.
However, after I tried READS, it is even slower than "execute" (by 0.02 s but still)! That is strange. I guess compiling a string takes less time than string conversion. READS might appear more efficient if a string to pass to "execute" was more complicated.

Dick Jackson wrote:

> You've got it, using ReadS (read from string). It looks like you have five
> bytes so you need a very long integer to store this, giving me the first
> chance I've had to use IDL 5.2's new data types:

Well, in fact I have 8 bytes (in the real data) but I am checking only on bits 12 and 13, so even INT will do. If I need to check on higher level bits, I can cut the string apart.
Thanks again,
Pavel

Subject: Re: hexadecimal variables
Posted by wmc on Fri, 23 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> wrote:
> This is received from the outside.
> h_string = '40000FC0DA'

> h_long = long(h_string)
> This, obviously, fails to recognize alphabetical hexadecimal part and
> sets h_long to 40,000.

> temp = execute("h_long = long(' "+h_string+" 'x)")
> This, naturally, works, setting h_long to the desired 1032410

As others have pointed out, its a 64 bit string.

Looking at your "execute", I thought that

wmc> h_long = long(h_string+''x)
Ought to work, but it produces
wmc> print,h_long
   -2556760
(I'd forgotten that its a 64-bit string)

However,
wmc> h_long = long64(h_string+''x)
Seems to work perfectly well
wmc> print,h_long
        4398043954346

- 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: hexadecimal variables
Posted by wmc on Mon, 26 Apr 1999 07:00:00 GMT
View Forum Message <> Reply to Message

wmc@bas.ac.uk wrote:
> Pavel Romashkin <promashkin@cmdl.noaa.gov> wrote:
>>   h_string = '40000FC0DA'

> wmc> h_long = long64(h_string+''x)
> Seems to work perfectly well
> wmc> print,h_long
>   4398043954346

Oops, I didn't check this properly. I just looked at the above & thought it
was about right, but IDL has been fooling me:

I thought the string+''x syntax ought to work (why doesn't it?) but it doesn't,
it just doesn't always generate an error message. Oh dear. In fact:

wmc> h_string='10' & h_long = long(h_string+''x) & print,h_long
   -2596734
wmc> h_string='10' & h_long = long(h_string+''xl) & print,h_long

h_string='10' & h_long = long(h_string+'
                              ^

% Long integer constant must be less than 2147483648.

I'm confused..., but I suspect IDL is too...

-W

btw: { alpha OSF unix 5.2 Oct 30 1998}

--
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

---