
Subject: print, long(1.0e10)
Posted by [Gary Fu](#) on Tue, 03 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

I got different results from the following example on SGI IRIX and PC
Linux:

```
a = 1.0e10
b = long(a)
print, b      ; 2147483647 for IRIX, -2147483648 for Linux
```

It looks like that IRIX and Linux handle it differently when assigning a floating value greater (or less) than the boundary of a long integer to a long variable. Should I always check the range of 'a' before assigning it to a 'long' variable to make the result consistent on IRIX and Linux ?

TIA,

Gary

--

```
*****
* Gary Fu, GSC (301) 286-7107          *
* email : "gfu@seadas.gsfc.nasa.gov"  *
* NASA/Goddard Space Flight Center    *
*****
```

Subject: Re: print, long(1.0e10)
Posted by [thompson](#) on Wed, 04 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Gary Fu <gfu@seadas.gsfc.nasa.gov> writes:

> Hi,

> I got different results from the following example on SGI IRIX and PC
> Linux:

```
> a = 1.0e10
> b = long(a)
> print, b      ; 2147483647 for IRIX, -2147483648 for Linux
```

> It looks like that IRIX and Linux handle it differently when assigning a
> floating value greater (or less) than the boundary of a long integer to
> a long variable. Should I always check the range of 'a' before
> assigning it to a 'long' variable to make the result consistent on IRIX

> and Linux ?

I get yet a different result with IDL/v5.2 on Digital Unix.

```
IDL> a = 1.0e10
IDL> b = long(a)
% Program caused arithmetic error: Floating illegal operand
% Detected at $MAIN$
IDL> print, b
1410065408
```

This is a completely sensible result, since it's equal to $(A \bmod 2^{32})$, i.e. the 32 lowest most bits of A expressed as an integer.

Since different machines seem to act in different ways, it's best to catch the error, either by testing the value beforehand, or by using something like CHECK_MATH.

William Thompson

Subject: Re: print, long(1.0e10)
Posted by [Liam Gumley](#) on Wed, 04 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Gary Fu wrote:

```
> I got different results from the following example on SGI IRIX and PC
> Linux:
> a = 1.0e10
> b = long(a)
> print, b ; 2147483647 for IRIX, -2147483648 for Linux
>
> It looks like that IRIX and Linux handle it differently when assigning a
> floating value greater (or less) than the boundary of a long integer to
> a long variable. Should I always check the range of 'a' before
> assigning it to a 'long' variable to make the result consistent on IRIX
> and Linux ?
```

Gary,

I guess it depends on what you are trying to do. If you want to interpret the binary representation of 1.0e10 as a 32-bit signed LONG, then the syntax is

```
result = long(1.0e10, 0)
```

However if you want to represent 1.0e10 as an integer value, I think the only option (other than a range check) is to use IDL 5.2, which allows

64-bit signed and unsigned integers, e.g.

```
result = long64(1.0e10)
```

Cheers,
Liam.

--

Liam E. Gumley
Space Science and Engineering Center, UW-Madison
<http://cimss.ssec.wisc.edu/~gumley>

Subject: Re: print, long(1.0e10)
Posted by [luthi](#) on Mon, 09 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hold on

```
> I got different results from the following example on SGI IRIX and PC
> Linux:
> a = 1.0e10
> b = long(a)
> print, b      ; 2147483647 for IRIX, -2147483648 for Linux
```

The same commands evaluated with PV-Wave (6.21, Solaris 2.6) gives

```
% Program caused arithmetic error: Floating divide by 0
% Program caused arithmetic error: Floating underflow
% Program caused arithmetic error: Floating illegal operand
% Detected at $MAIN$ (LONG).
print,b
1343554297
```

While I don't understand the "divide by 0" error at least there is some error message. Is this due to the wonderful world of IEEE numerics?

M. Luethi

--

```
=====
Martin Luethi  Tel. +41 1 632 40 92
Glaciology Section Fax. +41 1 632 11 92
VAW ETH Zuerich
CH-8092 Zuerich  mail luthi@vaw.baum.ethz.ch
Switzerland
=====
```

Subject: Re: print, long(1.0e10)
Posted by [thompson](#) on Mon, 09 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Gary Fu <gfu@seadas.gsfc.nasa.gov> writes:

> I got different results from the following example on SGI IRIX and PC
> Linux:
> a = 1.0e10
> b = long(a)
> print, b ; 2147483647 for IRIX, -2147483648 for Linux

> It looks like that IRIX and Linux handle it differently when assigning a
> floating value greater (or less) than the boundary of a long integer to
> a long variable. Should I always check the range of 'a' before
> assigning it to a 'long' variable to make the result consistent on IRIX
> and Linux ?

It appears that Digital Unix gives yet another answer:

```
IDL> a = 1.0e10
IDL> b = long(a)
% Program caused arithmetic error: Floating illegal operand
IDL> print, b
1410065408
```

which makes sense, since that's A modulo 2^{32} .

William Thompson

Subject: Re: print, long(1.0e10)
Posted by [R.Bauer](#) on Tue, 10 Aug 1999 07:00:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

Gary Fu wrote:

> Hi,
>
> I got different results from the following example on SGI IRIX and PC
> Linux:
> a = 1.0e10
> b = long(a)
> print, b ; 2147483647 for IRIX, -2147483648 for Linux
>

Results of WIN NT

```
IDL> a=1.0e10
IDL> b = long(a)
IDL> print, b
1410065408
```

Results of AIX

```
IDL> a=1.0e10
IDL> b=long(a)
% Program caused arithmetic error: Floating illegal operand
% Program caused arithmetic error: Conversion to integer error
IDL> print,b
2147483647
```

R.Bauer
