Subject: Re: array subscript conversion
Posted by Michael Galloy on Wed, 25 Jul 2007 16:09:52 GMT
View Forum Message <> Reply to Message

Subject: Re: array subscript conversion
Posted by Dick Jackson on Wed, 25 Jul 2007 17:46:49 GMT
View Forum Message <> Reply to Message

Hi,

```
 news:Pine.LNX.4.64.0707251721430.28405@bifur.rmki.kfki.hu...
>  Hi guys,
>
>  according to the manual, array subscripts are converted to long (or long64 on
>  64 bit systems) before use if necessary, so an explicit conversion should not
>  affect the result.
>
>  IDL> print, !version
>  { x86 linux unix linux 6.3 Mar 23 2006      32      64}
>  IDL>
>  IDL> a=lindgen(10)
>  IDL> print, a[[long(-1ull)]]
>            0
>  IDL> print, a[[-1ull]]
>            9
>
>  Is it a bug or I am missing something?
```

I think you're expecting -1ull to be negative, but the 'u' in 'ull' means 'unsigned'. What you end up with instead of -1 is the largest 64-bit integer (this is a nice shortcut when it's actually what you want to do!):

```
IDL> help,-1ull
<Expression>    ULONG64   =   18446744073709551615
```

That's why a[[-1ull]] gives 9, as the subscript is (somewhat) larger than the maximum index in the array a. It is similar to a[[11]] below...

```
IDL> help,long(-1ull)
<Expression>    LONG      =           -1
```

This is similar to a[[-1]] below...

```
IDL> print,a[[-1]]
```

```
           0
IDL> print,a[[11]]
           9
```

And we know this is different from simple subscripting which doesn't allow
out-of-range values:

```
IDL> print,a[-1]
Attempt to subscript A with <INT      (      -1)> is out of range.
Execution halted at: $MAIN$
IDL> print,a[11]
Attempt to subscript A with <INT      (      11)> is out of range.
Execution halted at: $MAIN$
```

--
Cheers,
-Dick

--
Dick Jackson Software Consulting   http://www.d-jackson.com
Victoria, BC, Canada   +1-250-220-6117   dick@d-jackson.com

Subject: Re: array subscript conversion
Posted by Foldy Lajos on Wed, 25 Jul 2007 18:04:10 GMT
View Forum Message <> Reply to Message

On Wed, 25 Jul 2007, mgalloy@gmail.com wrote:

```
>  On Jul 25, 9:36 am, FÖLDY Lajos <fo...@rmki.kfki.hu> wrote:
>>  Hi guys,
>>
>>  according to the manual, array subscripts are converted to long (or long64
>>  on 64 bit systems) before use if necessary, so an explicit conversion
>>  should not affect the result.
>>
>>  IDL> print, !version
>>  { x86 linux unix linux 6.3 Mar 23 2006      32      64}
>>  IDL>
>>  IDL> a=lindgen(10)
>>  IDL> print, a[[long(-1ull)]]
>>             0
>>  IDL> print, a[[-1ull]]
>>             9
>>
>>  Is it a bug or I am missing something?
>>
>>  regards,
```

>> lajos
>
> All I see in the manual about converting subscripts to longs is:
>
> "Subscripts can be any type of vector or scalar expression. If a
> subscript expression is not integer, a longword integer copy is made
> and used to evaluate the subscript."
>
> around the middle of this page:
>
> http://idlastro.gsfc.nasa.gov/idl_html_help/Understanding_Ar ray_Subscripts.html
>
> By "integer", I think they mean the more general any integer type:
> byte, integer, long, etc. And in your case, the type is "integer", so
> no conversion is made.
>
> I am I missing another statement in the help that says something more
> explicit?
>
> Mike
> --
> www.michaelgalloy.com
>

You are right. I remembered "not long" instead of "not integer". And also remembered IDL_MEMINT from idl_export.h, which represents memory offsets and sizes, and assumed that IDL always uses IDL_MEMINT for subscripting internally.

Second try:

```
IDL> a=lindgen(10)
IDL>
IDL> print, a[4294967296ll]
           0
IDL> print, a[[4294967296ll]]
           9
```

In the first case the scalar subscript is "integer", so no conversion is needed. But it is converted to LONG (=0). Why?

regards,
lajos

Subject: Re: array subscript conversion

Posted by Foldy Lajos on Wed, 25 Jul 2007 18:26:47 GMT

View Forum Message <> Reply to Message

On Wed, 25 Jul 2007, Dick Jackson wrote:

```
> Hi,
>
> "FÖLDY Lajos" <foldy@rmki.kfki.hu> wrote in message
>  news:Pine.LNX.4.64.0707251721430.28405@bifur.rmki.kfki.hu...
>> Hi guys,
>>
>> according to the manual, array subscripts are converted to long (or long64 on
>> 64 bit systems) before use if necessary, so an explicit conversion should not
>> affect the result.
>>
>> IDL> print, !version
>> { x86 linux unix linux 6.3 Mar 23 2006      32      64}
>> IDL>
>> IDL> a=lindgen(10)
>> IDL> print, a[[long(-1ull)]]
>>            0
>> IDL> print, a[[-1ull]]
>>            9
>>
>> Is it a bug or I am missing something?
>
> I think you're expecting -1ull to be negative, but the 'u' in 'ull' means
> 'unsigned'. What you end up with instead of -1 is the largest 64-bit integer
> (this is a nice shortcut when it's actually what you want to do!):
>
```

No, I have expected LONG(-1ull) to be used for subscripting, which is really negative. As Mike wrote, integers are not converted to LONG, that was my wrong assumption.

I am trying to find differences between IDL and FL behavior, and this is one example. In FL, I always convert non-LONG subscripts to LONG. (It's a pity we have no formal definition of IDL syntax and semantics.)

regards,
lajos