Subject: Re: Feature, or bug?
Posted by Lajos Foldy on Sun, 20 May 2012 12:08:12 GMT
View Forum Message <> Reply to Message

On Saturday, May 19, 2012 9:20:40 PM UTC+2, whdaffer wrote:
> Found an interesting, ummm, feature.
>
> I frequently use the following construct.
>
> if n_elements(a) * n_elements(b) * ... * n_elements(z) eq 0 then
> begin
> Message,'....'
> endif
>
> with a catch block to do my preliminary argument processing.
>
> It turns out, there are circumstances where this product can equal 0,
> even when all the n_element()'s return non-zero numbers
>
> To see this, consider...
>
> IDL> print, long(27072)^6
> 0
>
> Any more than 5 arrays with 27072 elements followed by whatever else
> and that construct will always evaluate to 0. I had 6, plus a few that
> had fewer elements.
>
> I also tried a case where I put the arrays with fewer alements up
> front. It failed too.
>
> IDL> a=(b=(c=(d=(e=(f=fltarr(27072))))))
> IDL> print,(n_elements(fltarr(10)) *n_elements(1) *n_elements(a))
> *n_elements(b) * n_elements(c) *n_elements(d) *
> n_elements(e)*n_elements(f) & print,check_math()
> 0
> 0
>
> and check_math says all is okay (If I understand check_math correctly)
>
>
>
> Doesn't seem to be a 32-bit/64-bit issue, I replicated it on a 64-bit
> machine.
>
>
>
>

> IDL> help,!version
> ** Structure !VERSION, 8 tags, length=76, data length=76:
>   ARCH   STRING   'x86'
>   OS   STRING   'linux'
>   OS_FAMILY   STRING   'unix'
>   OS_NAME   STRING   'linux'
>   RELEASE   STRING   '8.1'
>   BUILD_DATE   STRING   'Mar 9 2011'
>   MEMORY_BITS   INT   32
>   FILE_OFFSET_BITS
>   INT   64
> IDL>
>
>
> Since n_elements returns a long (not even a ulong, which, when you
> think about it for a second, it really should, but that wouldn't have
> helped me, in my particular case because that had the same behavior) I
> guess the upshot is: don't use that construct!
>
> Safer would be
>
> if (n_elements(a) eq 0)*... then begin ...
>
> I just never imagined that I could multiply nonzero integers together
> and get a zero!
>
> whd
>
> whd27072)^6

On Saturday, May 19, 2012 9:20:40 PM UTC+2, whdaffer wrote:
> Found an interesting, ummm, feature.
>
> I frequently use the following construct.
>
> if n_elements(a) * n_elements(b) * ... * n_elements(z) eq 0 then
> begin
>   Message,'....'
> endif
>
> with a catch block to do my preliminary argument processing.
>
> It turns out, there are circumstances where this product can equal 0,
> even when all the n_element()'s return non-zero numbers
>
> To see this, consider...

>
> IDL> print, long(27072)^6
> 0
>
> Any more than 5 arrays with 27072 elements followed by whatever else
> and that construct will always evaluate to 0. I had 6, plus a few that
> had fewer elements.
>
> I also tried a case where I put the arrays with fewer alements up
> front. It failed too.
>
> IDL> a=(b=(c=(d=(e=(f=fltarr(27072))))))
> IDL> print,(n_elements(fltarr(10)) *n_elements(1) *n_elements(a))
> *n_elements(b) * n_elements(c) *n_elements(d) *
> n_elements(e)*n_elements(f) & print,check_math()
> 0
> 0
>
> and check_math says all is okay (If I understand check_math correctly)
>
>
>
> Doesn't seem to be a 32-bit/64-bit issue, I replicated it on a 64-bit
> machine.
>
>
>
>
>
> IDL> help,!version
> ** Structure !VERSION, 8 tags, length=76, data length=76:
>   ARCH   STRING   'x86'
>   OS   STRING   'linux'
>   OS_FAMILY   STRING   'unix'
>   OS_NAME   STRING   'linux'
>   RELEASE   STRING   '8.1'
>   BUILD_DATE   STRING   'Mar 9 2011'
>   MEMORY_BITS   INT   32
>   FILE_OFFSET_BITS
>   INT   64
> IDL>
>
>
> Since n_elements returns a long (not even a ulong, which, when you
> think about it for a second, it really should, but that wouldn't have
> helped me, in my particular case because that had the same behavior) I
> guess the upshot is: don't use that construct!
>
> Safer would be

>
> if (n_elements(a) eq 0)*... then begin ...
>
> I just never imagined that I could multiply nonzero integers together
> and get a zero!
>
> whd
>
> whd

27072^6 is 393660688903146891330453504, too big for a long integer, so the last 32 bits are kept (27072^6 modulo 2^32). 393660688903146891330453504 = 91656271578545424 * 2^32, so the result is zero. It's a feature of integer representation. check_math does not report integer overflow.

regards,
Lajos

Subject: Re: Feature, or bug?
Posted by whdaffer on Mon, 21 May 2012 16:31:30 GMT
View Forum Message <> Reply to Message

On May 20, 5:08 am, fawltylangu...@gmail.com wrote:
> On Saturday, May 19, 2012 9:20:40 PM UTC+2, whdaffer wrote:
>> Found an interesting, ummm, feature.
>
>> I frequently use the following construct.
>
>> if n_elements(a) * n_elements(b) * ... * n_elements(z) eq 0 then
>> begin
>> Message,'....'
>> endif
>
>> with a catch block to do my preliminary argument processing.
>
>> It turns out, there are circumstances where this product can equal 0,
>> even when all the n_element()'s return non-zero numbers
>
>> To see this, consider...
>
>> IDL> print, long(27072)^6
>> 0
>
>> Any more than 5 arrays with 27072 elements followed by whatever else
>> and that construct will always evaluate to 0. I had 6, plus a few that
>> had fewer elements.

>
>> I also tried a case where I put the arrays with fewer alements up
>> front. It failed too.
>
>> IDL> a=(b=(c=(d=(e=(f=fltarr(27072))))))
>> IDL> print,(n_elements(fltarr(10)) *n_elements(1) *n_elements(a))
>> *n_elements(b) * n_elements(c) *n_elements(d) *
>> n_elements(e)*n_elements(f) & print,check_math()
>> 0
>> 0
>
>> and check_math says all is okay (If I understand check_math correctly)
>
>> Doesn't seem to be a 32-bit/64-bit issue, I replicated it on a 64-bit
>> machine.
>
>> IDL> help,!version
>> ** Structure !VERSION, 8 tags, length=76, data length=76:
>>   ARCH   STRING   'x86'
>>   OS   STRING   'linux'
>>   OS_FAMILY   STRING   'unix'
>>   OS_NAME   STRING   'linux'
>>   RELEASE   STRING   '8.1'
>>   BUILD_DATE   STRING   'Mar 9 2011'
>>   MEMORY_BITS   INT   32
>>   FILE_OFFSET_BITS
>>   INT   64
>> IDL>
>
>> Since n_elements returns a long (not even a ulong, which, when you
>> think about it for a second, it really should, but that wouldn't have
>> helped me, in my particular case because that had the same behavior) I
>> guess the upshot is: don't use that construct!
>
>> Safer would be
>
>> if (n_elements(a) eq 0)*... then begin ...
>
>> I just never imagined that I could multiply nonzero integers together
>> and get a zero!
>
>> whd
>
>> whd27072)^6
> On Saturday, May 19, 2012 9:20:40 PM UTC+2, whdaffer wrote:
>> Found an interesting, ummm, feature.
>
>> I frequently use the following construct.

>
>> if n_elements(a) * n_elements(b) * ... * n_elements(z) eq 0 then
>> begin
>> Message,'....'
>> endif
>
>> with a catch block to do my preliminary argument processing.
>
>> It turns out, there are circumstances where this product can equal 0,
>> even when all the n_element()'s return non-zero numbers
>
>> To see this, consider...
>
>> IDL> print, long(27072)^6
>> 0
>
>> Any more than 5 arrays with 27072 elements followed by whatever else
>> and that construct will always evaluate to 0. I had 6, plus a few that
>> had fewer elements.
>
>> I also tried a case where I put the arrays with fewer alements up
>> front. It failed too.
>
>> IDL> a=(b=(c=(d=(e=(f=fltarr(27072))))))
>> IDL> print,(n_elements(fltarr(10)) *n_elements(1) *n_elements(a))
>> *n_elements(b) * n_elements(c) *n_elements(d) *
>> n_elements(e)*n_elements(f) & print,check_math()
>> 0
>> 0
>
>> and check_math says all is okay (If I understand check_math correctly)
>
>> Doesn't seem to be a 32-bit/64-bit issue, I replicated it on a 64-bit
>> machine.
>
>> IDL> help,!version
>> ** Structure !VERSION, 8 tags, length=76, data length=76:
>>   ARCH   STRING   'x86'
>>   OS   STRING   'linux'
>>   OS_FAMILY   STRING   'unix'
>>   OS_NAME   STRING   'linux'
>>   RELEASE   STRING   '8.1'
>>   BUILD_DATE   STRING   'Mar 9 2011'
>>   MEMORY_BITS   INT   32
>>   FILE_OFFSET_BITS
>>   INT   64
>> IDL>
>

>> Since n_elements returns a long (not even a ulong, which, when you
>> think about it for a second, it really should, but that wouldn't have
>> helped me, in my particular case because that had the same behavior) I
>> guess the upshot is: don't use that construct!
>
>> Safer would be
>
>> if (n_elements(a) eq 0)*... then begin ...
>
>> I just never imagined that I could multiply nonzero integers together
>> and get a zero!
>
>> whd
>
>> whd
>
> 27072^6 is 393660688903146891330453504, too big for a long integer, so the last 32 bits are kept (27072^6 modulo 2^32). 393660688903146891330453504 = 91656271578545424 * 2^32, so the result is zero. It's a feature of integer representation. check_math does not report integer overflow.

Hmmmm... Well check_math _does_ claim that it will report integer
overflow, in bit 1.

But I wouldn't be using check_math to check for that condition in the
construct I was using anyway, so it's moot that check_math apparently
falls down on the job, at least in this case.

Thanks for the explanation.

whd

>
> regards,
> Lajos

Subject: Re: Feature, or bug?
Posted by Lajos Foldy on Mon, 21 May 2012 17:13:08 GMT
View Forum Message <> Reply to Message

> Hmmmm... Well check_math _does_ claim that it will report integer
> overflow, in bit 1.
>
> But I wouldn't be using check_math to check for that condition in the
> construct I was using anyway, so it's moot that check_math apparently
> falls down on the job, at least in this case.
>

> Thanks for the explanation.
>
> whd
>
>>
>> regards,
>> Lajos

The check_math help says: "Some hardware/operating system combinations may not report all of the math errors listed." Integer overflow is listed, but not checked and reported :-)

Integer overflow is "undefined behaviour" in standard C, so it can not be done in a portable way. The glibc manual says:

FPE_INTOVF_TRAP

Integer overflow (impossible in a C program unless you enable overflow trapping in a hardware-specific fashion).

regards,
Lajos

Subject: Re: Feature, or bug?
Posted by whdaffer on Wed, 23 May 2012 19:32:06 GMT
View Forum Message <> Reply to Message

On May 21, 10:13 am, fawltylangu...@gmail.com wrote:
>> Hmmmm... Well check_math _does_ claim that it will report integer
>> overflow, in bit 1.
>
>> But I wouldn't be using check_math to check for that condition in the
>> construct I was using anyway, so it's moot that check_math apparently
>> falls down on the job, at least in this case.
>
>> Thanks for the explanation.
>
>> whd
>
>>> regards,
>>> Lajos
>
> The check_math help says: "Some hardware/operating system combinations may not report all of the math errors listed." Integer overflow is listed, but not checked and reported :-)

Why yes! So it does. And just one line after the table where it claims
to report integer overflows!

The right hand giveth, and the left taketh away, I guess ;-)

>
> Integer overflow is "undefined behaviour" in standard C, so it can not be done in a portable way. The glibc manual says:
>
> FPE_INTOVF_TRAP
>
> Integer overflow (impossible in a C program unless you enable overflow trapping in a hardware-specific fashion).

Which, means, effectively, that check_math for integer overflow is
worthless since I doubt that ITT or whatever they're called this week
is going to enable overflow trapping in a hardware-specific fashion.

Is the situation similar for the other errors?

whd
