Subject: Re: More on Exp bugs
Posted by Peter Mason on Fri, 13 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

On 11 Dec 1996, David Siskind wrote:
> Anyway, regarding the Crimson user who found wierd things with
> Exp(-710.something), it must be an SGI/IDL thing. I use
> an Indigo and IDL 4.0.1 and I once spent a day (plus a post to this
> group) trying to figure out what appears to be the exact same thing.
> Now I'm real careful about floating under or overflows when doing Exp.
> This did not seem to be a problem with 3.6.1.

Here I go holding forth and stirring once again.

I'm convinced that this problem is due to the introduction of "NaN" and
"Infinity" support since IDL 4.0. I also think that the addition of
these features is what caused the floating-point slowdown observed on at
least some platforms, when comparing IDL 3.6.1 to IDL 4.0.x.

When IDL4.0 first came out, I must say that I didn't really notice the
slowdown on the ALPHA/OSF system I use. (Perhaps this O/S supports
denormals by default, and RSI's enhancements didn't add much extra overhead?)
But I didn't like the way I could segfault IDL with just exp() calls.
(By the way, I STILL can. On a DEC 3000/500 with exp(-90.0) followed by
exp(-9000.0). (Single precision.) Can anyone else do this?)

Anyway, I think that RSI was faced with a tough task in implementing
these FP features on all the IDL platforms; I imagine that each one has
its qirks and peculiarities when it comes down to FP denormals and
exceptions. And some take a bigger performance hit than others. (ALPHA/NT
certainly takes a major performance hit. Doing FP the lean way it likes to,
it makes ALPHA/OSF look tired. Doing it with denormals enabled and
various exceptions changed takes away its edge.)

If anyone's still reading this...
I realise that this is a MAJOR stir, but I was wondering what people's views
are on IDL's "NaN" and "Infinity" support?
Personally: I haven't yet implemented "Infinity" in my IDL programs,
and I haven't used "NaN" much at all. I like the idea of "NaN", but I
started many of my programs before it was around in IDL, and so I found other
ways to cope with "bad values" and the like. I can't be bothered with
FP underflows (just give me 0). Overall, I actually prefer FP support in
IDL the way it was in 3.6.1.

Peter Mason

Subject: Re: More on Exp bugs
Posted by thompson on Mon, 16 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:

> On 11 Dec 1996, David Siskind wrote:

> But I didn't like the way I could segfault IDL with just exp() calls.
> (By the way, I STILL can.   On a DEC 3000/500 with exp(-90.0) followed by
> exp(-9000.0).   (Single precision.)   Can anyone else do this?)

I tried this on a DEC 3000/400 running OSF and IDL v4.0.1, and it did not give a segmentation fault.

> I realise that this is a MAJOR stir, but I was wondering what people's views
> are on IDL's "NaN" and "Infinity" support?

Because our software must also support platforms that do not use IEEE floating point notation, specifically VMS, we do not use any IEEE-specific features. When we read in data files containing NaN values, e.g. from FITS files, we convert them to a normal number representing bad pixels.

> ...   I can't be bothered with FP underflows (just give me 0).

I agree that the error messages about floating underflows is quite a pain. It has caused great confusion among users who are worried that it means that the software is now broken, when in fact it's still working normally.

William Thompson

Subject: Re: More on Exp bugs
Posted by steinhh on Wed, 18 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Mason wrote:
|> -I realise that this is a MAJOR stir, but I was wondering what people's views
|> -are on IDL's "NaN" and "Infinity" support?
|> -Personally: I haven't yet implemented "Infinity" in my IDL programs,
|> -and I haven't used "NaN" much at all.   I like the idea of "NaN", but I
|> -started many of my programs before it was around in IDL, and so I found other
|> -ways to cope with "bad values" and the like.
|>

In article <598caq$6g9@cpca3.uea.ac.uk>, f055@uea.ac.uk (T.Osborn) writes:

|> NaN is, I think, a good improvement in dealing with missing data values - but

|> perhaps only because the previous support was so patchy. But remaining
|> problems with it are:
|>

A major problem that I see is that only "machines which implement the IEEE
standard for binary floating-point arithmetic have two special values for
undefined results".

Which platforms do have IEEE FP arithmetic, and which platforms do not?
In other words, there's no way of using NaN values in IDL in any
portable sense, is there? If your program uses and relies upon
NaN/Infinity, what happens when you run on a platform that doesn't
support it?

IMHO, this is a major obstacle for using this (in principle very good)
idea. If you have to put in extra lines of code to handle these exceptions
on non-IEEE platforms anyway, well, there's not really any point, is
there? You'll get *portable* and *shorter* programs devising your own
schemes and using only those, although at the cost of efficiency.

I, for one, would prefer that RSI wouldn't waste time implementing
new "features" that will not be available on all their platforms.
Sure, you may think that some new stuff is cool etc, and that it
might be a nice sales pitch. But for the "big" customers, writing
software that has to run on as many platforms as possible, there's
just no added value at all in such gizmos. These are also the people
expanding your list of potential customers (those that want to run
the software).

Now, if there was a way to turn on/off "fudged" NaN/Infinity values for
any platform, that would do the trick. Yes - it would be at the expense of
efficiency, but not as much as writing IDL statements to handle the cases.

|> 1) You can't use it to indicate missing values in integer variables (a
|> problem if part of a calculation involves integers, even if the start and end
|> points do not).

Quite agree. There should be a way of specifying NaN_INTEGER, NaN_LONG, (and
perhaps even NaN_BYTE!), as well as NaN_FLOAT/DOUBLE for non-IEEE FP platforms.
Sure, it takes away a little speed, a little of the accessible range of values,
but all of this should be *optional* and *platform independent*.

|>
|> -I can't be bothered with FP underflows (just give me 0).
|>
|> Hear hear.
|>

Quite agree...

Stein Vidar

Subject: Re: More on Exp bugs

Posted by f055 on Wed, 18 Dec 1996 08:00:00 GMT

View Forum Message <> Reply to Message

-But I didn't like the way I could segfault IDL with just exp() calls.
-(By the way, I STILL can.   On a DEC 3000/500 with exp(-90.0) followed by
-exp(-9000.0).   (Single precision.)   Can anyone else do this?)

Doesn't happen on a DEC 3000/600 with IDL 4.0.

-I realise that this is a MAJOR stir, but I was wondering what people's views
-are on IDL's "NaN" and "Infinity" support?
-Personally: I haven't yet implemented "Infinity" in my IDL programs,
-and I haven't used "NaN" much at all.   I like the idea of "NaN", but I
-started many of my programs before it was around in IDL, and so I found other
-ways to cope with "bad values" and the like.

NaN is, I think, a good improvement in dealing with missing data values - but
perhaps only because the previous support was so patchy. But remaining
problems with it are:

1) You can't use it to indicate missing values in integer variables (a
problem if part of a calculation involves integers, even if the start and end
points do not).

2) Many routines still don't have an option to ignore missing values.

3) You cannot write out and read back in NaN data when using ASCII data files.

-I can't be bothered with FP underflows (just give me 0).

Hear hear.

```
......................... Dr Tim Osborn                    . t.osborn@uea.ac.uk
.... ___/.. __ /.. /.. /. Senior Research Associate       . phone:01603 592089
... /..... /. /.. /.. /.. Climatic Research Unit          . fax:  01603 507784
.. /.....  __/.. /.. /... School of Environmental Sciences.
. /..... /\ ... /.. /.... University of East Anglia       .
____/.._/..\_..____/..... Norwich  NR4 7TJ                .
......................... UK                              .
```