

Re: Problem with log function – Intel fortran compiler under Linux

Source: <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2007-10/msg00257.html>

- *From:* Herman D. Knoble <SkipKnobleLESS@xxxxxxxxxxxxxxxx>
 - *Date:* Tue, 09 Oct 2007 11:18:30 -0400
-

Arjen: If you can get access to the Lahey compiler (for Windows or Linux) use the compiler options:

`-chk(a,e,s,u,x) -chkglobal -g -pca -stchk -trace -nzero -w -o0 -trap diou`

or under Linux:

`---chkglobal -g ---co ---nsav ---trace ---o0 ---xref ---pca ---warn`

This has very good potential to flush out a programming error. If there is no run-time errors detected, that would be stronger evidence that it is the Intel compiler.

Skip

On Mon, 08 Oct 2007 23:58:35 -0700, Arjen Markus <arjen.markus@xxxxxxxxxxx> wrote:

-| On 8 okt, 14:58, Gordon Sande <g.sa...@xxxxxxxxxxxxxxxx> wrote:

-|> On 2007-10-08 06:44:15 -0300, Arjen Markus <arjen.mar...@xxxxxxxxxxx> said:

-|>

-|>

-|>

-|>

-|>

-|> > On 4 okt, 15:35, Arjen Markus <arjen.mar...@xxxxxxxxxxx> wrote:

-|> >> Hello,

-|>

-|> >> we are experiencing a rather nasty problem with the Intel Fortran

-|> >> compiler under Linux

-|> >> (version 9.0). The program is big and so far we have not been able to

-|> >> trim the code

-|> >> so that a moderately sized program displays the problem too.

-|>

-|> >> The point is this:

-|>

-|> >> The first application of the log function on double precision reals

-|> >> causes

-|> >> an NaN (while the argument is a perfectly acceptable value of about

Re: Problem with log function – Intel fortran compiler under Linux

-|> >> 0.6).
-|>
-|> >> If we add a dummy statement like:
-|>
-|> >> aa =log(r)
-|>
-|> >> as one of the first statements, the value of aa becomes NaN and some
-|> >> later computations
-|> >> succeed. (NaNs occur at a different positions).
-|>
-|> >> Here is some more information:
-|>
-|> >> Intel Fortran: 9.0
-|> >> Linux: Red Hat Enterprise Linux, ES release 4.
-|>
-|> >> Has anyone encounter similar problems? Does anyone know how to solve
-|> >> this problem?
-|>
-|> >> Regards,
-|>
-|> >> Arjen
-|>
-|> > Hm, things are brightening a bit:
-|> > – My original posting reflected the situation with all
-|> > sources compiled with debugging on
-|> > – I then tried with debugging on and array bound checking
-|> > – no error message about possible array bound violation
-|> > but the first NaN appears later on
-|> > – Without any options (so only the defaults) the first
-|> > NaN appears earlier on.
-|>
-|> > Especially this latter observation means that I have
-|> > a lot less code to worry about :)
-|>
-|> > Regards,
-|>
-|> > Arjen
-|>
-|> It is rather easy to lie about array sizes in a way that confuses
-|> array bounds checking when using F77 (assumed or explicit sizes)
-|> semantics so the lack of out of bounds diagnostics can be misleading.
-|> To check against that you need to use one of the "sturdier" debugging
-|> systems like Salford/Silverfrost that supplies their own descriptors
-|> at all calls. It makes third party compiled libraries "difficult" if
-|> not impossible. (The problem is that the "new" declaration is not checked
-|> against the "old" declaration across the call but the subscript checking
-|> is only against the "new" declaration and so can be easily mislead.
-|> The better systems check against the descriptors but that requires much
-|> more information than the "traditional" Fortran implementation to be
-|> available at the call.)
-|>

Re: Problem with log function – Intel fortran compiler under Linux

-> When you get into this sort of problem it is time to think multiple
-> compilers, and even other operating systems. Salford is on Windows.
-> The choice is between bowing to the needs of the other system (i.e.
-> running under Windows for a while) or continuing to battle with
-> ineffective tools. It sounds like you have all sources readily
-> available so using Salford for a while may not be much of a bother.– Tekst uit
oorspronkelijk bericht niet weergeven –
->
-> – Tekst uit oorspronkelijk bericht weergeven –
-|
-|Well, the program runs under Windows, using CVF 6.6C (and has
-|been developed on several platforms over the years), so we
-|were unhappily surprised to see these nasty problems occur
-|under Linux.
-|
-|To further elaborate on the problem:
-|
-|The program uses a single REAL array stored in a blank COMMON
-|block to pass the data around. This REAL array is used as
-|double precision REAL data and as double precision COMPLEX data
-|by passing various array elements to the subroutines involved.
-|(The pieces that arise that way do NOT overlap – I checked)
-|
-|This is NOT the cause of the problem though:
-|When I replace that REAL array by a double precision array,
-|the SAME problem occurs.
-|
-|My strategy right now is to radically skip pieces of the
-|program that seem irrelevant to the computations that show
-|the problem. I will even try if valgrind offers any insight.
-|
-|But the problem is very resilient: I have experimented with
-|many different changes to the code (explicit declaration of
-|all variables, using INTRINSIC, changing the type of the array
-|as described above), and most have no effect on the symptoms.
-|
-|Adding one write statement does make the NaNs disappear, but
-|that hardly seems a good solution.
-|
-|Regards,
-|
-|Arjen