Subject: Re: Array has a corrupted discriptor, reliable?
Posted by Joost Aan de Brugh on Tue, 29 Jul 2008 16:01:04 GMT
View Forum Message <> Reply to Message

On Jul 29, 3:50 pm, Norbert Hahn <ihates...@nospam.invalid> wrote:

> Some Fortran compilers allow range checking with arrays as a compile
> time option. If your compiler support this function you'll get an
> error condition within the Fortran program. However, I don't know
> how to handle errors within Fortran code.
>
> Norbert

Unfortunately, I don't know that much of Fortran as well. I use
%val(argv(..)) in Fortran to convert from memory address to value, but
Fortran has no clue what data type in in there. So if I put an integer
varibale with value '2' and give it to Fortran. And I use Call
routine(%val(argv(1)) and in the routine the variable is defined as a
real, it becomes a stupid value corresponding to the only the second
bit set (or however the integer '2' is stored).

The same way I can pass whole arrays. For example, if I give the
memory address of a 10-element floatarray and in Fortran i Call
routine(%val(argv(1))) and the the corresponding variable in the
routine is a Real, Dimension(10), it works fine. For safety, I might
as well pass the 10 as an integer from IDL to Fortran so that Fortran
knows that there are 10 elements.

But if someone with humour sets the flag -fdefault-real-8 at the
Fortran compiler somewhere in an obscure Makefile, Fortran will think
it is a double array, taking five values constructed somehow from my
floatarray and the next 40 bytes in memory. This though Fortran still
thinks that the array is 10 elements long and there is no problem as
long as the computer allows me to write (or read) from the 40 extra
bytes (God knows what is in there).

In my first example, I was expecting a segmentation fault caused by
Fortran and not the nice little "Array has a corrupted discriptor".
But now it looks like solved, since I tried this example (Fortran
thinking they are doubles while they are floats) and I get my desired
Segmentation fault.

Kind regards,
Joost Aan de Brugh