Subject: Re: Common block compilation error
Posted by Jeremy Bailin on Tue, 23 Nov 2010 15:49:05 GMT
View Forum Message <> Reply to Message

On Nov 22, 9:00 am, Gray <graylikethec...@gmail.com> wrote:
> Hi all,
>
> I'm using the NASA IDL astronomy library's MINF_BRACKET and
> MINF_PARABOLIC routines to minimize a function. Doing so requires
> that I use a common block for some other parameters. The routine that
> calls the minimization routines gets passed the parameters and stores
> them in the common block. I had thought that I could name the input
> parameters for the calling function as the same as the common block
> variables, so that they would be automatically stored, but doing so
> gives a compilation error, and so I have to name them differently and
> assign their values to the common block variables. Is there any
> reason why this should be so? It seems an unnecessary extra step...
>
> --Gray
>
> Code (gives error):
>
> FUNCTION m87_scl, scl
> common m87scal, img, im0
> return, robust_sigma(img*scl-im0,/zero)
> end
>
> FUNCTION m87_scale, img, im0
> common m87scal
> xa = 0.1 & xb = 10.
> minf_bracket, xa, xb, xc, func_name='m87_scl'
> minf_parabolic, xa, xb, xc, scl, func_name='m87_scl'
> return, scl
> end

The problem is that it doesn't know about the common block in the function definition, so it creates variables named "img" and "im0" that are not part of the common block... and then when you tell it that you want variables of the same name that are part of a common block, it complains that it already has variables with those names that are using their own memory.

If they weren't being passed in as function arguments, it's easy, but since they are I don't see any easy way around it. If the problem is that those are huge and you're walloping tons of memory by having two copies of them, you could try passing pointers instead, so you just need to make copies of the pointers, and then dereference them inside m87_scl.

-Jeremy.