Subject: Re: !ERR and MPFIT
Posted by davidf on Tue, 16 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:

> * to everybody: any suggestions on how to generically signal an error
> condition? My thoughts were:
>
> - ERROR keyword variable - don't like this, since then the
> function has to accept keywords
>
> - define a new system variable - don't like this either, since
> it's hard to do system variables right
>
> - common block variable - not very pretty, but gets the job done.
>
> To be clear, this is some kind of error flag that a user routine would
> (optionally!) set to signal an abnormal termination condition. Right
> now I am leaning toward the common-block approach. Sorry David.

Oh, I like it. And to tell you the truth, this might
be the *perfect* situation for a common block. Just
don't be putting 'em in a widget program! :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: !ERR and MPFIT
Posted by Craig Markwardt on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> writes:
>
> How about putting something like a conditional
>
> message, 'Number of iterations exceeded limits'
>
> in the user procedure, to signal MPFIT. Put no CATCH in user

> procedure. Then, in MPFIT you have CATCH statment followed by a
> graceful return. If a parameter is so out of bounds that user
> procedure signals, its unlikely you can obtain anything meaningful
> from MPFIT, then why not let it quit with CATCH? No common blocks,
> pointers or anything else. Fully self-contained, no conflicts due to
> multiple instances running, etc.

Good suggestion, and that actually what happens right now :-)

But I also want something a little more formal. I still feel a little attached to the common block implementation. Still purely optional on the part of the user. We'll see multi-threaded IDL coming from a mile away, if it ever comes. I'll deal with it then.

Right now I multiprocess MPFIT on my dual-CPU machine by running two IDL sessions. Works great!

Craig

--

---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: !ERR and MPFIT
Posted by Pavel Romashkin on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

> In this case it's more of a termination condition than an error. For
> example, the user function may decide that a parameter has gotten out
> of bounds unrecoverably. I would like MPFIT to return gracefully
> rather than barfing if possible, so the user routine needs a way to
> signal MPFIT that something is wrong.

How about putting something like a conditional

message, 'Number of iterations exceeded limits'

in the user procedure, to signal MPFIT. Put no CATCH in user procedure. Then, in MPFIT you have CATCH statment followed by a graceful return. If a parameter is so out of bounds that user procedure signals, its unlikely you can obtain anything meaningful from MPFIT, then why not let it quit with CATCH? No common blocks, pointers or anything else. Fully self-contained, no conflicts due to multiple instances running, etc.

Cheers,

Pavel

Subject: Re: !ERR and MPFIT
Posted by Craig Markwardt on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

Pavel Romashkin <promashkin@cmdl.noaa.gov> writes:
>
> I apologize if I am missing a problem with error handling that Craig is
> solving. I just want to ask why can't you use CATCH to handle errors
> conditions? It seems to me that CATCH combined with MESSAGE procedure works
> quite well for user-defined errors, and CATCH by itself works great for
> internal IDL routines. This also eliminates the need for separate error
> handlers.

In this case it's more of a termination condition than an error. For example, the user function may decide that a parameter has gotten out of bounds unrecoverably. I would like MPFIT to return gracefully rather than barfing if possible, so the user routine needs a way to signal MPFIT that something is wrong.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: !ERR and MPFIT
Posted by Pavel Romashkin on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

I apologize if I am missing a problem with error handling that Craig is solving. I just want to ask why can't you use CATCH to handle errors conditions? It seems to me that CATCH combined with MESSAGE procedure works quite well for user-defined errors, and CATCH by itself works great for internal IDL routines. This also eliminates the need for separate error handlers.

Cheers,
Pavel

Craig Markwardt wrote:

> Unfortunately I chose to use the !ERR system variable. If !ERR is set
> to a negative number, then MPFIT aborts the run, assuming that there
> was an unrecoverable error.
>
> I now realize that using !ERR was probably a mistake. Why? RSI seems
> to want to make it obsolete. Through their error-handling flavor of
> the month program, !ERR seems to have fallen out of favor. Also,
> there are quite a few actions which might set !ERR accidentally in the
> user's function without actually signalling an error condition.
>
> So I have two questions:
>
> * to people who use MPFIT: does anybody actually use the !ERR status
> variable to control the fitting process? If not, then I would
> consider removing it.
>
> * to everybody: any suggestions on how to generically signal an error
> condition? My thoughts were:
>
> - ERROR keyword variable - don't like this, since then the
> function has to accept keywords
>
> - define a new system variable - don't like this either, since
> it's hard to do system variables right
>
> - common block variable - not very pretty, but gets the job done.
>
> To be clear, this is some kind of error flag that a user routine would
> (optionally!) set to signal an abnormal termination condition. Right
> now I am leaning toward the common-block approach. Sorry David.
>
> Thanks,
>
> Craig

Subject: Re: !ERR and MPFIT
Posted by Craig Markwardt on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

m218003@modell3.dkrz.de (Martin Schultz) writes:
>
> In article <MPG.129be687cb5374e598996c@news.frii.com>,
> davidf@dfanning.com (David Fanning) writes:
>> Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:
>>
>>> now I am leaning toward the common-block approach. Sorry David.

>>
>> Oh, I like it. And to tell you the truth, this might
>> be the *perfect* situation for a common block. Just
>> don't be putting 'em in a widget program! :-)
>>
>
> Now, what happens if you have two or three widgets open each of which
> is calling MPFIT through some means. Would a common block still work?
> And please think once more: may not be possible now, but I am quite certain
> that RSI will one day support SMP, so it could indeed happen that those
> calls to MPFIT were executed simultaneously! I'd go for the keyword - this
> is clean.

I totally understand what you are saying. Keywords make everything
clean. But consider the following function:

```
function myfunc, x, p
  ...
end
```

It doesn't accept keywords. Now, if MPFIT tries to call this function
with an ERROR keyword, everything crashes. I could try CATCHing such
an error, and retrying the function call with without the ERROR
keyword, but then what's more ugly? By the way, I've changed the
implementation of MPFITFUN to use common blocks instead of pointers
(handles really), and it's become much more clean and easy to read
now.

The common block implementation has its virtues. It's totally
optional. It wouldn't collide with any other system error variables.
It's certainly better than my current use of !ERR. And, currently,
it's guaranteed that only one session of MPFIT can be running
simultaneously.

I am sure that when (if!) RSI implements multithreaded IDL, almost
everything is going to crash. Not just MPFIT. Consider every program
that uses common blocks, *especially* the thousands of IDL library
scripts, assumes a single thread of execution. RSI will have to
implement some kind of new functionality to keep things working and I
for one will depend on that! :-)

Craig

--

 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: !ERR and MPFIT
Posted by m218003 on Wed, 17 Nov 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.129be687cb5374e598996c@news.frii.com>,
davidf@dfanning.com (David Fanning) writes:
> Craig Markwardt (craigmnet@cow.physics.wisc.edu) writes:
>
>> now I am leaning toward the common-block approach. Sorry David.
>
> Oh, I like it. And to tell you the truth, this might
> be the *perfect* situation for a common block. Just
> don't be putting 'em in a widget program! :-)
>

Now, what happens if you have two or three widgets open each of which
is calling MPFIT through some means. Would a common block still work?
And please think once more: may not be possible now, but I am quite certain
that RSI will one day support SMP, so it could indeed happen that those
calls to MPFIT were executed simultaneously! I'd go for the keyword - this
is clean.

Cheers,
Martin

--
[[[[[[[[[[
[[Dr. Martin Schultz Max-Planck-Institut fuer Meteorologie [[
[[Bundesstr. 55, 20146 Hamburg [[
[[phone: +49 40 41173-308 [[
[[fax: +49 40 41173-298 [[
[[martin.schultz@dkrz.de [[
[[[[[[[[[[