Subject: Re: Calling IDL from Fortran called by IDL
Posted by Mark Rivers on Wed, 28 Nov 2001 14:45:01 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> wrote in message
news:onherfoc0q.fsf@cow.physics.wisc.edu...
>
> Oooh, but be careful, because if you allow your IDL variable to be
> released, say by reassigning it, then your external C routine will
> probably end up overwriting some part of memory you didn't want it to.

Yes, this technique should only be used for carefully written IDL
"applications", and not for interactive use. Any IDL variable whose address
has been passed to the external C or FORTRAN code can never appear on the
left hand side of an assignment.

As I think about it I actually use the technique I described all the time
now, without really being aware of it. I am using it in our real-time
control system, which is a client/server system using TCP/IP networking.
IDL interacts with this by doing the following:

- IDL wants to do a "put" or "get" of a real-time system parameter
- IDL calls external C code which issues a TCP/IP request and returns
immediately.
- TCP/IP response comes back to the C code
- IDL polls the C code to see if the response has come back
- IDL reads the response from the C code

This idea can be directly applied to the problem of interacting with
compute-bound FORTRAN code.

- IDL calls C wrapper which spawns a new FORTRAN process. It does not need
to be a thread, but can be a new process because it is not going to share
memory with IDL.

- FORTRAN process sends progress messages to C wrapper interface via
sockets, pipes (or any other inter-process communication mechanism)

- IDL polls progress via C wrapper interface

> Also, one always needs to worry that things like malloc() and printf()
> may not be threadsafe on one's platform. If your work thread and the
> IDL thread clash, then *pow*, that will hurt.

This eliminates that problem because the FORTRAN is a new process, not a
thread.

On a related note, I have recently added a GUI interface to my IDL tomography processing routines, which were formerly accessible only via the command line. The IDL routines that do the processing are compute-bound for 10 minutes or longer. It was trivial to make these routines "GUI-friendly" by making them accept 2 optional keywords, the IDs of a "status widget" and an "abort widget". If these keywords are defined, and point to a valid widget then on each time through a loop (every couple of seconds) progress information is written to the status widget, so status information is visible in the GUI. "widget_event" is called for the abort widget, and the uvalue of it is read. If it is 1 then the computation is aborted:

```
if (widget_info(abort_widget, /valid_id)) then begin
   event = widget_event(/nowait, abort_widget)
   widget_control, abort_widget, get_uvalue=abort
   if (abort) then return
endif
```

A very nice side-effect of this technique is that, because widget_event is being called every couple of seconds, both my GUI screen and the IDLDE window respond to mouse events, so they are refreshed and can be moved around on the screen. This was not possible previously, they were totally unresponsive for 10 minutes when the calculation was in progress.

Mark Rivers