Subject: Re: Can DLLs Multi-task?
Posted by Rick Towler on Thu, 03 Oct 2002 18:15:45 GMT
View Forum Message <> Reply to Message

"David Fanning" <david@dfanning.com> wrote >

> Here is a question from an IDL user who has no newsgroup
> access.

> ;**
> I am developing an IDL program which have to call some external C
> routines (actually some DLL, because I work on Windows environment).
> I am wondering if it is possible in some ways to call more than one DLL
> at the same time, I.e. in multitasking, so telling to IDL not to wait
> for the DLL return.

I don't think so... I don't think you can tell IDL to return immediately
after the call to your external routine since AFAIK (and I don't know much)
IDL doesn't really expose any mechanism for callbacks from external
routines. And how would you get your data back to IDL?

But...

> For example I need to launch a DLL to continuously monitor some
> temperature sensors, but in the meanwhile I have to run another DLL to
> read an image from a CCD device. I would need to have the temperature
> variables continuously updated by the first DLL so that I can read and
> display their values while getting the image from the CCD device.
> Is all that possible with IDL?
> (I have IDL 5.5)

IDL does expose one callback mechanism that can be used by external
routines, WIDGET_STUB and it's associated functions.

I am no astrophysicist (really, the closest I come is Sky & Telescope) so
please pardon my ignorance. I am guessing you have this CCD device that is
rather slow in acquiring an image. It is temperature sensitive and you need
to monitor the temperature of the CCD while it is acquiring the image so you
can apply some corrections?

Off the top of my head you could issue a call to acquire from your CCD
device. That call would set a one shot timer and then return to IDL. That
timer in your external routine would fire and call its callback which would
call your CCD acquisition routine. While this is going on, a timer loop in
IDL would poll your temp sensors. When your CCD acquisition routine returns
(in your external timer event callback), you make a call to

IDL_WidgetStubIssueEvent() and return the memory address of your CCD data.

The trick, which I haven't been able to figure out, is how to get at that
data which is located at the address you would return with your call to
IDL_WidgetStubIssueEvent(). I have played around with this before, but
didn't know what to do with the address I returned to IDL. My guess is that
you make another call to your external routine where you pass the address
and it returns your data? But like I said, I didn't get it to work.

Anyone care to elaborate on that?

-Rick

Subject: Re: Can DLLs Multi-task?
Posted by Peter Mason on Thu, 03 Oct 2002 22:32:09 GMT

View Forum Message <> Reply to Message

Here's my "string and glue" approach to this sort of problem. I haven't
done image-capturing like this in IDL, but I have done other threaded
programs.
The temperature-monitoring calls are probably okay as they are (fast?) but
the image-capturing call needs some work. From the sound of it, it is a
synchronous call. (Gianluca is basically asking about ways around
synchronous calls.) Some wrapper functions are required to make it run in
a separate thread so that it is asynchronous as far as your IDL program is
concerned. Briefly (assuming you'll be capturing many images): A call to
create a C-side thread, a call to trigger an image capture (which is then
done in the separate thread), possibly a status-polling call, possibly one
or two retrieve-capture calls, and finally a call to end the thread and
clean up once you're done with all your image capturing.
If the image-capturing call can be passed an array (allocated on the IDL
side with a statement like img=BYTARR(x,y)) into which to store the image
then life is relatively simple. All you need is some way for the C routine
to let the calling IDL program know when the image is ready. The simplest
approach is to write a little C routine that just reports whether or not the
image-capturing thread is busy (or something like that). (This is off the
main IDL thread, not the image-capture thread.) You'd then poll this off a
timer event on the IDL side. A more sophisticated approach is to issue an
event from the C image-capture thread to IDL when the image is ready, along
the lines that Rick has described. (I haven't used
IDL_WidgetStubIssueEvent() myself, but I have other, less elegant ways of
achieving this sort of thing.)
If the image-capturing insists on its own address for the image then things
could be a bit more complicated. Three possibilities come to mind. If
the image is always a fixed size you could allocate it up front in a "heap"
variable on the IDL side, pass this to the image-capture trigger call, and

just make the image-capture call copy the image across when it's ready.
The next approach is to have an "image size" C routine that you call from
IDL when the image is ready, allocate an array for the image in IDL, then
call an "image retrieve" C routine that simply copies the image to this
array.   The slickest approach requires that you build a DLM rather than
just use CALL_EXTERNAL.   Here, the "image-retrieve" C routine returns an
IDL array of the captured image.   This is very easy to do with RSI's
IDL_ImportArray() C function.

The mechanics of threading on Win32 are relatively straightforward.   The
hard parts are knowing what to thread (not a problem here) and retaining a
clear understanding of which thread might be accessing what memory when,
especially on multiprocessor computers (often a problem; nasty little
teeth, too).   There are various ways to prevent memory-access conflicts
(e.g., Win32 critical sections, events, mutexes), but simply taking care
with your "logic flow" goes a long way towards avoiding them.

Cheers
Peter Mason

"David Fanning" <david@dfanning.com> wrote in message
news:MPG.1805ef78a52763df9899d1@news.frii.com...
> Folks,
>
> Here is a question from an IDL user who has no newsgroup
> access. He asked me if I would pass this question on
> to you:
>
> ;**
> I am developing an IDL program which have to call some external C
> routines (actually some DLL, because I work on Windows environment).
> I am wondering if it is possible in some ways to call more than one DLL
> at the same time, I.e. in multitasking, so telling to IDL not to wait
> for the DLL return.
>
> For example I need to launch a DLL to continuously monitor some
> temperature sensors, but in the meanwhile I have to run another DLL to
> read an image from a CCD device. I would need to have the temperature
> variables continuously updated by the first DLL so that I can read and
> display their values while getting the image from the CCD device.
> Is all that possible with IDL?
> (I have IDL 5.5)
>
>
> Thank you very very much for your valuable help,
>

> Gianluca Li Causi
> licausi@coma.mporzio.astro.it
> ;**
>
> Cheers,
>
> David
>
> --
> David W. Fanning, Ph.D.
> Fanning Software Consulting, Inc.
> Phone: 970-221-0438, E-mail: david@dfanning.com
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Can DLLs Multi-task?
Posted by Paul Probert on Fri, 04 Oct 2002 14:26:43 GMT
View Forum Message <> Reply to Message

Gianluca,
 You could use threads, but when I tried that (under windows) I had
many
problems. A solution which is fairly robust is to have a separate
executable,
a console application, doing your data taking, and writing the data to a
memory
mapped file. Then your DLL can map that file and read the data. It is
quite
fast that way. You must use mutexes and events to synchronize and
communicate,
but this is all clearly explained in the win32 documentation.

clip
> I am developing an IDL program which have to call some external C
> routines (actually some DLL, because I work on Windows environment).
> I am wondering if it is possible in some ways to call more than one DLL
> at the same time, I.e. in multitasking, so telling to IDL not to wait
> for the DLL return.
>
> For example I need to launch a DLL to continuously monitor some
> temperature sensors, but in the meanwhile I have to run another DLL to
> read an image from a CCD device. I would need to have the temperature
> variables continuously updated by the first DLL so that I can read and
> display their values while getting the image from the CCD device.
> Is all that possible with IDL?
> (I have IDL 5.5)
>

> Thank you very very much for your valuable help,
clip

--
Paul Probert
The University of Wisconsin-Madison