Subject: Re: Passing file LUN to C routine
Posted by JD Smith on Tue, 06 May 2003 18:33:56 GMT
View Forum Message <> Reply to Message

On Tue, 06 May 2003 11:22:29 -0700, Ben Tupper wrote:

> Hello,
>
> I have IDL interfaced (via DLM in C) with a frame grabber for collecting
> video. I want to pass a file's LUN to C (repeatedly) so that the C
> routine can write the most recent frame to to the file. My idea is to
> place IDL in an interruptable loop (widget timer); in each iteration the
> C routine is passed the LUN, writes the image and then returns a flag
> such as the number of bytes written to IDL. Later I'll poke around with
> the images by using an ASSOCiated variable within IDL.
>
> It's a reasonable plan that is rapidly going amuck; what I have tried so
> far causes IDL to crash. I have been using the IDL_FileStat() function
> to get the required FILE pointer for C. The compiler doesn't C complain
> about the setup; but it kills IDL when I run it. Methinks
> IDL_FileStat isn't my friend anymore.
>
> So ...
>
> (1) How do I properly convert the LUN in IDL into a FILE pointer? (I
> guess the question maybe better phrased as how do I get IDL to give me
> the FILE pointer associated with the LUN I pass?) I think I need this
> because the C routine fwrite requires it.
>
> (2) Is this creating an unstable situation by leaving the file open all
> the time until some condition is met in IDL?
>
> (3) Would I be better off passing the filename to C and having C
> open-write-close for each iteration?
>
>
I'm just reaching here, but I suspect the answer depends on how fast
you're grabbing data. Probably the fastest way is to use the new shared
memory features of IDLv5.6. As the data is collected into the shared
memory, the C code writes it to file (only if you need to archive it), and
then signals IDL with a boolean flag located somewhere in the shared
memory segment. IDL has already mapped this segment, so it has access to
the data as fast as the DLM can place it there, without any need for
copying or going through the I/O subsystem.

JD