Subject: Re: simultaneous read/write to tcp/ip socket
Posted by suicidaleggroll on Mon, 09 Sep 2013 16:05:06 GMT
View Forum Message <> Reply to Message

On Tuesday, September 3, 2013 1:41:05 AM UTC-6, Helder wrote:
> On Tuesday, September 3, 2013 2:25:28 AM UTC+2, suicida...@gmail.com wrote:
>
>> I have a device I'm attempting to interface with via TCP/IP. I've already implemented the read side of the problem, the IDL process is pulling in continuous data, parsing it, logging it, etc. I haven't implemented the write side of the problem, but it shouldn't be an issue. The real problem is I'm not seeing how I'll be able to implement both read and write at the same time.
>
>>
>
>>
>
>>
>
>> The writing is user-driven, messages are only sent to the device when the user requests them via command prompt. The reading is device-driven, it's constantly sending data that needs to be read, interpreted, logged, etc in real time. Only one TCP/IP connection to the device is allowed at a time. It should be all command line, no GUI.
>
>>
>
>>
>
>>
>
>> I keep thinking of different ways I could implement this, but keep running into walls. Obviously I can do continuous blocking read without issue, and I'm sure I could do continuous write (blocking on user IO) without issue, but how do I implement a code that can receive continuously while simultaneously prompting the user for input and sending when requested? I've considered IDL_IDLBridge, and have used this successfully in the past for other problems, but I'm not presently seeing how this could solve the problem given the constraints of the situation.
>
>>
>
>>
>
>>
>
>> Any help would be appreciated.
>
>
>
> Hi,
>

> I'm not sure if your question is about how to implement a loop in the command line to do this or on how to read/write to a socket.
>
> In the last case, how about using after the socket command a openu to open for input and output?
>
>
>
> There are quite a few ways to implement the former with a GUI, but I guess you don't like that.
>
> I came up with this. It's not great, but it does (with some delay) the job:
>
>
>
> pro testkeyboardloop
>
> keepworking = 1b
>
> l = list()
>
> while keepworking do begin
>
> k = get_kbrd(0)
>
> case byte(k[0]) of
>
> 0b:;begin
>
> ; print, 'read to device'
>
> ; wait, 0.2
>
> ; end
>
> 113b:keepworking=0b ;'q' to escape
>
> 35b:begin ;'#' to do something with the string
>
> print, l
>
> l->Remove, /all
>
> end
>
> else:l->add, k
>
> endcase
>

> ;
>
> endwhile
>
> print, 'finished'
>
> end
>
>
>
> I would be interested in knowing if you were happy with this or not... because eventually I would also be interested in a better solution.
>
>
>
> Regards,
>
> Helder

Thanks, I didn't realize that you could use get_kbrd to do a non-blocking read. This may work in the interim, but ultimately I was hoping for a method that would allow a blocking read so the user could do the normal text editing on what they're typing (moving the cursor, backspace, delete, etc), similar to what a basic "read" command gives you. get_kbrd doesn't appear to support delete/backspace, and implementing arrow keys and cursor position would be a bit of a nightmare to code up on the back-end.

If the user input was just a letter or two it would be fine, but they're complicated commands with directory/file names, etc. and lack of support for delete/backspace would cause problems.

At this this will give me a temporary work-around though, thanks.