
Subject: Re: Non-blocking socket

Posted by [Helder Marchetto](#) on Fri, 08 Nov 2013 08:49:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Thursday, November 7, 2013 5:54:43 PM UTC+1, Jim P wrote:

> On Tuesday, November 5, 2013 3:50:53 PM UTC-7, Helder wrote:

>

>> Hi,

>

>>

>

>> well, working on sockets is not my thing, but I can't get around the fact that IDL "waits" for an answer and this is not the case elsewhere.

>

>>

>

>> Let me explain...

>

>>

>

>> I open a socket this way:

>

>>

>

>>

>

>>

>

>> SOCKET, Unit, IP, Port, \$

>

>>

>

>> CONNECT_TIMEOUT=Connect_Timeout, \$

>

>>

>

>> READ_TIMEOUT=Read_Timeout, \$

>

>>

>

>> WRITE_TIMEOUT=self.Write_Timeout, \$

>

>>

>

>> Error=RetError, /GET_LUN

>

>>

>

```

>>
>
>>
>
>> The IP is a local one and points to another software on the pc (Win 7-64 bits) that acts as
socket server. The timeouts are set to 5, 1 and 1 seconds for connection, read and write,
respectively.
>
>>
>
>>
>
>>
>
>> Now when I attempt to read a simple string (a device name) in a loop of 100, it takes roughly
100 seconds.
>
>>
>
>> To read I use something like:
>
>>
>
>>
>
>>
>
>> Value = "
>
>>
>
>> WRITEU, Unit, 'cmd '+STRTRIM(I,2)
>
>>
>
>> READF, Unit, Value
>
>>
>
>>
>
>>
>
>> I could not get around this by changing the timeouts or removing them (even worst!). So, what
I did is take some time to learn to do this in another language and find out if the socket server is
slowing me down or the way the socket is implemented in IDL. Well I tried this in python and after
quite some pains and fiddling with a language I don't know about, I managed to get the job done
in 0.007 seconds (1.5 min in IDL).

```

>
>>
>
>> Now I know very well, that the problem is in the way the socket command is implemented... is there any way to get around this without "waiting" in IDL and read directly byte/char data in while loops?
>
>>
>
>>
>
>>
>
>> Maybe not an elegant solution, but that's how I see it done elsewhere.
>
>>
>
>>
>
>>
>
>> Many thanks,
>
>>
>
>> Helder
>
>
>
>
> Helder,
>
>
>
>
> When dealing with sockets, it's best to know how much data you are going to read. Most protocols with which I'm familiar either use fixed sizes or pass data lengths associated with entities first. In your example of using READF on a string, you might consider instead reading a BYTARR() of the appropriate size, then converting via STRING().
>
>
>
>
> Also take a look at the docs for the function FILE_POLL_INPUT, the /RAWIO keyword to SOCKET and the TRANSFER_COUNT keyword to READU. They might come in handy later on if you're working with "large" data and need to deal with network latency, if localhost isn't both your server and client.
>
>
>
>
> IDL 8.3 will have a cool new TIMER() static class that you can use to get around the need for

widgerts with timers, in the event that you need to read your data in chunks and don't want to block the main thread.

>
>
>
> Jim P.

Hi Jim,
thanks for the illumination ;-)
Well, now it's blowing fast. So, here's what I've changed.
After writing with
WRITEU, Unit, 'cmd '+STRTRIM(i,2)
(with i going from 1 to 100)

I use a function to read the string like this:
Value = ReadSocketString(Unit)

The function is the following:

```
FUNCTION ReadSocketString, Unit
Compile_Opt idl2
Catch, theError
IF theError NE 0 THEN BEGIN
    Catch, /Cancel
    void = Error_Message()
    RetError = 1
    RETURN, "
ENDIF
OneByte = BYTARR(1) + BYTE('0')
StrOut = "
WHILE OneByte NE 0 DO BEGIN
    ReceivedLen = 0L
    WHILE ReceivedLen EQ 0 DO BEGIN
        READU, Unit, OneByte
        ReceivedLen++
    ENDWHILE
    IF OneByte NE 0 THEN StrOut += STRING(OneByte)
ENDWHILE
RETURN, StrOut
END
```

Now reading about 100 strings does not take longer than 70 mSec and that's just great.

Summarizing: main change was the use of READU instead of READF and reading a Bytarr of 1 element (I don't know how long the string will be beforehand)
