Subject: Re: socket generated event
Posted by condor on Tue, 15 Apr 2003 17:44:34 GMT
View Forum Message <> Reply to Message

lejardi@sandia.gov (Lauren) wrote in message
news:<444b7c15.0304090649.679083fa@posting.google.com>...
> Is there any event that can be generated when there is data on the
> socket connection to be read? Or does my code just need to keep
> checking the socket for any information. Also, how is the information
> sent over the socket? Would it be possible for me to read only part of
> a message that the server is writing because it has not finished
> writing the message to the socket. Or are all messages on a socket
> sent in a whole packet. Then I would never get just a section of a
> message.

Caveat: I've never done socket programming in IDL. I have, however, done my share of network-transparent client-server stuff in other languages, and the socket mechanism is really not language dependent. Or at least I feel qualified to post about. the parts that aren't. :)

Events generated upon readability or writability of a socket are certainly language-dependent (or better: dependent on the implementation of the socket access, which at the bottom is certainly done in C...)

Partial information: this depends on the buffering by the server (assuming you're writing a client): The standard C stdio mechanism buffers full lines when talking to terminals, blocks when talking to anything else. However the application that sends data can decide to buffer(/send) line-by-line independent of the receiving application. Or it can do block-buffering, but explicitely flush the port at certain strategic points in the transmission. Or it can decide to send stuff byte-by-byte if it wants to. However if nothing else is said or done, you can expect stuff to arrive in blocks, which are often 8192bytes (but can be other sizes!) which usually don't happen to end conveniently on a line-ending.

Glancing over the IDL documentaion, it might be easiest to insist on reading whole lines (s='' & readf, unit, sc) with some reasonable read_timeout and catching/processing all errors that IDL might throw. The documentation for 'socket' does not look as if RSI intended for users to do much socket I/O in IDL -- and the documentation for IOCT makes it clear that you're not supposed to play with that interface at all. You might just be better off doing the networking stuff in a language that is geared towards that kind of thing (like TCL) -- but that depends on the overall complexity of your problem of course.