

Re: Palatable Windows IO using Ada

Source: <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2006-04/msg00073.html>

- *From:* "Dmitry A. Kazakov" <mailbox@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Mon, 10 Apr 2006 17:05:33 +0200
-

On Mon, 10 Apr 2006 06:50:53 -0700, Steve wrote:

"Dmitry A. Kazakov" <mailbox@xxxxxxxxxxxxxxxxxxxxx> wrote in message [news:o2i2brhcp49x\\$.1lmgnrqgqzf7\\$.dlg@xxxxxxxxxxxxxxxxx](mailto:news:o2i2brhcp49x$.1lmgnrqgqzf7$.dlg@xxxxxxxxxxxxxxxxx)

How would you tell that to the driver? Do you mean a true driver here or merely a program that uses an OS serial driver? I meant the later...

So did I. On the systems I have used there have been systems calls "read until character or timeout" or at least a mechanism of implementing that functionality.

Then it should be "until delimiter or timeout or buffer is full." This is a too specific, IMO. For some devices you would need a pattern matching rather than simple delimiter. I don't think that it is worth to have something like that in OS. If protected objects were supported, one could just give a protected ring buffer to the driver.

I prefer two threads as well. It is my understanding that the only way to do this on NT is using overlapped I/O. I have this working in my high level interface to NT's serial port.

We used both overlapped and synchronous I/O over Windows serial port, with no noticeable difference. In almost all cases the communication was without either hardware handshake or parity, which should be a quite hard test, if OS wouldn't buffer. In my experience, a correct use of SetCommTimeouts is more delicate issue.

The clumsy interface has no relation to the choice of language of the API.

Re: Palatable Windows IO using Ada

And it shouldn't. It is based on the design of the driver. In the good (or bad?) old days, the OS interfaces used to be described in a language neutral manner... as it should be.

How can you do it language neutral? The problem is that asynchronous I/O inherently requires concurrency. If the language does not support concurrency, you simply cannot comprehensively describe what's going on.

Sure you can. You just have to be explicit about how concurrency is handled, and can't take anything for granted.

But you just cannot be explicit. You have to document, to comment etc, rather than to code. It is like types, you don't need typed languages. Just carefully document each memory cell. The only problem is the abstraction level.

[...]

Ada covers enough bases I only have to learn a small part of working with the OS. This comes back to my claim that you can develop a system in almost any programming language, given enough time and money... but if you want to minimize either, use Ada.

Actually there is a physical limit of complexity at which the amount of money becomes infinite. Though, long before that it will be bigger than the national product. With time it is even worse. I wouldn't even try to understand the Assembler code I wrote 20 years ago... (:-))

—
Regards,
Dmitry A. Kazakov
<http://www.dmitry-kazakov.de>

.