Subject: Re: IDL verses other interpertative languages (tcl/tk, khouros, pv_wave, etc).
Posted by rivers on Wed, 17 Dec 1997 08:00:00 GMT
View Forum Message <> Reply to Message

In article <3496CA0E.DD25021F@sandia.gov>, "Stuart E. Murray" <semurra@sandia.gov> writes:

> I am working on a Satellite Sensor Simulator and will require that I
> display several windows of data. I am trying to sort out what display
> package might be the best bet. My group currently uses IDL and tcl/tk
> and I have been asked to considered them since they are in house. I have
> talked with these folks about both languages and both have strong view
> points. It is obvious to me that this discussion group will also. But,
> for the sake of argument. I will pose this question on the tcl
> discussion group as well.

> 1. I will have a real-time connection to another embedded machine.
> 2. There will be one data spigot (be it parallel port, USB, serial port,
> or network) to the other embedded machine. Does IDL support interfaces
> such as those or do I have to write my own drivers?
> 3. Has anyone used IDL in a real-time environment successfully (or
> unsuccessfully)?

We are doing exactly those types of applications at a number of synchrotron radiation beamlines at Argonne and Brookhaven National Labs. IDL is successfully (and easily) controlling and displaying data from many real-time experiments. IDL can call C code directly, and it is really very easy (the hardest part is usually learning how to create shareable libraries under your operating system).

The types of "data spigots" we are using include
- VME crates running vxWorks and communicating on the network via TCP/IP
- CAMAC crates with direct computer bus interfaces
- Multichannel analysers with network interfaces
- Frame grabbers with computer bus interfaces

You get the picture - if you can talk to the device from C you can talk to it from IDL. For the kinds of experiments we do, which are frequently reconfigured IDL is really nice, since it is a nice scripting language.
The recent addition of object oriented techniques makes it even more powerful.
Here is an example: the IDL class 'EPICS_MOTOR' talks to stepping motors which are controlled in a VME crate running vxWorks and the EPICS control system.
Here is all that a user has to program to move a motor and wait for it to get to its destination:

```
motor = obj_new('EPICS_MOTOR')
motor->move, 10.
```

motor->wait

> 3.1 The display of data in real-time may not be a strong issue if you
> can throw a lot of horsepower at the problem. However, since IDL is an
> interpertative language, you have to wonder about the impact of
> performance over straight compiled code? Is this a problem? Is Windows a
> problem?

IDL graphics are very fast, as fast as most compiled packages, because typically just a few interpreted statements are required to produce a complete graphics display. The real work is done by IDL internal routines, which are compiled C code.

We use IDL for >10 Hz display of spectral (1-D data) and for real-time microscopy (imaging).

> 4. How hard is it to incorporate C/C++ code into IDL? Literature
> suggests a thorough knowledge of IDL before attemping this and I don't
> have that yet.

See above. It is NOT difficult. Basically this line is what you need:
result = call_external('MY_LIBRARY', 'MY_ENTRY_POINT', param1, param2, ...)

You just need to create MY_LIBRARY as a shareable library with your C compiler with MY_ENTRY_POINT globally visible.

> 5. IDL will probably be executed on a Laptop (166 MHz or better) under
> Win95 or NT (the OS may have a Real-Time exec handler, although there
> are people that would question a Real-Time exec handler can be done
> correctly). Other real time OSs might be considered like VxWorks, QNX,
> etc.

I am not sure I indestand you here - IDL cannot run under vxWorks or other real-time OS. I use it under Windows NT, VAX/VMS, AXP/VMS, and many Unix flavors (sun4, Solaris, Digital Unix, IRIX, HPUX) and am doing real-time data collection and control on all of these platforms.

6. IDL learning curve steep?

It is different, but IMHO a lot more familiar to most programmers than tcl.

> 7. What is the downside of IDL? (bloat code, etc).
> 8. IDL cost (both for the software & maintenance of code)? How easy is
> it for someone to come in and pick up where you have left off?

Maintenance is not a big issue. IDL is expensive, but people are much more so. I have no trouble justifying IDL cost in terms of save man-hours compared to third generation languages. It really shines when you want to try things

quickly.

Mark Rivers (773) 702-2279 (office)
CARS (773) 702-9951 (secretary)
Univ. of Chicago (773) 702-5454 (FAX)
5640 S. Ellis Ave. (708) 922-0499 (home)
Chicago, IL 60637 rivers@cars.uchicago.edu (e-mail)

or:
Argonne National Laboratory (630) 252-0422 (office)
Building 434A (630) 252-0405 (lab)
9700 South Cass Avenue (630) 252-1713 (beamline)
Argonne, IL 60439 (630) 252-0443 (FAX)

Subject: Re: IDL verses other interpertative languages (tcl/tk, khouros, pv_wave, etc).
Posted by Peter Mason on Wed, 17 Dec 1997 08:00:00 GMT
View Forum Message <> Reply to Message

On Tue, 16 Dec 1997, Stuart E. Murray wrote:
<...about writing a Satellite Sensor Simulator>

As much as I like IDL, I don't think that it is the right tool for this kind of task.

I haven't done this sort of thing in IDL, but I think that it may be tricky. You would have to link to an external library (e.g., a Windows DLL) to access data acquisition boards and maybe even parallel/serial ports. (IDL doesn't support port access up front. It may be possible using IDL's I/O or DEVICE routines in some sly way, but I haven't tried.) Real-time may be an issue if you have a fair amount of processing, and it certainly will be a tough issue (perhaps impossible?) if you want to act on interrupts instead of polling things. (And the timer resolution for polling might not be good enough.) And so on.

I have a colleague who is designing software to operate a hyperspectral profile scanner. (He's also designing much of the scanner itself.) Data acquisition, a fair amount of processing, real-time displays, etc. He's using a Windows-based "thing" called LabView. I say "thing" because it's quite unlike any other application I've seen. It's a "graphical programming environment for instrumentation" - one can generate very sophisticated programs with it, without having to type a line of code. Apparently it's very popular for this kind of work. My colleague can't seem to say enough good things about it. Indeed he has quite an evangelical attitude towards it, and wouldn't give up on catching me at teatime and remarking about "LabView" until I had spent a

morning with him being (genuinely) impressed by its power and ease-of-use.

I understand that a demo version is available.
Check out: http://www.labview.com

Cheers

Peter Mason
CSIRO division of Exploration and Mining
P.O Box 136, North Ryde, NSW, 2113, Australia
E-Mail: p.mason@syd.dem.csiro.au Tel: +61 2 9490-8883 Fax: 9490-8960/8921
Web: http://www.syd.dem.csiro.au/research/MMTG/