Subject: Re: IDL interpreter questions - can someone (D.Fanning) explain - TIA
Posted by John-David T. Smith on Fri, 18 May 2001 16:21:41 GMT
View Forum Message <> Reply to Message

dadada wrote:
>
> Topics in question:
>
> How are variables referenced by default?

I'm not sure what you mean here. Pointer references? They are explicit only... i.e. you can't create a reference of an existing variable.

> Does IDL have first class procedures?

Only in-as-much as run-time evaluation is allowed. So the answer is no, I guess. You can call methods and routines by string name... e.g.

call_procedure,'mypro',arg1,arg2

The overhead is only slightly higher than real anonymous functions, but the syntax obviously isn't transparent (i.e. you always *know* you're calling a procedure by reference to it's name, vs directly).

> Continuations?

No continuations. Non-local exits from recursions are possible using standard argument passing, or common blocks.

> Suspensions (ie. thunks)?

No thunks or closures.

> How is IDL interpreted in relation to Scheme or LISP?

IDL is a very straightforward procedural language similar to FORTRAN, and lately with more and more C-like paradigms, but with vector operator overloading and built-in, convenient support for arrays of up to 8 dimensions (once you learn the peculiarities of slinging them around). It is a data-based, not function-based language, and sometimes it really shows. A simple object-oriented framework was added in the last few years that introduces very little complexity, and trades features for ease of development.

In part because of this simple design, IDL is actually quite fast, and the interpreter can assemble large chunks of precompiled code after the first pass.

If you are comfortable in C and/or FORTRAN or other imperative languages, you will be utterly amazed at the speed with which you can accomplish things in IDL.

If you are a LISP/SCHEME or other functional language guru, you will wonder how anyone can get anything done in a language so devoid of higher-level constructs (a lambast you'd also levy against C, FORTRAN, and the rest). If you are a hardcore OO programmer (C++, Smalltalk, Objective-C, Java), you might feel IDL's OOP is somewhat toylike. If you are a guy with a bunch of data, probably without a CS degree, and a need to process, visualize, and combine your data, you'll feel right at home in IDL.

JD

Subject: Re: IDL interpreter questions - can someone (D.Fanning) explain - TIA
Posted by <mankoff[1] on Fri, 18 May 2001 16:39:59 GMT
View Forum Message <> Reply to Message

On Fri, 18 May 2001, JD Smith wrote:

> dadada wrote:
>> How are variables referenced by default?
> I'm not sure what you mean here. Pointer references? They are explicit
> only... i.e. you can't create a reference of an existing variable.

Not sure either, but here is my interpretation of the question/answer:

In functions, variables are *always* 'by value'
In procedures, they are 'by value' unless you put a "return" statement anywhere in the procedure. If this exists, then they are passed 'by reference'

>> Continuations?
> No continuations. Non-local exits from recursions are possible using
> standard argument passing, or common blocks.

isn't this beginning to be a part of the language? They just added "break" and a few other C style conventions. I would expect to see a 'continue' in one of the upcoming releases. But JD is right, it doesn't exist yet.

>> Suspensions (ie. thunks)?
> No thunks or closures.
what is a thunk?

I searched briefly and read that it is a "zero argument closure", but that doesn't clarify much...

-k.

--
Ken Mankoff
LASP://303.492.3264
http://lasp.colorado.edu/~mankoff/