Subject: Re: Interesting Rant
Posted by Paul Van Delst[1] on Tue, 14 Nov 2006 18:36:47 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Folks,
>
> Someone sent me a link to this interesting IDL rant this morning:
>
>   http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html

Hee hee. I thought it was pretty funny. I probably would've done something similar back when I was one of dem young whippersnappers (well, I would've if anything but Fortran was available....)

Apart from the fact that some of the info was just wrong, the rant shows the writers lack of experience with programming languages in general. To say nothing of exiting college and entering the real world where being able to distinguish and effectively handle the differences between the the way things *should* be and the way they actually *are* are a definite plus.

Ah... the innocence and passion of youth. :o)

Seeing as the ranter states he/she is a CS major, I think the following quote is somewhat topically pithy (seeing as the original article was brought to my attention on this newsgroup... I think):

"Formal logical proofs, and therefore programs ï¿½ formal logical proofs that particular computations are possible, expressed in a formal system called a programming language ï¿½
are /utterly meaningless/. To write a computer program you have to come to terms with this, to accept that whatever you might want the program to mean, the machine will blindly follow its meaningless rules and come to some meaningless conclusion. In the test the consistent group [good programmers] showed a pre-acceptance of this fact: they are capable of seeing mathematical calculation problems in terms of rules, and can follow those rules wheresoever they may lead. The inconsistent group [so-so programmers], on the other hand, looks for meaning where it is not. The blank group [bad programmers] knows that it is looking at meaninglessness, and refuses to deal with it."

From: www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

If one can program well in IDL, I reckon one's "utterly meaningless" detector must be pretty well calibrated. :oD

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Interesting Rant
Posted by news.qwest.net on Tue, 14 Nov 2006 18:56:51 GMT
View Forum Message <> Reply to Message

"David Fanning" <news@dfanning.com> wrote in message
news:MPG.1fc3ad90bc2fc367989dbb@news.frii.com...
> Folks,
>
> Someone sent me a link to this interesting IDL rant this morning:
>
> http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html

N00b. N00b indeed.

Thanks for the amusing lunchtime read.
It is more of a list of things the ranter did wrong, and the complaint
is that IDL let him. That kind of attitude will serve him well in his
future as a programmer. Look for his future exploits at thedailywtf.com

The fault dear stephen, lies not in IDL, but in ourselves. :)

Cheers,
bob

Subject: Re: Interesting Rant
Posted by David Fanning on Tue, 14 Nov 2006 19:09:00 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> are /utterly meaningless/. To write a computer program you have to come to terms with
> this, to accept that whatever you might want the program to mean, the machine will blindly
> follow its meaningless rules and come to some meaningless conclusion. In the test the
> consistent group [good programmers] showed a pre-acceptance of this fact: they are capable

> of seeing mathematical calculation problems in terms of rules, and can follow those rules
> wheresoever they may lead. The inconsistent group [so-so programmers], on the other hand,
> looks for meaning where it is not. The blank group [bad programmers] knows that it is
> looking at meaninglessness, and refuses to deal with it."
>
> From: www.cs.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

I didn't know scientific papers like this were written
any more. What a delight! And my own personal experience
teaching IDL programming courses confirms that their
analysis is dead-on accurate. :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Interesting Rant
Posted by Richard Edgar on Tue, 14 Nov 2006 19:13:14 GMT
View Forum Message <> Reply to Message

Paul van Delst wrote:

>> Someone sent me a link to this interesting IDL rant this morning:
>>
>> http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html
>
> Hee hee. I thought it was pretty funny. I probably would've done
> something similar back when I was one of dem young whippersnappers
> (well, I would've if anything but Fortran was available....)
>
> Apart from the fact that some of the info was just wrong, the rant shows
> the writers lack of experience with programming languages in general. To
> say nothing of exiting college and entering the real world where being
> able to distinguish and effectively handle the differences between the
> the way things *should* be and the way they actually *are* are a
> definite plus.

If I were to write my list of IDL annoyances, I think mine would be
somewhat different to this.

I'd comment that the first one, about 7/2 vs 7/2.0 is a place where I'd
say that IDL definitely does the right thing. And comment 12 about

comparisons contradicts the first comment (and seems to show a lack of understanding of floating point arithmetic).

Row-major vs column major is a silly point... you just need to know which way the language does it, and that's the end of the matter. And I'd dispute the bit about 'every other language in the history of mankind' too ;-)

There is some basis for the comments about the 'compiler' and need for recompilation. I'd say that this is due to a bit of bad nomenclature on IDL's part, since IDL is more an interpreted than a compiled language.

The complaint about inconsistency in how variables are passed to routines is fair enough. ISTR my encounter with this was based on passing a structure vs. passing elements of the structure. However, the passing method is irrelevant. All I should have to do is declare whether I intend to modify the variables or not, and whether they should be defined on entry. Something like INTENT(IN), INTENT(OUT) and INTENT(INOUT) would be ideal ;-)

Richard

Subject: Re: Interesting Rant
Posted by James Kuyper on Tue, 14 Nov 2006 23:25:55 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Folks,
>
> Someone sent me a link to this interesting IDL rant this morning:
>
>   http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html
>
> This is what happens, I suppose, when the language
> you are using is older than you are. Very little
> historical perspective. But it may go a little way
> towards explaining why people coming out of college
> seem to prefer Matlab by a large margin.
>
> I disagree with much of what he has to say, but I
> can't tell whether this is because I am old enough
> to remember what things were like *before* IDL, or
> whether I've just become inured over the years. :-(

His point about colum-major ordering proves that he's not familiar with Fortran. However, I'm not personally familiar with any languages other than IDL and Fortran that handle it that way; does anyone else know of any?

Subject: Re: Interesting Rant
Posted by Robbie on Wed, 15 Nov 2006 01:13:56 GMT
View Forum Message <> Reply to Message

He is just pointing out the obvious. IDL appears to be a haphazardly
defined language. Many things are simply historical, and no one company
or person has been able to gain enough control to make things right.
Python is a good example of a language which has been built properly,
but it's lacking the tools and speed of IDL. I'm very excited about the
Python Array interface (http://numpy.scipy.org/array_interface.shtml).
I would really like to see Visual Numerics or ITT create a fast package
which works with the array interface. I'm sure that it is never going
to happen anyway.

Matlab suffers from the same fate. The toolboxes are Matlab's killer
apps.

Java, C++ and C# do not have a standard array interface which I would
use. I can still use them for image processing, but I must wrap
everything up into objects or I must send my head spinning with
templates.

Subject: Re: Interesting Rant
Posted by Maarten[1] on Wed, 15 Nov 2006 13:25:54 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> I disagree with much of what he has to say, but I
> can't tell whether this is because I am old enough
> to remember what things were like *before* IDL, or
> whether I've just become inured over the years. :-(

On the contrary, I'd like to imagine what things can become *after*
IDL. While I would list a different set of annoyances, I'd say you have
become inured to the IDL idiosyncrasies over the years. Lovely word,
"inured", had to look it up though (can you tell I'm not a native
speaker?)

* Arrays in structures can not be resized. This also applies to
objects.
* You can have resizable arrays in structures, if you use pointers. The
way to access data in the array changes, you'll have to rewrite your
code.

* Pointers that can get lost in a scripted language. If I want a memory leak, I'll use C, Fortran, ...
* Default integer size is 16 bits (how long have computers been at 32bits by now; how much code will break if you change that to 32 (or even 64 bits)? How can code break on such a change? I think the code was broken in the first place if it relies on this bit-size.
* Individual floating point constants are float, not double.
* Direct graphics seem to be dead, object graphics are not practical for interactive use. Hello, the I in idl stands for interactive, right?
* Procedure arguments as output parameter and array elements.
* Brain dead for loops. Nice to have a vector engine (which doesn't work in all cases, idl_validname() only accepts strings, not string arrays, so for loops cannot be avoided in all cases), but sometimes an explicit for loop aids readability (histogram jugling, anyone?).
* Stupid end of line behaviour, especially on interactive use.
* Logical test on least significant bit only
* The UI on Mac OS X. Using X11 on Mac OS X does not make a port.
* Special characters in graphs, combined with the butt ugly Hershey fonts. The graphs look old-fashioned, and like they were made with some homebrew software, rather than a professional, expensive tool. Yes, you can do better in IDL, many don't.

So while IDL probably is an improvement over what came before it (I'm too young to really tell), those programs are truly dead and buried. IDL seems to survive, despite better solutions, especially for interactive use. The only things that keep idl alive are the dinosaurs that use it, and the legacy code that has been written for it. Others have suggested to use Python as a basis, and I think I agree, although some array indexing issues will be as annoying as IDL. At least there is a large community behind it, and the core of python is free.

Maarten

Subject: Re: Interesting Rant
Posted by David Fanning on Wed, 15 Nov 2006 14:10:16 GMT
View Forum Message <> Reply to Message

Maarten writes:

> The only things that keep idl alive are the dinosaurs
> that use it, and the legacy code that has been written for it.

Now just a second here.... Oh, never mind. I'm going back to bed. :-(

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Interesting Rant
Posted by K. Bowman on Wed, 15 Nov 2006 14:47:08 GMT
View Forum Message <> Reply to Message

In article <1163597154.282197.250510@m7g2000cwm.googlegroups.com>,
 "Maarten" <maarten.sneep@knmi.nl> wrote:

> * Default integer size is 16 bits (how long have computers been at
> 32bits by now; how much code will break if you change that to 32 (or
> even 64 bits)? How can code break on such a change? I think the code
> was broken in the first place if it relies on this bit-size.

This one, at least, can be fixed easily. Just put

COMPILE_OPT IDL2

in your startup.pro.

I include that line in *all* procedures and functions.

> * Individual floating point constants are float, not double.

I'm happy with that the way it is, but I expect they could add a

   DEFFLOAT64

option to COMPILE_OPT if enough people requested it.

> * Direct graphics seem to be dead, object graphics are not practical
> for interactive use. Hello, the I in idl stands for interactive, right?

I can't say that I've been born again (yet), but I am finding the iTools
to be very useful for interactive graphics. The user interface is
awkward in several ways, the learning curve is steep, and hardcopy output
remains a real problem, but there are a number of really handy features
in the iTools.

Ken

Subject: Re: Interesting Rant
Posted by greg michael on Wed, 15 Nov 2006 15:03:17 GMT
View Forum Message <> Reply to Message

I've used IDL for about three years - I think that makes me a
relatively new user, at least in IDL history. In my early experience
with it, I was frequently frustrated by its incoherent design -
especially the completely arbitrary (and awful) naming of built-in
procedures (also that strange comma/bracket distinction for
functions/procedures). But these are really only cosmetic things - once
you've written something in a dozen lines that took a hundred in
another language, there's no turning back.

The complaints about IDL loops seem to me to have missed the point - my
old hoards of triple-nested loops with the inevitable bugs to be fixed
around the end conditions have disappeared forever. You don't have to
think about individual elements *ever*, until you want one. This is the
single reason for me that makes it better than any other language I've
used.

It's true the graphics are either ugly or awkward, and there are
atrociously deficient functions like read_ascii() lying around, but
with some experience you can overcome these, and the benefits of the
array handling easily make it worth it. I like very much that you can
make platform-independent code, too. Would be nice, of course, if a new
version turns up one day with plots that look decent. Will have to come
eventually - if not in IDL, then the next language. I'm not a dinosaur
- I'd change to something better, but I haven't seen it yet.

regards,
Greg

Subject: Re: Interesting Rant
Posted by Maarten[1] on Wed, 15 Nov 2006 15:17:27 GMT
View Forum Message <> Reply to Message

Kenneth Bowman wrote:
> In article <1163597154.282197.250510@m7g2000cwm.googlegroups.com>,
>   "Maarten" <maarten.sneep@knmi.nl> wrote:
>
>> * Default integer size is 16 bits (how long have computers been at
>> 32bits by now; how much code will break if you change that to 32 (or
>> even 64 bits)? How can code break on such a change? I think the code
>> was broken in the first place if it relies on this bit-size.
>
> This one, at least, can be fixed easily. Just put
>

> COMPILE_OPT IDL2
>
> in your startup.pro.
>
> I include that line in *all* procedures and functions.

Have it, do it, well something similar:
    compile_opt defint32, strictarr, strictarrsubs
(the first two are the same as idl2). Still I would like it to be the
default for *all* functions. I mean, IDL2 came out how long ago?

>> * Individual floating point constants are float, not double.
>
> I'm happy with that the way it is, but I expect they could add a
>
>   DEFFLOAT64
>
> option to COMPILE_OPT if enough people requested it.

That would be nice. Where was that feature request box again?

>> * Direct graphics seem to be dead, object graphics are not practical
>> for interactive use. Hello, the I in idl stands for interactive, right?
>
> I can't say that I've been born again (yet), but I am finding the iTools
> to be very useful for interactive graphics. The user interface is
> awkward in several ways, the learning curve is steep, and hardcopy output
> remains a real problem, but there are a number of really handy features
> in the iTools.

Perhaps, but as long as my primary task is to produce printable output
as well, I don't feel the need to learn *two* graphing methods. Just
give me *one* that really works. WaveMetrics got it right with Igor
around 1990, so what is ittvis waiting for?

Maarten

Subject: Re: Interesting Rant
Posted by Jo Klein on Wed, 15 Nov 2006 15:20:55 GMT
View Forum Message <> Reply to Message

> I can't say that I've been born again (yet), but I am finding the iTools
> to be very useful for interactive graphics. The user interface is
> awkward in several ways, the learning curve is steep, and hardcopy output
> remains a real problem, but there are a number of really handy features
> in the iTools.
I have started using the iTools only quite recently, so maybe I just

haven't got the hang of it quite yet. I think they're handy for quick
interactive data visualisation, but:
- they are incredibly slow to start up
- the documentation leaves a lot (and I mean a big, big lot) to be
desired (new book chapter please, David)
- it just takes too many lines of code to trim down their humongous UI
to something useful
I'd much rather have a few iTools features available for direct
graphics, like resizable plots, proper labelling ...
Well, just my opinion,
Jo

Subject: Re: Interesting Rant
Posted by Braedley on Wed, 15 Nov 2006 15:47:26 GMT
View Forum Message <> Reply to Message

His comments about the 7/2 by themselves tell me that he doesn't know
jack about programming. I know for sure that at least 2 of the 4
competing languages (the C family, Java and possibly Maltab) all do the
exact same thing as IDL. EXACT! (Okay, maybe Matlab doesn't, I
haven't used it in 8 months.) The fourth is Maple, and it may still do
the same thing as IDL (it's been so long since I did simple math in
Maple as well). His complaint is utterly baseless.

I'd also like to point out that Java isn't entirely consistent in how
it passes variables to functions. If memory serves, structures and
objects, and pointers by extension are the only things passed by
reference, whereas everything else is passed by value. This caused
much fussing on my part during my first year programming courses, since
I was coming from high-school where I learned C++. I still resent Java
to this day for that reason. IDL is much more consistent. None of the
stupid "This is an int, so it's by value, that's an object, so it's by
reference," which makes sense from an engineering standpoint, but not
from a programming standpoint.

Richard Edgar wrote:
> Paul van Delst wrote:
>
>>> Someone sent me a link to this interesting IDL rant this morning:
>>>
>>>   http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html
>>
>> Hee hee. I thought it was pretty funny. I probably would've done
>> something similar back when I was one of dem young whippersnappers
>> (well, I would've if anything but Fortran was available....)
>>
>> Apart from the fact that some of the info was just wrong, the rant shows

>> the writers lack of experience with programming languages in general. To
>> say nothing of exiting college and entering the real world where being
>> able to distinguish and effectively handle the differences between the
>> the way things *should* be and the way they actually *are* are a
>> definite plus.
>
> If I were to write my list of IDL annoyances, I think mine would be
> somewhat different to this.
>
> I'd comment that the first one, about 7/2 vs 7/2.0 is a place where I'd
> say that IDL definitely does the right thing. And comment 12 about
> comparisons contradicts the first comment (and seems to show a lack of
> understanding of floating point arithmetic).
>
> Row-major vs column major is a silly point... you just need to know
> which way the language does it, and that's the end of the matter. And
> I'd dispute the bit about 'every other language in the history of
> mankind' too ;-)
>
> There is some basis for the comments about the 'compiler' and need for
> recompilation. I'd say that this is due to a bit of bad nomenclature on
> IDL's part, since IDL is more an interpreted than a compiled language.
>
> The complaint about inconsistency in how variables are passed to
> routines is fair enough. ISTR my encounter with this was based on
> passing a structure vs. passing elements of the structure. However, the
> passing method is irrelevant. All I should have to do is declare whether
> I intend to modify the variables or not, and whether they should be
> defined on entry. Something like INTENT(IN), INTENT(OUT) and
> INTENT(INOUT) would be ideal ;-)
>
> Richard

Subject: Re: Interesting Rant
Posted by Paolo Grigis on Wed, 15 Nov 2006 17:02:27 GMT
View Forum Message <> Reply to Message

Braedley wrote:
> His comments about the 7/2 by themselves tell me that he doesn't know
> jack about programming. I know for sure that at least 2 of the 4
> competing languages (the C family, Java and possibly Maltab) all do the
> exact same thing as IDL. EXACT! (Okay, maybe Matlab doesn't, I
> haven't used it in 8 months.) The fourth is Maple, and it may still do
> the same thing as IDL (it's been so long since I did simple math in
> Maple as well). His complaint is utterly baseless.

I think in Maple 7/2 evaluates to seven halves...

Ciao,
Paolo

>
>

Subject: Re: Interesting Rant
Posted by Earl F. Glynn on Wed, 15 Nov 2006 17:07:33 GMT
View Forum Message <> Reply to Message

"Kenneth Bowman" <k-bowman@null.tamu.edu> wrote in message
news:k-bowman-1096B8.08470815112006@news-new.tamu.edu...
> In article <1163597154.282197.250510@m7g2000cwm.googlegroups.com>,
> "Maarten" <maarten.sneep@knmi.nl> wrote:
>
>> * Default integer size is 16 bits (how long have computers been at
>> 32bits by now; how much code will break if you change that to 32 (or
>> even 64 bits)? How can code break on such a change? I think the code
>> was broken in the first place if it relies on this bit-size.
>
> This one, at least, can be fixed easily. Just put
>
> COMPILE_OPT IDL2
>
> in your startup.pro.
>
> I include that line in *all* procedures and functions.

I've been using IDL for about a year (but have programmed for years in a
variety of languages ... even back to FORTRAN), and have only wanted 32-bit
integers from day 1 in IDL. It's a pain and an annoyance not to have 32-bit
integers as a default -- just like IBM mainframes provided in the early 70s
with FORTRAN. [I'm trying to get back to where I started.]

efg

Earl F. Glynn
Scientific Programmer
Stowers Institute for Medical Research

Subject: Re: Interesting Rant
Posted by Earl F. Glynn on Wed, 15 Nov 2006 17:19:58 GMT
View Forum Message <> Reply to Message

"Richard Edgar" <rge21@pas.rochester.edu> wrote in message
news:ejd4g4$416$1@mail.rochester.edu...
> Paul van Delst wrote:
>
>>> Someone sent me a link to this interesting IDL rant this morning:
>>>
>>> http://www.sccs.swarthmore.edu/users/07/sstvinc2/research/st upid.html

> Row-major vs column major is a silly point... you just need to know
> which way the language does it, and that's the end of the matter. And
> I'd dispute the bit about 'every other language in the history of
> mankind' too ;-)

Perhaps a bit exaggerated above, but I don't think this is a silly point at
all, especially when one must switch between programming languages to
maintain software. Perhaps, I'm wrong, but IDL's arrays seem to be unique,
and unlike C or FORTRAN.

C: row-major format, [nrows, ncolumns], 0-origin
"Elements are stored in rows, that is, the rightmost subscript varies
fastest as elements are accessed in storage order." Kernighan & Ritchie,
The C Programming Language.

```
X[0][0]  X[0][1]  X[0][2]
X[1][0]  X[1][1]  X[1][2]
...
```

The data are stored in memory from left-to-right across a row, with rows
ordered from top to bottom.

FORTRAN or R: column-major format [nrows, ncolumns], 1-origin

```
X[1,1]  X[1,2]  X[1,3]
X[2,1]  X[2,2]  X[2,3]
...
```

The data are stored in memory from top-to-bottom across a column, with
columns ordered from left to right.

But IDL doesn't follow EITHER of these models completely and is strangely
unique:
IDL: column-major format, [ncolumns, nrows], 0-origin

IDL Array Storage and Indexing

http://www.dfanning.com/misc_tips/colrow_major.html

"Arrays in IDL are stored in row order which means the first index varies
the fastest. Therefore, whenever you write any kind of loop that accesses
an array try to vary the first element the fastest." P 1-11, Ronn Kling,
"Application Development in IDL"

```
X[0,0]  X[1,0]  X[2,0]
X[0,1]  X[1,1]  X[2,1]
...
```

The data are stored in memory from left-to-right across a row, with rows
ordered from top to bottom. SO, IDL is just like C if you reverse the order
of the subscripts. Forget about that reversal if you're thinking in C, and
there's a bug in your code.

But when dealing with images, IDL arrays really go from bottom to top?

```
...
X[0,1]  X[1,1]  X[2,1]
X[0,0]  X[1,0]  X[2,0]
```

Or did I miss something about how IDL stores arrays in memory?

The lack of a standard convention here could easily contribute to
programming errors, especially if one uses IDL and other programming
languages. Converting code from other languages could also be problematic
because of silly mistakes getting subscripts right.

This may not be a problem if IDL is the only language one uses, but if one
works with a variety of programming languages this uniqueness is not
desirable (at least to me).

efg

Earl F. Glynn
Scientific Programmer
Stowers Institute for Medical Research

Subject: Re: Interesting Rant
Posted by fburton on Wed, 15 Nov 2006 18:24:06 GMT
View Forum Message <> Reply to Message

In article <1163605646.246847.326220@i42g2000cwa.googlegroups.com>,
Braedley <mike.braedley@gmail.com> wrote:

> His comments about the 7/2 by themselves tell me that he doesn't know
> jack about programming. I know for sure that at least 2 of the 4
> competing languages (the C family, Java and possibly Maltab) all do the
> exact same thing as IDL. EXACT! (Okay, maybe Matlab doesn't, I
> haven't used it in 8 months.) The fourth is Maple, and it may still do

In MATLAB, both 7/2 and 7/2. evaluate to 3.5 (displayed as
3.5000 by default).

Imho, Delphi/Pascal does the sensible thing by requiring the
programmer to distinguish between real and integer division
through use of separate / and div binary operators, leading
to fewer nasty surprises.

Francis

Subject: Re: Interesting Rant
Posted by David Fanning on Wed, 15 Nov 2006 18:44:33 GMT
View Forum Message <> Reply to Message

Richard Edgar writes:

> If I were to write my list of IDL annoyances, I think mine would be
> somewhat different to this.

I think most of the items on his list fall into the "Short on
Time and Not What I'm Used To" whining category most of us fall into
sooner or later. (I'm thinking of my own issues with iTools.)

In the old days, you could count on about 1/3 of
the people in an IDL programming course succumbing
to these kinds of problems. These days (shorter attention
span of the younger generation? too may video games?) the
proportion seems much higher than this. Maybe there just
aren't that many "programmers" these days. Everyone
has caught on to the fact that nanotechnology is the
wave of the future.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: Interesting Rant
Posted by Paul Van Delst[1] on Thu, 16 Nov 2006 16:02:38 GMT
View Forum Message <> Reply to Message

Robbie wrote:
> He is just pointing out the obvious. IDL appears to be a haphazardly
> defined language. Many things are simply historical, and no one company
> or person has been able to gain enough control to make things right.
> Python is a good example of a language which has been built properly,

Regarding properly built languages, if you like Python, take a lookee at Ruby. Veddy nice... and I can indent as much or as little as I want! :o)

I wonder how the decisions were/are made at RSI/ITTVIS regarding the routine naming and syntax rules. Formally standardised languages (e.g. Fortran90/95/2003, C89/99, etc) typically lag behind (sometimes a wee bit too much) the bleeding edge of CS innovations -- and that's usually considered a feature. When the neato programming trick du jour is validated with years of use/trial/error, it's included in the standard with (hopefully) sufficient care taken to not break existing legacy code. "Community" standardised languages (e.g. Ruby, and I presume Python, Perl, etc) have a faster turnaround, but I think the end result/want/need is the same - more useful (as oppoed to just flashy) features but don't break old code. Proprietary language (e.g. IDL, Matlab) standardisation.... well, I don't know how they do it. There is no formal user community input sought by as far as I can tell (for IDL at least, dunno about matlab). I don't think a marketing type asking "What do you want/need?" fully qualifies.

Anyway...

cheers,

paulv

--
Paul van Delst Ride lots.
CIMSS @ NOAA/NCEP/EMC Eddy Merckx
Ph: (301)763-8000 x7748
Fax:(301)763-8545

Subject: Re: Interesting Rant
Posted by Jeff Hester on Sat, 18 Nov 2006 21:42:44 GMT
View Forum Message <> Reply to Message

There is an interesting generational aspect to this discussion. I grew up during a fairly narrow window of time when data analysis (and science more generally) was becoming more and more computerized, but before specialized software had become available. During that time one *had* to just write the code yourself, whether it was data I/O, array manipulations, numerical code, or even primitives that allowed you to talk to an image display device.

In that environment you had to be cognizant about everything from how information was stored at the bit level to the way memory was utilized to things like the efficiency differences between stepping through an array with *array++ as opposed to array[i], or the tradeoffs between efficiency and flexibility in data storage formats. (I can't count the number of times as a graduate student or postdoc that I was handed a tape and told, "Get the data off of it," without so much as a suggestion as to what the format might be.) You thought in terms of, "What would I like to do to/with my data, and how can I do it?" And when you found that a door was shut, you didn't think twice before starting to look around for the nearest window. Data processing and analysis meant getting your hands dirty. Period. Those of you who came of age in such an environment know what I mean.

As new tools came along, we were happy to learn to use them. It was kind of a relief to not have to write *everything* yourself, and there is no way that one person can keep up with a whole world full of programmers. But at the same time, we carry that early mindset with us. When we use a tool we can't help but build a mental picture of what that tool must be doing internally. We've got a pretty good guess about what is there inside the black box, and how it might go wrong.

It came as a shock to me when I realized the extent to which this mindset has vanished. Don't get me wrong. There are a lot of very good programmers coming up through the ranks. But the majority of people who sit down in front of a data set these day begin with the the question, "What convenient tools has someone written for me, and which might be best to apply to these data?" In other words, their thinking about problems tends to be heavily shaped by the parameters of their software environments. A lot of those tools are exceedingly powerful and allow you to do great stuff, often very efficiently. But a cage is still a cage, no matter how gilded.

Which arrives at the issue at hand. While I have developed a number of "packages" in IDL, for the most part every data set that comes across my desk is unique. Probably 75% of the code that I write is for the immediate purpose, and may not see use again after that day, much less after the end of that project. I include image and other data processing, numerical modeling, and data visualization in that category. I tend to cut, paste, hack, modify, stick together with duct tape, and in general do whatever I need to in an effort to tease information from a data set. Some of the code that I write is admittedly kind of ugly... but it shows me a lot of really interesting things in my data.

This is the kind of application that IDL was built for ("Interactive

Data Language..."), and IDL remains at least for me a more productive environment than any other I have run across. Is it the best programming language out there? No. Does it produce the prettiest plots with the spiffiest fonts? No. Do I use other tools? Sure. But when it comes time to get down and dirty with your data, IDL's combination of power and on-the-fly flexibility is hard to beat.

So when students complain, "I can't do that in Maple/Matlab/Whatever", I chuckle, then point out that quite often those who are limited only by their imagination and creativity tend to win out over those who are limited by the flexibility of their software. I then point them to an IDL tutorial.

BUT... as for personal pet peeves, it is the inefficiency of loops, hands down. (I will now resist the temptation of telling stories about compiling C code and linking it into Forth kernels to do real-time instrument control or doing image processing on a PDP 11/55....)

Surviving to be a dinosaur sure beats hell out of the alternative... ;-)

Cheers,
Jeff Hester

Professor and Dinosaur in Residence
School of Earth and Space Exploration
Arizona State University