Subject: Re: IDL 8.0 compile_opt changes
Posted by penteado on Sat, 19 Dec 2009 02:39:08 GMT
View Forum Message <> Reply to Message

It is nice that you are asking us, instead of just dropping the
changes.

On Dec 18, 7:51 pm, Chris Torrence <gorth...@gmail.com> wrote:
> The primary change is the use of the dot "." for object method calls.
> The use of the "." for method calls is now industry standard, for
> example in languages such as Java, Python, etc. For example, in IDL
> 8.0, the following code will create a plot, and retrieve the first
> child object:

I agree with the change, but the big problem I see with it is that
code
written in the new syntax would not work on older versions, which
would be a problem when we write code to be shared with other users.
In the academic
environment (the only I am familiar with), I still see a lot of people
using
IDL 6 (and many only using command line, editing source files with
vi).
Many times it is not their choice, because they are stuck with what is
installed (or the subset that works) in the department computer they
use.
Often I have found that such system administrators excel at making
users
suffer, from bad choices, and from not making things work properly. In
this
particular case, I am talking about administrators not updating IDL,
or not
bothering to make idlde work.

So since that change (and possibly others) would make the new syntax
backwards-incompatible, I would favour the use of a new extension to
identify
files with the new syntax, regardless of using or not option #5. And
if all
syntax changes are that simple, it would be nice to have a translator,
so that
we could write in the new syntax, but still have usable code for older
versions. Otherwise, I (and I guess others) would have to refrain from
using
the new syntax for a while.

The worst problem I see with the new extension is that it would allow
for the same routine to exist in a .pro and .prx file. The default

should be to pick the file with the new extension when there are both,
but it does create a new source of confusion. However, considering how
easily things can already get confusing due to files with the same
names existing in different places in the path, I do not find this
problem very significant.

>
> 1. How much code do you have that would break? Are you willing to
> retrofit your code?

I have a lot of old stuff written when I did not know any better.
However, any time I use that stuff for a new application, or to send
to
somebody else, it already takes a substantial overhaul anyway, to
replace all
the ugly stuff I used back then. So, the code I would have to retrofit
is code I already have to fix regardless of that change.

>
> 2. Do you use existing libraries (like Astrolib or JHUAPL's) that
> would break? Could ITTVIS retrofit these 2 libraries and give them
> back to the community?

Occasionally I use them. I would find it very nice if ITTVIS did the
necessary work on them.

>
> 3. Are there potential issues with changing to "defint32" (32-bit
> integers), or is the only problem with parentheses for arrays?

I do not see a problem with that, since the default is being promoted.
If any code needs a particular data type, it should be picking the
type explicitly anyway, not relying on defaults.

Generally, I welcome the changes, to get rid of awkwardness or
limitations
inherited from ancient times, as David indicated (I also agree with a
rainbow
ban). Because of that, in principle, I would prefer for myself option
#1.

However, considering the effect on other users, I find option #5 to be
better.
If option #1 is used, I can already see a bunch of users not upgrading
to IDL
8 because it will break the code they use. Even worse, I see
department system
administrators not replacing old IDL versions because it would upset

some old
professor, whose code written in 1972 would stop working, and who
cannot be
bothered to think about why.

This is related to the general situation I saw in many universities,
where new
generations of students keep suffering, stuck with 1970s programming
because
that is the way they get "taught" by the old professors who keep doing
things
the same way they did 30 years ago. Using IDL is already a big step
from the
still frequent practice of doing everything in F77, writing text
files, and
then plotting them on some other, awkward software. Also, a lot of
people keep
writing F77-style IDL.

So while I agree with David that making the default break old code has
the positive effect of inducing change, I think that the side effect
of inhibiting upgrades may be even worse. The license files being
version-specific and the difficult installation in Linux are already
two big obstacles, that keep people using old versions. With option
#5, the new extension will make it more visible to old users that
there is something new and better, without being so aggressive as to
just make their code stop working.

Though option #3 is more flexible, it has the bad side effect of
making the execution environments less uniform, since the default
would not be known when writing the code, while option #5 gives an
unchanging rule. More users would be confused when their code worked
on one computer, and not in another (creating a new way for the sky to
fall).

I find options #4 and #2 to be undesirable. In case of #4, it would
keep the tolerance to the old style, and keep the current additional
complication to enforce the new style. I always saw the idl2 option as
a form to ease the transition, which should eventually become default.
And even without the new syntax, it is already confusing to deal with
the errors that arise when idl2 is not used, so it is good to have it
as default. As for option #2, it is a compromise between #1 and #4, it
has the problems of both options, and creates additional complexity.

Regardless of which option gets chosen, it is important that the
changes are well advertised and explained on the website, without
having to download the new version to read about it on the help. Even
though I look forward for all the new features at each new version,

sometimes I only notice one of them when I read this newsgroup or somebody else's website (one recent example being the iTool changes which Michael Galloy wrote about). Also, it is good to explain the reasons for changes, as I often see people frustrated by what they perceive as arbitrary and needless changes in software.