Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by hegde on Wed, 05 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning (davidf@dfanning.com) wrote:

> do not understand how programs get compiled. Many of these
> people try to run IDL in the compile-link-run fashion of
> the good ol' Fortran days. :-)

Compile-link-run fashion is still the standard (best?) way.

> If you use the IDL executive command .Compile to compile
> a file, all the program modules in the file get compiled.

Major deficiency with IDL compiled state is that it doesn't even check
whether calling function(/procedure)'s argument list matches with the
compiled one. So if one makes a spelling error, the program will crash
at runtime. If the software is of considerable size, chances are that
one might miss that particular state during testing; whereas a simple check
would have avoided that.

> But one of the wonderful features of IDL is the ability
> to have your programs compiled *automatically* when they
> are needed. This makes it possible to write IDL code that
> is not cluttered up with a bunch of INCLUDE statements, etc.
> You simply put all of your program files in some directory
> that is on the IDL path and you don't have to worry whether
> the files are pre-compiled or not when you write the next
> IDL program that uses one of these commands.

Isn't it similar to 'library path' in C/C++?. INCLUDE is not intended for
including a huge source code anyway.

Say if one's display library contains 20 different files and if they were
modified after compilation, to recompile one has to type 20 .Runs or quit
the session and start all over again !

OR sometimes it is hard to predict which function will be called first. ex: string libraries for
parsing/manipulating strings. The only solution then will be
to keep 10-20 line codes in separate files generating an army of files.

> Automatic compilation doesn't solve *all* the problems
> you might run into. (Hence, the Forward_Function command.)
> But in my experience it solves a lot of them. And it sure
> beats having to compile all of your program modules before
> you can do anything in IDL. :-)

This feature might be good for running few things from IDL prompt. But as a programmer, I would like to better manage source code instead of pondering each time I call a function whether IDL would have compiled it before.

The arguments for 'separate files' appear to me as though telling that one could do something similar but not quite similar.

-M. Hegde

Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by wonko on Wed, 05 Nov 1997 08:00:00 GMT

View Forum Message <> Reply to Message

davidf@dfanning.com (David Fanning) wrote:

> If you use the IDL executive command .Compile to compile
> a file, all the program modules in the file get compiled.
> (The executive command .Run can also be used this way,
> although it confuses new IDL users because nothing actually
> gets *run*.)

Unless the file really contains a program (some stuff outside pro/
function declarations, ended by an END).

> But one of the wonderful features of IDL is the ability
> to have your programs compiled *automatically* when they
> are needed. This makes it possible to write IDL code that
> is not cluttered up with a bunch of INCLUDE statements, etc.

Right...

> But the key word here is "command". When you write a program
> you are, in effect, creating another IDL command. The key rule
> for getting your programs to compile automatically is to make
> sure that the "command" module is the *last* program module
> in a file that has the *same name* as the command you are
> trying to build (with a ".pro" extension, of course).

... but I wouldn't mind if IDL just compiled the whole file. This means some extra time for compiling, but would make things a bit easier. Or do I forget something?

Alex

--
Alex Schuster   Wonko@weird.cologne.de   PGP Key available

alex@pet.mpin-koeln.mpg.de

Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by davidf on Wed, 05 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

Martin Schultz (mgs@io.harvard.edu) writes:

> Hmmm... This sounds to me like you are fond of make
> files etc. I actually like IDL's way of automatic
> compilation, EXCEPT if one wants to distribute
> a program package and has to find out which files are needed. It would
> certainly be useful to have a tool which would look for all
> the routines that may be called from a "command" (i.e. pro or function
> identical to filename), and lists the files in which they are found. Of
> course, this does depend on your installation (order of searchable
> libraries). I guess it would come down to a real or pseudo compilation
> and could possibly be achieved by
> some tricky use of the journal output ??? Has anyone written something
> like this ?

It would be useful to have a tool like this. But having
just gone through this exercise for a client, I'll tell you
what I did. I wanted to make a run-time application. The
application has 10-12 modules, each of which has calls
to any number of other library routines.

I've been using automatic compilation all through the
development cycle, but I needed something now that would
get all the library routines gathered up in one place
so I could make a save file. I decided to construct a
"make" file.

It looked something like this:

```
  PRO COYOTE_MAKE
  Resolve_Routine, 'coyote1'
  Resolve_Routine, 'coyote2', /Is_Function
  Resolve_Routine, 'coyote3'
  ...
  ...
  Resolve_All
  Save, /Routines, File='coyote.sav'
  END
```

I just put all my program modules at the beginning to
make sure they are compiled (this part only took 4-5

iterations :-), the Resolve_All routine at the end
catches all the library routines I am using, and I
am set. Now, as I add modules to the application, I
just include the name in here and I can make a save
file whenever I need it by typing "coyote_make" at the
IDL command line. Pretty slick, I think.

> Another useful option of such a program would be
> to filter out dead or duplicate routines which linger
> around from old versions of a code.

Don't need it. Anytime I want to find bad code I just
explain it to a novice IDL user. I've found that
the prouder I am of my hot-shot code the more dumb
mistakes I find in it. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by Martin Schultz on Wed, 05 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> M. Hegde (hegde@PROBLEM_WITH_INEWS_DOMAIN_FILE) takes
> exception with several things I wrote, but finally
> ends up with this:
>
>> This feature might be good for running few things from IDL prompt. But as a
>> programmer, I would like to better manage source code instead of pondering
>> each time I call a function whether IDL would have compiled it before.
>
> I clearly mangled whatever point it was that I was trying
> to make, because I couldn't agree more with *this* statement. :-)
>
> Cheers,
>
> David

Hmmm... This sounds to me like you are fond of make files etc. I
actually
like IDL's way of automatic compilation, EXCEPT if one wants to
distribute
a program package and has to find out which files are needed. It would
certainly be useful to have a tool which would look for all
the routines that may be called from a "command" (i.e. pro or function
identical to filename), and lists the files in which they are found. Of
course, this does depend on your installation (order of searchable
libraries). I guess it would come down to a real or pseudo compilation
and could possibly be achieved by
some tricky use of the journal output ??? Has anyone written something
like this ?
Another useful option of such a program would be to filter out dead or
duplicate routines which linger around from old versions of a code.

Regards,
Martin

--- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
IDL-homepage: http://www-as.harvard.edu/people/staff/mgs/idl/
--- -------

Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by davidf on Wed, 05 Nov 1997 08:00:00 GMT

View Forum Message <> Reply to Message

M. Hegde (hegde@PROBLEM_WITH_INEWS_DOMAIN_FILE) takes
exception with several things I wrote, but finally
ends up with this:

> This feature might be good for running few things from IDL prompt. But as a
> programmer, I would like to better manage source code instead of pondering
> each time I call a function whether IDL would have compiled it before.

I clearly mangled whatever point it was that I was trying
to make, because I couldn't agree more with *this* statement. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Automatic Compiliation of IDL Programs, Was: Lost Functions
Posted by Stein Vidar Hagfors H on Thu, 06 Nov 1997 08:00:00 GMT

View Forum Message <> Reply to Message

M. Hegde wrote:
[..]
> Major deficiency with IDL compiled state is that it doesn't even check
> whether calling function(/procedure)'s argument list matches with the
> compiled one. So if one makes a spelling error, the program will crash
> at runtime. If the software is of considerable size, chances are that
> one might miss that particular state during testing; whereas a simple check
> would have avoided that.

On the other hand, with IDL being an interpreting language, it allows
dynamically created (or modified) programs (which are compiled only
when *called* during runtime, not when *referenced* during compilation
of another procedure).

In fact I do have some programs which writes procedures "on the fly",
compiling them as they are needed - obviously this could cause
problems if syntax checking was performed at compile-time (not
very hard to fix, though).

But it would certainly be nice to have a few extra tools to manage
large IDL software collections - e.g., checking for number of
parameters,
legality of keywords, etc..

> Say if one's display library contains 20 different files and if they were
> modified after compilation, to recompile one has to type 20 .Runs or quit
> the session and start all over again !

Actually, you can type e.g., ".run prog1 prog2 prog3 prog4 prog5"....

Usually, when working on a large set of routines in one "session",

I put such statements into a file, e.g., "c.pro", and then I simply
type "@c" when I need to recompile...

(And when working with the idl-shell mode, my fingers seem to
have a will of their own, typing ^C-^D-^C (saving and recompiling)
faster than I can think)

> This feature might be good for running few things from IDL prompt. But as a
> programmer, I would like to better manage source code instead of pondering
> each time I call a function whether IDL would have compiled it before.

I do agree with you, I just haven't experienced IDL's standard
"compile-when-needed" approach as a problem. I agree that you may
get a silly number of tiny procedures, but it's always possible
to put them into one file (let's say, "stringlib.pro") and then
have a procedure at the end of that called STRINGLIB. If the user
wishes to use the string library - have the statement "stringlib"
in the startup file.

And let a stand-alone program (IDL procedure or otherwise) do the
syntax/parameter checking.

Regards,

Stein Vidar