Subject: Re: Compile or not compile?
Posted by Michael Wallace on Thu, 03 Mar 2005 17:06:35 GMT
View Forum Message <> Reply to Message

Nuno Oliveira wrote:
> There's something that sometimes in while it bores me a little bit. How
> do I know if I need to compile a function/procedure or it will be able
> to compile "on fly".
>
> At the beginning I thought this was related to the paths idl "knew", but
> this happens for two functions/routines in the same directory.
>
> Can someone tell when a function can be runned without be compile?

When the directory containing the function is in your IDL_PATH
environmental variable and the name of the file is the same as the name
of the function within it. If you have a function "foo", it must be in
a file called "foo.pro" in order for IDL to find it automatically.

-Mike

Subject: Re: Compile or not compile?
Posted by JD Smith on Thu, 03 Mar 2005 17:25:05 GMT
View Forum Message <> Reply to Message

On Mon, 28 Feb 2005 16:38:44 +0000, Nuno Oliveira wrote:

> There's something that sometimes in while it bores me a little bit. How do
> I know if I need to compile a function/procedure or it will be able to
> compile "on fly".
>
> At the beginning I thought this was related to the paths idl "knew", but
> this happens for two functions/routines in the same directory.
>
> Can someone tell when a function can be runned without be compile?

If it's on the path, has a file name which corresponds to the routine
name, and is locatable by IDL, it does not need to be explicitly
compiled. Explicit compiling *is* required for:

1. Files which are not in the IDL path (to see what your path is
   currently, try print,transpose(strsplit(!PATH,':',/EXTRACT)) must
   be compiled explicitly. I think the current directory when you
   start IDL is on the PATH as well.
2. Files which have been added to IDL's path after it started. IDL
   scans its path at startup, and only files which exist at that time
   can be found automatically.

3. For functions or procedures which do not correspond to the
   filename, you must explicitly compile the files if you don't first
   invoke the function or procedure. This is not a good idea. You
   should really stick to names like function_name.pro and
   procedure_name.pro to avoid this situation, and put auxiliary
   helper routines in those files before the eponymous function or
   procedure.
4. Files which have been modified since IDL first compiled them must
   be re-compiled (only once: I've seen more than one person who
   compiles their routines twice "for good measure").
5. Files which have the same name as other files on the IDL path, but
   which show up later on the path (lower in the list reported in #1)
   must be explicitly compiled by full pathname. This is also not a
   good idea, if it can be avoided. See
   http://www.dfanning.com/tips/namefiles.html.

JD

Subject: Re: Compile or not compile?
Posted by Benjamin Hornberger on Thu, 03 Mar 2005 17:27:07 GMT
View Forum Message <> Reply to Message

Nuno Oliveira wrote:
> There's something that sometimes in while it bores me a little bit. How
> do I know if I need to compile a function/procedure or it will be able
> to compile "on fly".
>
> At the beginning I thought this was related to the paths idl "knew", but
> this happens for two functions/routines in the same directory.
>
> Can someone tell when a function can be runned without be compile?
>

Have a look at this:

http://www.dfanning.com/tips/namefiles.html

Also, search the IDL help index for "automatic compilation".

Note that this works for source (.pro) files as well as for compiled
routines in .sav files (say, if you call a function or procedure "taco"
from the command line or from within some other routine, it can be found
if its source is in a file taco.pro or if it is precompiled in a file
taco.sav).

Good luck,
Benjamin

Subject: Re: Compile or not compile?
Posted by marc schellens[1] on Fri, 04 Mar 2005 11:42:50 GMT
View Forum Message <> Reply to Message

2. Files which have been added to IDL's path after it started. IDL
scans its path at startup, and only files which exist at that time
can be found automatically.

Under linux at least this is not true. Here IDL always searches through
the path again. Maybe it is different for the library path (the initial
!PATH setting), I haven't checked it there.
And what happens if !PATH is changed?

marc

Subject: Re: Compile or not compile?
Posted by JD Smith on Mon, 21 Mar 2005 16:40:40 GMT
View Forum Message <> Reply to Message

On Fri, 04 Mar 2005 03:42:50 -0800, m_schellens@hotmail.com wrote:

> 2. Files which have been added to IDL's path after it started. IDL
> scans its path at startup, and only files which exist at that time
> can be found automatically.
>
> Under linux at least this is not true. Here IDL always searches through
> the path again. Maybe it is different for the library path (the initial
> !PATH setting), I haven't checked it there.
> And what happens if !PATH is changed?

I suspect you are referring IDL<6.0, which was when path caching was
added to IDL. I think it must depend on whether your IDL is caching
its path, which it is by default starting at IDL 6.0. For me (under
Linux), newly created subdirectories and routines are not discovered:

The PATH_CACHE procedure is used to control IDL's use of the path
cache. By default, as IDL searches directories included in the
!PATH system variable for .pro or .sav files to compile, it creates
an in-memory list of all .pro and .sav files contained in each
directory. When IDL later searches for a .pro or .sav file, before
attempting to open the file in a given directory, IDL checks the
path cache to determine whether the directory has already been
cached. If the directory is included in the cache, IDL uses the
cached information to determine whether the file will be found in
that directory, and will only attempt to open the file there if the
cache tells it that the file exists. By eliminating unnecessary
attempts to open files, the path cache speeds the path searching

process.

The path cache is enabled by default, and in almost all cases its operation is transparent to the IDL user, save for the boost in path searching speed it provides. Because the cache automatically adjusts to changes made to IDL's path, use of PATH_CACHE should not be necessary in typical IDL operation. It is provided to allow complete control over the details of how and when the caching operation is performed.

The other thing to note is that the PATH is not cached on startup, only after the first time it is read. So try something like:

IDL> .run file_in_my_path

create another_file_in_my_path.pro in the same location as file_in_my_path.pro

IDL> .run another_file_in_my_path.pro

This will fail if the paths are cached.

JD
