Subject: library or one file
Posted by R.Bauer on Sun, 04 May 2003 09:26:55 GMT
View Forum Message <> Reply to Message

Hi,

I know this was discussed in the past. But I got yesterday a new idea.
Sometimes if you like to give someone a routine it is easier to give him/her one file where all the routines are collected in one file instead of a whole library. But if you like to work with the subroutines this is bad because always the whole file must be compiled and you are not able to call them directly.

Yesterday I found a compromise.

for example

helpon has two routines

```
pro helpon_info
end
pro helpon
end
```

Both routines are defined in the same file.

To simmulate them as separate files I set two links

```
ln -s ./helpon.pro ../links/helpon.pro
ln -s ./helpon.pro ../links/helpon_info.pro
```

The internal routine names could be resolved by the routine

http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_html/dbase/download/get_internal_source_names.tar.gz
or as idl 5.x bninary
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_source/idl_html/dbase/download/get_internal_source_names.sav

```
IDL> print, get_internal_source_names(file_which('helpon.pro'))
helpon_info helpon
```

with the file_link routine the links could be set.

regards

Reimar

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 == ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: library or one file
Posted by tam on Mon, 05 May 2003 14:04:56 GMT
View Forum Message <> Reply to Message

Reimar Bauer wrote:
> Hi,
>
....

> Yesterday I found a compromise.
>
>
> for example
>
> helpon has two routines
>
> pro helpon_info
> end
> pro helpon
> end
>
>
> Both routines are defined in the same file.
>
>
> To simmulate them as separate files I set two links
>
> ln -s ./helpon.pro ../links/helpon.pro
> ln -s ./helpon.pro ../links/helpon_info.pro
>

>
...

One caveat, wouldn't this require that the user cannot call any
routines in the library that uses a routine that
comes later in the library? I.e.,

```
   pro f1,x
      if (...) then f2,x
   end
   pro f2,x
      if (...) then f1,x
   end
```

(linked as f1.pro and f2.pro)

would work fine if the user first calls f2.
If the user calls f1 first, then initally IDL
doesn't compile f2. When the user does call f2,
then IDL will try to compile the file again and
give an error that the user is attempting to recompile
an active module (since it recompiles f1 as well).

I find that this kind of co-dependency among routines
is pretty common though not usually quite as obvious
as what you have above.

 Tom McGlynn

Subject: Re: Library
Posted by Craig Markwardt on Sat, 08 Nov 2003 19:50:24 GMT
View Forum Message <> Reply to Message

Guillermo Fernandez <guillermo.fernandez@epfl.ch> writes:
>
> I've a bunch of procedures that I use all the time. In order to avoid spaguetti
> coding, I've decided to put them togheter in a library (well... library in C,
> module in Python, you get the idea I guess... the equivalent in IDL).
>
> I've been Googling in order to find how to create and use libraries and have
> been unable to find any documents (nor examples nor tutorials) that explain
> that subject.
>
> Could you please point me to a ressource, join an example or simply give me a
> starting point to get me out of this gap?

Command: Make me a library!
Response: Okay, you're a library.

Seriously, IDL libraries are more conceptual than technical.
Typically, one puts library routines into a single subdirectory in
one's IDL path. [but that's not required.]

Often, routines in the same library have names which begin with the
same prefix. [but that's not required. And JD is not fond of that
technique.]

Often, libraries are distributed as a single .zip or .tar.gz archive
file with several .pro files inside. [but that's not required.]

It is possible to package your library into a single IDL .sav file,
but I don't recommend that for several reasons. First, it's version
dependent. If you ever use a different version of IDL, you'll
probably have to re-make the .sav file. Second, you still need to
restore the .sav file, which is not straightforward to do
automatically.

In conclusion, just put your routines in their own subdirectory, call
them a library, and they will be one.

Yours,
Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Library
Posted by JD Smith on Wed, 12 Nov 2003 00:12:32 GMT

View Forum Message <> Reply to Message

On Sat, 08 Nov 2003 12:50:24 -0700, Craig Markwardt wrote:

> Guillermo Fernandez <guillermo.fernandez@epfl.ch> writes:
>>
>> I've a bunch of procedures that I use all the time. In order to avoid
>> spaguetti coding, I've decided to put them togheter in a library
>> (well... library in C, module in Python, you get the idea I guess...
>> the equivalent in IDL).

>>
>> I've been Googling in order to find how to create and use libraries and
>> have been unable to find any documents (nor examples nor tutorials)
>> that explain that subject.
>>
>> Could you please point me to a ressource, join an example or simply
>> give me a starting point to get me out of this gap?
>
> Often, routines in the same library have names which begin with the same
> prefix. [but that's not required. And JD is not fond of that technique.
>]
>

Au contraire: I am eminently in favor of this technique. I have griped in
the past about a particular choice of prefix which elevates the status of
the programmer, but keeping the name-space clean and uncluttered is
crucial.

> Often, libraries are distributed as a single .zip or .tar.gz archive
> file with several .pro files inside. [but that's not required.]
>
> It is possible to package your library into a single IDL .sav file, but
> I don't recommend that for several reasons. First, it's version
> dependent. If you ever use a different version of IDL, you'll probably
> have to re-make the .sav file. Second, you still need to restore the
> .sav file, which is not straightforward to do automatically.
>
> In conclusion, just put your routines in their own subdirectory, call
> them a library, and they will be one.

I'd add a step: scan your library with idlwave_catalog before tar'ing, so
users of IDLWAVE will have convenient, auto-loading access to routine
information for your files (it's available with IDLWAVE, or separately on
idlwave.org).

Alas, if what I think you're actually after is a way to segment the
namespace and only use those modules you actually need, you're out of
luck: IDL has no built-in concept of modules as partitioned name spaces.
You can prepend prefixes to your library code which will do a similar
thing, at the cost of reduced brevity and increased line noise.

JD