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
