Subject: Re: XSTRETCH and Library Lessons
Posted by mmiller3 on Wed, 26 Apr 2006 13:34:44 GMT
View Forum Message <> Reply to Message

>>>> > "JD" == JD Smith <jdsmith@as.arizona.edu> writes:
[much text snipped...]

> Another way of thinking about it... how would you recommend
> fixing the originally proposed XSTRETCH problem using !PATH
> alone? The solution has to allow me to run XSTRETCH, and
> the other random code which contain shadowing routines (by
> now quite obsolete and incompatible), all in the same
> session.

Hi JD,

I see your point. My recommendation was that the IDL_PATH needs to be set "right," but that really boils down to "that's the only way I know how to handle it with existing IDL."

The simplest way to handle fine tuning of paths, again with the current IDL (and I encourage my colleagues to avoid this at all costs!) is to explicitly .compile or .run files (not routines!) with the full path specified. This makes code very unportable though - and I've spent plenty of time frustrated about why my changes don't seem to have any effect, only to find someone has slipped an absolute path into some code somewhere.

If I had my choice, I'd go with versioned imports and flexible name spaces, like you suggested, so I could do something like "import AstroLib" to get the default versions of the whole collection. Then I'd use elements of AstroLib with calls something like x = AstroLib.calculate_x(). If I wanted Foo from the default version, I'd have it. If I wanted Bar from another version, I'd like to be able to add Bar to the AstroLib name space (or replace it, if it is already there) with something like "import AstroLib-other-version.Bar as AstroLib.Bar". Then calls to AstroLib.Bar would not have to be changed in any code, but I could get them from what ever version I want.

In the absence of a name space mechanism, maybe this could be implemented by installing multiple versions of libraries in a directory tree something like this:

```
IDL_lib_root -- AstroLib -+- default (link to latest/prefered version)
                          +- 1.0     (contains version 1.0 files)
                          +- 1.1     ...
                          ...
```

```
                    \- 27.0    (contains latest and greatest...)
```

If I adhere to the name-files-so-IDL-can-automatically-find-and-compile-routines rule, the initial import of AstroLib can be done with

```
pro install_library, lib, version=version, root=root
if n_elements(root) ne 1 then root='/local/IDL/lib/install/dir/'
if n_elements(version) eq 1 then begin
   !path = root + lib + '/' + version ':' + !path
endif else begin
   !path = root + lib + '/default:' + !path
endelse
end
```

Replacing routines with other versions, could be done with something like

```
pro replace_library_routine, lib, routine, version=version, _extra=extra_keywords
current_path = !path
if n_elements(version) ne 1 then version='default'
install_library, lib, version, _extra=extra_keywords
resolve_routine, routine
!path = current_path
end
```

From the command line, or start up script, or where ever, I can do

```
IDL> install_library, 'AstroLib'
```

Now all the AstroLib routines are available to me by name. Replacing Bar with version 1.0 and Foo with my local modification (installed in a similar tree somewhere else...), could be done with

```
IDL> replace_library_routine, 'AstroLib', 'Bar', version='1.0'
IDL> replace_library_routine, 'AstroLib', 'Foo', version='mine', $
        root='/home/me/lib/IDL/AstroLib'
```

Later on, when I realize that my local changes are really not so usefull, I can go back to the default version with

```
IDL> replace_library_routine, 'AstroLib', 'Foo'
```

I certainly haven't thought this out enough to see where it would or would not work! (I'm still on my first bit of coffee for the day :-)

Mike