Subject: XSTRETCH and Library Lessons
Posted by JD Smith on Fri, 21 Apr 2006 20:55:42 GMT
View Forum Message <> Reply to Message

XSTRETCH is pretty fun. I especially like the curve plot for non-linear (but why not draw it for linear as well?). I downloaded the new version, and immediately had problems: the histogram didn't show up for me any longer, as it did in older versions. Just a gray background. The min/max lines showed, and could be interacted with, but no histogram.

Surely, I thought to myself, the good Dr. Fanning would not distribute such a mal-configured tool. So I looked into a bit deeper. It turns out, among the bread and butter COYOTE routines like FSC_COLOR and TVIMAGE, and FSC_FILESELECT, I had 3-4 copies of each of these on my IDL_PATH, included directly in other libraries, like PAN, ICG, CM, etc. Presumably you have since updated those files, and a standard load path shadowing issue (aka name space collisions --- the wrong routine of the same name getting called) caused XSTRETCH to break in a most unilluminating way.

For all you library distributors out there... I think a good rule of thumb is, if you'd like to pluck a routine from a random library, and distribute it with your own (after getting permission of course), you should rename it by adding a unique additional prefix, so, e.g., ICG_TVIMAGE, instead of plain old TVIMAGE. This saves your users from future changes to the tool breaking your code, and saves the other library maintainer from fielding all kinds of spurious "your code doesn't work" complaints that arise from simple load path shadowing.

An even better solution, in my opinion, is not to include routines from other libraries at all, but just state in your install notes:

FOOLIB requires the COYOTE library, version X.Y or later, available at .....

This puts a higher burden on your users to install another library, and on yourself to make sure that future changes to that library don't break your tools (i.e. to migrate your tools along), but the end result is much cleaner, and bug fixes and feature additions then get communicated back to the original library maintainer, so that everyone benefits. The worst offenders are those that "snapshot" another entire library and bundle it directly in their own. This severely pollutes the name-space, and for little added benefit. Why not just provide a pointer to the additional library?

The final solution, if you feel your users are too lazy to sort any of this out, is just to compile a .sav file of your entire library, and

distribute that. These don't suffer name space collisions. As long as they are loaded first, their versions of, e.g., TVIMAGE, will trump any others, and since they have to explicitly or implicitly loaded, the true-blue TVIMAGE would continue to load and work as expected in sessions where your tool isn't being used.

JD

P.S. IDLWAVE can help you identify offenders. Scan a your full library into an IDLWAVE library catalog, and then select IDLWAVE->Routine Info->[Load Path Shadow] Check Everything. You'll get a report on multiply defined routines, sorted in the order they will most likely be loaded. RSI even re-defines the same routine a few times in it's !DIR/lib routines!

---