Subject: Re: XSTRETCH and Library Lessons
Posted by mmiller3 on Mon, 24 Apr 2006 19:36:29 GMT
View Forum Message <> Reply to Message

>>>> > "David" == David Fanning <davidf@dfanning.com> writes:

> This is really starting to be a pain.

For our local libraries, I've had to define release tags for them. Then, every "application," which means "every thing that we expect to work the same way each time, gets started from a script that includes setting the IDL_PATH to include the proper release. (This can be a shell script or an IDL script.) To handle releases, I use CVS[1] and release tags. This problem is not intrinsically different than release/version-ing problems with any other language. In any case it requires some discipline. Compared to getting some of my colleagues to properly name files[2], this is a piece of cake!

Here's how I handle it... When ever I make a release, I commit my changes to the CVS repository and create a new release tag, usually using a sequential, increasing number, say 2-72. Then I check out that release and put it a directory named using that tag, say mylib-2.72. If I want that to be the default release for anyone using release 2.x, I can link mylib-2.72 to mylib-2. If I want that to be the default release for anything that doesn't care about versions (pretty rare!), I can link mylib-2.72 to mylib.

This is purely patterned after what I see in my /usr/lib directories, and can, I think, be done under any OS. This can be done with Subversion[3] as well as CVS, or any other version control system. (Note that version is a different thing from release. Version means version of a file - release means snap shot of a collection of files).

Here's how I would do this for a collection of codes called myIDLlib

0 - make sure my version control system is set up (in this case CVS)

1 - get a copy of myIDLlib and put it into a directory that I've cleverly named myIDLlib

2 - change directory to myIDLlib. If there is cruft in this directory that I don't want in the repository, delete it now.

3 - add everything to the repository:

```
> cvs import -m "Imported sources" myIDLlib yoyo start
```

4 - Make the initial release tag. Tags are not allowed to have names with periods in them, so I use dashes instead:

```
> cvs tag rel-1-0
```

5 - Move this directory to where ever I store production code and name it myIDLlib-1.0

```
> cd ..
> mv myIDLlib myIDLlib-1.0
```

6 - set up application scripts to have myIDLlib-1.0 in the IDL_PATH

7 - Now when I want to change my code, I check out a fresh copy:

```
> cvs checkout myIDLlib
```

8 - I corrupt it/change it/improve it...

9 - I check in my changes frequently so I can undo them as needed...

10 - repeat 8 and 9 until I decide I'd better hang onto a snap shot of this before I get myself into deeper trouble.

11 - Make a new release tag:

```
> cvs tag rel-1-1
```

12 - Move this directory to myIDLlib-1.1

```
> cd ..
> mv myIDLlib myIDLlib-1.1
```

13 - change the start up scripts for applications that need the latest and greatest. Applications that need older releases already have the correct release in their IDL_PATH, so nothing needs to be changed.

Regards, Mike

[1] http://www.nongnu.org/cvs/
[2] http://www.dfanning.com/tips/namefiles.html
[3] http://subversion.tigris.org/

--
Michael A. Miller   mmiller3@iupui.edu
Imaging Sciences, Department of Radiology, IU School of Medicine