Guidelines for maintaining JED-related Debian packages

Author: Rafael Laboissiere and the Debian JED Group
Contact: pkg-jed-devel@lists.alioth.debian.org
Date: Sun Apr 26 16:24:28 2009 +0200

1. Coordination

The development of JED-related packages in Debian is done as a collaborative work by the Debian JED Group (DJG) at alioth.debian.org. Any maintainer/developer, either a DD or not, interested in JED-related packages is encouraged to join the pkg-jed project at Alioth.

Issues are discussed in the pkg-jed-devel mailing list. For more information about subscription and for looking at the archives, please visit http://lists.alioth.debian.org/mailman/listinfo/pkg-jed-devel

2. Building and uploading packages

The first thing to do when contributing a new package is to set the maintainer to Debian JED Group <pkg-jed-devel@lists.alioth.debian.org> in debian/control.

The Uploaders field of debian/control should list all uploaders, also those who are not Debian developers. This way, uploads (even sponsored ones) will not be considered as non-maintainer uploads (NMUs).

The debian/changelog entries should look like this:

```
jed-cool (1.2.3-4) unstable; urgency=low

  * Rebuilt with cool stuff [JD]
  * New user interface [SD, JD, SU]
  * Added "-v" option (closes #3456)

 -- Sam Uploader <sam.u@lists.example.org>  Sat, 19 Mar 2005 09:29:18 +0100
```

Where JD, SD and SU are the initials of the developers associated with each change and should appear at the end of each entry. Changes by the uploader do not need a tag. (Cf. the last change in the above example)

In sequential or grouped changelog entries, you can void repetition of the tags like this:

```
* Updated debconf templates translations: [JD]
  + German: thanks to Helge Kreutzmann (closes: #424115)
  + Dutch: thanks to Bart Cornelis (closes: #424664)
  + Marmish: thanks to Mar Mish [SU]
```

URLs of the SVN repository can be added to the Source section of the debian/control file, which will appear in the Package Tracking System (http://packages.qa.debian.org). They should read like this:

```
Vcs-Svn: svn://svn.debian.org/svn/pkg-jed/<source-package>/
Vcs-Browser: http://svn.debian.org/wsvn/pkg-jed/<source-package>/
```

Once a developer thinks that a given package in the SVN repository is ready for upload, she should drop a message in the pkg-jed-devel mailing list containing a request for upload [RFU]. If there are no objections within two business days, then the upload can be done by one of the official Debian developers of the DJG.

3. SVN repository

The packages are under Subversion control at svn.debian.org. For more information on Subversion, please look at the web site http://subversion.tigris.org. The Subversion book is also a valuable resource http://svnbook.red-bean.com.

SVN activity is automatically sent to the pkg-jed-commit mailing list. For more information, please visit http://lists.alioth.debian.org/mailman/listinfo/pkg-jed-commit

The repository is organized as follows:

```
svn://svn.debian.org/svn/pkg-jed/
  jed/
    branches/
      [...]
    tags/
      0.99.16-4/
      0.99.16-5/
      [...]
    trunk/
      debian/
  jed-extra/
    tags/
      0.1.8/
      1.0-1/
      [...]
    trunk/
      debian/
```

Notice that only the debian files are put under Subversion control. No upstream files, please.

As regards tag conventions, use the following path:

```
svn://svn.debian.org/svn/pkg-jed/<package>/tags//<version>
```

Only create a tag in <package>/tags after the Debian version is actually released. This avoids having to issue extra svn copy commands if the tag is created too early.

Comparison between two released Debian versions of a given package can then be done with a command like the following:

```
svn diff \
svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/tags/1.2.3-3 \
svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/tags/1.2.3-4
```

To put a new package under Subversion control, use the svn-import-pkg script or follow this cookbook:

- Create the directory for the new package, as well as the subdirectories trunk and tags:

```
    svn mkdir svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool        \
      svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/trunk          \
      svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/trunk/debian   \
      svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/tags           \
      --message 'Created directory structure for jed-cool package'
```

note: This only works if your user name in the system you are working is the same as your user name in Alioth. If this is not the case, you will need to set the SVN_SSH environment variable, like this:

```
export SVN_SSH="ssh -l my-alioth-username"
```

Otherwise, you can add your Alioth user name directly in the URL:

```
svn co svn+ssh://my-alioth-username@svn.debian.org/svn/pkg-jed
```

Yet another solution, probably the best one, is adding to your personal configuration file ~/.ssh/config something like:

```
Host svn.debian.org
User my-alioth-username
```

- Get the latest released version of the package:

```
apt-get source jed-cool
```

- Import the Debian-related files into the repository:

```
svn import jed-cool-1.2.3/debian                          \
  svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/trunk/debian \
  --message 'Initial import of package jed-cool'
```

- Tag its current version:

```
svn copy svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/trunk \
  svn+ssh://svn.debian.org/svn/pkg-jed/jed-cool/tags/1.2.3-4 \
  --message 'Debian release 1.2.3-4'          \
```

Developers who are not members of the Debian JED Group can access the SVN repository anonymously.

- The whole tree:

```
svn co svn://anonymous@svn.debian.org/svn/pkg-jed
```

- The sources of a specific package:

```
svn co svn://anonymous@svn.debian.org/svn/pkg-jed/<pkg>/trunk
```

When committing changes that do not result in a Debian release, keep an indication in debian/changelog that the package is not yet released, like this:

```
jed-cool (1.2.3-4) UNRELEASED; urgency=low

  * First import

 -- Sam Uploader <sam.u@lists.example.org>  Sat, 19 Mar 2005 09:29:18 +0100
```

4. GIT repository

In the GIT repositories we track in the branch "upstream" the upstream source and in the branch "master" we track the package development. For this we merge in the upstream branch on a regular basis.:

```
  +---o---D---o---o---o---E---+---o---o---F         master
 /                            /
U---o---o-----------o---o---V                       upstream
```

U and V are upstream releases, D, E and F are Debian releases and the + are merges of the upstream branch in the master branch.

This way you can checkout the master branch and build the current version or start hacking the source.

Patches to the upstream source must still be dpatches (or anything similar) to make the Debian source package as much as useful. With dpatches it's easy to see which changes belong together and it's easy to disable some changes. The command "git diff --stat upstream..master" must only list files in the subdirectory "debian."

4.1 One Debian release cycle

Make sure you have the most up to date files with:

```
git fetch               (for Debian sources)
git svn fetch           (for upstream sources, if upstream uses SVN)
```

You should checkout your master branch [1] and import the last changes [2] from the central GIT repository by merging. This should result in a fast forward merge. Otherwise something is broken. You should ask for help on the mailinglist or throw your changes away [3]:

```
git checkout master                 [1]
git merge origin/master             [2]
git reset --hard origin/master      [3]
```

Now, you have two options: Start a new Debian version, e.g. 1.2-2, or import the recent upstream changes and start a new version, e.g. 1.3-1.

4.1.1 The beginning

When you start a new version, you have to merge the current upstream source into the master branch.:

```
git merge upstream
```

Now you are here::

```
  +---o---D---+         master
 /           /
U---o---o---V           upstream
```

The first commit in the new cycle must be a change of the file debian/changelog that starts a new Debian version, regardless if it a new Debian version or a new upstream version. Edit the file [1], add it to the GIT index [2] and make the commit [3].:

```
xjed debian/changelog                         [1]
git add debian/changelog                      [2]
git commit -m "Start a new version"           [3]
```

The commit should look similar to this:

```
% git show
commit ca731f5f39b5b8982a8b38d04b569c1fedf9c45b
Author: Joerg Sommer <joerg@alea.gnuu.de>
Date:   Sat Apr 26 10:56:20 2008 +0200

    Start a new version
```

```
diff --git a/debian/changelog b/debian/changelog
index f1ccf3b..1f95cae 100644
--- a/debian/changelog
+++ b/debian/changelog
@@ -1,3 +1,10 @@
+slrn (0.9.9~pre102-1) UNRELEASED; urgency=low
+
+  * This packages is based on the svn snapshot from
+    https://slrn.svn.sourceforge.net/svnroot/slrn/trunk
+
+ -- Joerg Sommer <joerg@alea.gnuu.de>  Thu, 01 Jan 1970 00:00:00 +0000
+
 slrn (0.9.9~pre99-1) unstable; urgency=high

   * This packages is based on the svn snapshot from
```

Now, you should start a new branch for the changes you want to do. Name it work-$YOUR_SIGN.:

```
git checkout -b work-$ME
```

Then you should try to build the package and if it fails, you should fix this, e.g. update the dpatches.

4.1.2 Making changes

After you brought the package in shape, you should start making other changes. But keep the following commit policy in mind:

- The package must build after every commit---except of the first few commits of a new cycle needed to bring the package in shape.
- Every commit should change only one thing. Don't so summary commits. GIT is a distributed VCS and you can commit every time. It's not SVN!
- Do all changes needed for a change, e.g. update the debian/changelog, if necessary, update debian/patches/00list and so on. Ask yourself the question: "If my change gets reverted, is it enough to revert this patch?"
- Describe in the commit message, what you want to do and why you want to do it. You have much more space in the commit message than in the Debian changelog. You can put statistics there or explain design decisions.
- Don't update the timestamp in the changelog file. Changing the timestamp with every commit clutters the diff and might cause problem on merging and cherry-picking.

To make a commit you must add the changed files to the GIT index. You can use add whole files [1] or only parts [2]. After this you can view the changes [3] and commit them [4]. You can also combine the steps [1] and [4] to [5] or [2] an [4] to [6].:

```
git add debian/rules                 [1]
git add -i                           [2]
git diff --cached                    [3]
git commit                           [4]
git commit debian/rules              [5] = [1] + [4]
git commit --interactive             [6] = [2] + [4]
```

4.1.3 Fixing commits

The history shouldn't look like this:

- dpatch to support something
- sorry, I forgot the 00list file, here is it
- make funny stuff
- changelog entry for something above
- improve spelling of funny stuff

GIT gives you the tools to fix the history of your branch "work-$YOUR_SIGN." Don't do this in the branch "master!"

You can change the last commit with the option --amend [1] or you can use rebase [2] to make changes like joining, reordering, removing and editing of commits.:

```
git commit --amend                [1]
git rebase -i HEAD~10             [2]
```

4.1.4 Publishing your changes

You don't have to publish every single commit. You can publish many commits at a time and there can be days between the commit and the publishing. You have enough time to think about what you have done and maybe change it!

After you have done your changes, you should publish them by running:

```
git push
```

If you used rebase you must use the option --force.

4.1.5 Closing the development cycle

After all is in shape and the changes are reviewed by the other team members, someone should step up and make the release. He should merge [2] missing commits in the master branch [1] and do the final commit by updating the Debian changelog [2] and add a tag [3]. After this is:

```
git checkout master               [1]
git merge work-$X                 [2]
git commit debian/changelog       [3]
git tag 1.3-1                     [4]
```

The final commit should look like this one::

```
commit 1c95c225679ff863d29d21f50b6b1b0fe38a544f
Author: Joerg Sommer <joerg@alea.gnuu.de>
Date:   Sat Apr 26 16:22:34 2008 +0200

    Release of 0.9.9~pre102-1

diff --git a/debian/changelog b/debian/changelog
index 591c5c4..d4e5168 100644
--- a/debian/changelog
+++ b/debian/changelog
```

```
@@ -1,4 +1,4 @@
-slrn (0.9.9~pre102-1) UNRELEASED; urgency=low
+slrn (0.9.9~pre102-1) unstable; urgency=low

   * This packages is based on the svn snapshot from
     https://slrn.svn.sourceforge.net/svnroot/slrn/trunk
@@ -40,7 +40,9 @@ slrn (0.9.9~pre102-1) UNRELEASED; urgency=low
   * Dropped the dpatch 206_branding, because I don't see what's the value
     of this patch except for fun.

- -- Joerg Sommer <joerg@alea.gnuu.de>  Thu, 01 Jan 1970 00:00:00 +0000
+  * Upload sponsored by Norbert Tretkowski
+
+ -- Joerg Sommer <joerg@alea.gnuu.de>  Sat, 26 Apr 2008 16:19:22 +0200

 slrn (0.9.9~pre99-1) unstable; urgency=high
```

(view source) (PDF version)