From Marmalade to Emacs – EmacsConf 2013

Michael Olson

Mar. 29, 2013

Target Audience

- Authors of add-ons for Emacs that may have already gotten them into ELPA, el-get, or Marmalade, that might be interested in getting them into Emacs itself
- Power-users with project management skills who may want to work with an add-on maintainer to help get that project into Emacs
- Authors of relatively new add-ons who want to keep their options open for getting the add-on into Emacs
- Not about: packaging for ELPA, el-get, or Marmalade

Scope of Talk

- Why get code into Emacs?
- Licensing concerns w.r.t. Debian
- Assessing amount of work to get existing code into Emacs
- Bookkeeping practices for new contributions
- Whether to keep a non-Emacs fork
- Best practices for maintaining a fork

Definitions

Emacs-ready

Copyright is fully assigned to FSF in terms of the name of this project (not Emacs); could go into Emacs with a small amount of technical effort and Emacs maintainer consent

About Me

- Maintained ERC; Emacs IRC Client, got it into Emacs—model project for this talk
- Maintained Emacs Muse; similar to Org, backed publishing of Planner Mode; “Emacs-ready”
- Maintained EMMS; Emacs music player, wrote MPD backend; “Emacs-ready” as of 2009

Why Get Code into Emacs: Pros

- Increases userbase of your add-on
- Generates excitement for existing Emacs users, which helps grow the community
- Allows code to keep up faster with the latest Emacs features, because deprecated functions and variables will show up when Emacs developers grep the codebase

Why Get Code into Emacs: Pros

- Great resume addition: "I have code in Emacs"; Worked for me anyway... YMMV
- Time capsule; Emacs has been around 37 years, could see active use for that many again

Why Get Code into Emacs: Cons

- Contributors with > 12 lines must assign copyright `http://www.gnu.org/prep/maintain/html_node/Copyright-Papers.html`
- Have to keep track of authorship and reach out to contributors as a result of the above
- Need to get copyright assignment from your place of work; this is usually more time-consuming than genuinely difficult
- Initial efforts can be challenging

Why Get Code into Emacs: Cons

- Can sometimes come down to clean-room reimplementation of feature when contributors are not willing/able to assign code
- Process can take half a year for code with many contributors
- Users of the add-on via Emacs may take a while to see new features, depending on whether you're able to get changes into minor Emacs releases; distinguishing version strings for bug reports is useful here

Social concerns

- Your code will need to be GPL v3
- Your documentation is recommended to be GFDL, but maintainers can (last time I checked) accept a looser license like GPL or LGPL
- Debian project will not accept GFDL'd documentation because of invariant (non-modifiable) sections; even documents without invariant sections are sketchy to them
- Debian makes you install an optional package with name "-non-dsfg" to get Emacs manuals; Ubuntu will instead install that package for you

Getting Code Emacs-Ready

- Track down all the people who have contributed code to the project
- If the project has an AUTHORS file or similar, that would be a good file to keep updated with findings and compare notes
- You'll need to look at logs, grep for 'thanks' and other attributions in commit messages. Senging `git log -p` output to a file and looking through it multiple times may be useful.
- Once you have a list of name, email address, and number of nontrivial modified lines (discounting indentation and similar), you'll need to contact anyone who has done more than 12 lines of code

Getting Code Emacs-Ready

- Use this as a guide for how to send copyright requests: `http://www.gnu.org/prep/maintain/html_node/Copyright-Papers.html`
- Contact the FSF Copyright Clerk at `assign@gnu.org` for help/questions, especially around whether to assign to Emacs or the name of your program; preferences can vary, though last I checked, name of program is preferred
- May want to pre-emptively ask to assign contributions for both program and Emacs, to save time
- Decide how long you want to wait for folks to respond and finish the process. Expect it to take up to 5 months.
- If you can't get ahold of a contributor, you may need to consider rewriting their contributions or omitting the file with their changes, though this is an unpleasant last resort.

Bookkeeping practices

- People will send assignments to the FSF, who will contact you as they come in; you may or may not be provided with a GNU shell account with read-only access to the master copyright assignment file
- Once they're all in, change copyright notices on all your source files to be:
  Copyright (C) 20XX - 20YY Free Software Foundation, Inc.
- Copyright notices should note that this add-on is not part of GNU Emacs
- If copyright notices are missing from files that would be considered source, add them
- Your code is now "Emacs-ready"
- As new contributions come in, check them against the AUTHORS file and keep that up-to-date to save yourself time later

Getting code into Emacs

- It may be a good idea to have documentation prepared in Texinfo format before proceding further; this allows your project to be easily discovered while browsing Emacs manuals, and might be a requirement for larger add-ons
- Depending on your comfort level, contact either `emacs-devel@gnu.org` or Stefan Monnier and Chong Yidong (current maintainers of Emacs) to get an idea for level of interest
- You and your active contributors might need new assignments for Emacs itself, if the previous assignments were just for the program

Getting code into Emacs

- If the maintainers agree to accept your code into Emacs, they'll tell you which branch, and get you added to the list of Emacs contributors on Savannah so you have write access
- You may need to change bzr configuration to use this new writable access method
- If your add-on is large, you may need a new directory added to the source tree, with a Makefile; otherwise an existing directory is fine; maintainers will give direction in this
- Documenation and miscellenea go into other directories

Getting code into Emacs

- Copyright assignments will need to change to note that the files are part of GNU Emacs
- Someone will prepare a short summary of your add-on for the new features list in the next Emacs release
- It might take until the next major Emacs release for the changes to go live
- When updating your add-on in Emacs, may need to update ChangeLog file, though a discussion appears to be ongoing about that

Keeping a fork: worth it?

- If you already have an active community around your add-on, you may want to maintain a fork so that you can do quick releases for your community members
- Otherwise, may be best to consider Emacs to be the canonical place to get your add-on
- If XEmacs or support for older versions is a concern, may wish to keep a fork for them, though this is less of a concern now that Emacs maintainers are so quick with new releases

Keeping a fork: maintenance

- If you do decide to maintain a fork, best practice is to keep a dedicated branch of your project; I like "emacs-24"
- This branch should have the exact content of your source files; if you want to track documentation, may want to create subdirs like doc, src, and have a script that can copy files to/from Emacs and your branch; but keep directory structure exactly the same as your master branch
- Process is something like:
  - Switch to emacs-24 branch
  - Run import script that copies changes from Emacs to your workdir
  - Spot-check results to catch new files, deleted files, and changes needing compatibility hacks

Keeping a fork: maintenance

- Process, continued:
  - Commit these changes
  - Merge them back to master, summarizing changes in commit log
  - Merge from master branch
  - Resolve conflicts, commit
  - Copy files to Emacs source tree using export script
  - Run tests on Emacs source (compile and sample use cases at minimum)
  - Check in changes to Emacs
  - Do an "ours-style" merge from your emacs-24 branch to your master branch to indicate that you have up to these changes in Emacs

Compatibility functions

- Upon committing code to Emacs, you'll need to remove any compatibility hacks to make code work with older versions of Emacs or XEmacs
- When introducing new ones to your fork, make sure you use your project's namespace, not the global namespace
  - Ex: erc-format-string instead of format-string
- Good compatibility hacks should check for latest function name first using fboundp, then fall back to earlier names
- For speed, do eval-when-compile and make (for example) erc-format-string an alias for format-string

Questions?