

The why and how of getting packaged

Michael Hanke

Debian Developer
Otto-von-Guericke University, Magdeburg

5th BrainScaleS CodeJam

Mar 15th 2012

About me

- PhD in Neuroscience/imaging [<http://mih.voxindeserto.de>]
- Python Software-Developer (e.g. PyMVPA) [http://www.py_mvpa.org]
- Debian Developer [<http://www.debian.org>]
- One of the NeuroDebian founders [<http://neuro.debian.net>]
- Seven years of Debian packaging experience (neuroimaging, electrophysiology, distributed computing, psychophysics, network security)
[<http://qa.debian.org/developer.php?login=mih@debian.org>]



Why bother?

... most Unix-based physics software produced by research organizations fails to meet even the simplest expectations one might have for quality software. Let me clarify that I am not referring to the actual code, which is generally quite good, and is a testament to the skills and intelligence of the authors. No, I am talking about how the process of compiling and installing a well-reputed piece of physics software is fraught with confusion, hassle and worse. There is absolutely no excuse why it should be this way.

– Kevin B. McCarty, Post-doc, software/Debian developer, sysadmin

... most Unix-based ~~physics~~**neuroscience** software produced by research organizations fails to meet even the simplest expectations one might have for quality software. Let me clarify that I am not referring to the actual code, which is generally quite good, and is a testament to the skills and intelligence of the authors. No, I am talking about how the process of compiling and installing a well-reputed piece of physics software is fraught with confusion, hassle and worse. There is absolutely no excuse why it should be this way.

– Michael Hanke, Post-doc, software/Debian developer, sysadmin

What means “to be packaged”?

- Integration into a software distribution eco-system
- Standardization of build and installation procedures
- Uniform specification of meta-information (e.g. dependencies)
- Distribution through a common repository

Why: Ease installation and upgrade procedures

Not packaged

Installation

- Find instructions
- Compatible binaries?
- “If on XXX, add few symlinks” – to trick the linker
- Find dependencies, obey versions!
- Figure out version conflicts yourself
- You can always change operating systems!

Upgrade

Do it again... Well, maybe later...

Packaged

Installation

```
sudo apt-get install ...
```

Upgrade

```
sudo apt-get update  
sudo apt-get upgrade
```

Why: Improve user experience through system integration

- Software and all its dependencies come together and work together
- Predictable and reliable locations of tools, libraries, and data
- Opportunity for elegant and simple works-out-of-the-box default configurations

Why: Improved visibility and perceived maturity

Quick check

- How many of you regularly search for new software in your field?
- How many of you routinely scan scientific journals for new software?

Why: Improved visibility and perceived maturity

Quick check

- How many of you regularly search for new software in your field?
- How many of you routinely scan scientific journals for new software?
- Most users (want to) get software from system repositories (only)
- Apple got it (after 10+ years), and Microsoft too (after 20+ years)
- Being in the system repository shows that:
 - someone cares
 - it gets extra QA
 - it must be worth the effort
- If you're not in the system repository, you
 - need to get in touch with users that don't know you exist
 - convince them to go the extra mile to install your software by hand

Why: Encourage and facilitate external contributions

- Easier access to any software source code through a common repository
- Uniform build-procedures for all packages
- Efficiently deploy, track, and test distribution-wide changes/improvements
- Every package is a first class citizen
- Internationalisation/translation teams
- Porting to other architectures
- Detect problems due to compiler/library transitions
- Mutual awareness, e.g. documented 3rd-party dependencies

How to get there?

Attract a packager

Two types of people that know how to package

- 1 Those that aren't interested
- 2 Those that don't have time

Be an easy target!

Don't be different for no reason

- Be as much like your peers as possible
- Follow standards
(File System Hierarchy, FreeDesktop, ...)
- Stick to the respective default build-system
- ...

Don't scare away potential users/contributors with avoidable complications

More in Valentin's talk later on

Be transparent

- Use a publicly accessible version control system (VCS)
- Use a publicly accessible and archived mailing list (e.g. to announce your releases)
- Use a publicly accessible bug tracker
- Maintain a changelog
- Maintain a project website

Look active, healthy, and sane

Do verifiable versioned releases

- Have predictable and stable filenames and download URLs
- Allow for automatic downloads
- Never change the content of a file without a corresponding version change

Facilitate reliable automated processing

Have a **standard** license

- Be consciencious about your license (e.g. conflicts)
- Restrictions hinder audiences and contributors
- Do not ammend licenses with non-licencing issues (e.g. not FDA approved)

Use a standard license!

Allow build/install configuration modification from outside

- Allow for adjusting compiler flags and environment variables (e.g. -Werror issue)
- Allow for proper (out of source tree) installations
- Honor standard variables like “prefix” and “DESTDIR”
- Support running tests and build docs without prior installation

Support alternative configurations without having to patch the source

Allow for system library dependencies

- Do not embed 3rd-party code unless you have to
- Never modify this code
- Allow to satisfy all dependencies with system packages. Do not require your embedded copies to be built!
- Do not enforce static linking – not even against system libraries
- Build shared libraries whenever possible

Use system libraries whenever possible!

- Have unit tests
- Have regression tests
- Have some tests that don't need huge data blobs from a separate download
- Make sure your tests actually pass for your releases
- Tests are indispensable for reproducible research
- Tests are even more important for non-compiled code

Always test, and make it easy for others to test

Debian Upstream Guide

Short summary of key-points to obey to facilitate Debian packaging

<http://wiki.debian.org/UpstreamGuide>

This is why you FAIL

Talk by Tom Callaway (Red Hat) on common mistakes of open-source software release/management practices

<http://www.socallinuxexpo.org/scale9x/presentations/why-you-fail>

Do it yourself!

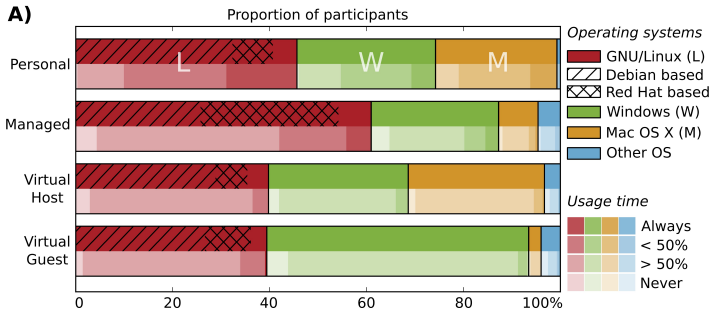
(Take packaging as a test of ease of deployment for users)

Mentoring/Support for Debian packaging of neuroscience software

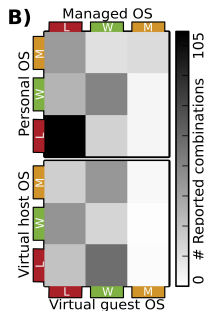
- Start with a tutorial, e.g.
`http://www.lucas-nussbaum.net/blog/?p=640`
- Look at packages of similar software
- Mailinglist: `neurodebian-upstream@lists.alioth.debian.org`
- IRC: #neurodebian on OFTC

Into what distros do I have to get for maximum effect?

A)



B)



Hanke & Halchenko, 2011, Frontiers in Neuroinformatics

- Effort inside Debian (started six years ago by Yaroslav Halchenko and me, both Debian developers)
- Mentors research software projects to get their work into Debian
- Focus on all of neuroscience (but on-demand)
- Provides backports for Debian and Ubuntu release
- Readily usable virtual machine image
- ≈ 100 packages (so far)
- 8 public international mirrors
- Currently about 15 new users per day
- Not funded, no pre-defined end-of-life

Visit <http://neuro.debian.net>

Michael Hanke
mih@debian.org
<http://mih.voxindeserto.de>

about the slides:

available at

copyright © 2012

<http://neuro.debian.net/#publications>

Michael Hanke, slide style inspired by Stefano Zacchiroli

CC BY-SA 3.0 — Creative Commons Attribution-ShareAlike 3.0