Integrating Perl in a wider distribution: The Debian pkg-perl group

Gunnar Wolf — gwolf@debian.org
http://www.gwolf.org/soft/debian_sysadmin

Instituto de Investigaciones Económicas, UNAM
Desarrollador del proyecto Debian

YAPC::Europe
August 28-30, 2007

Contents

Who am I?

- Perl programmer for almost ten years (although by far not yet an expert)
- I presented talks both at YAPC::NA 2001 and YAPC::EU 2002
- I currently maintain two (simple, small, low-maintenance) CPAN packages
- Involved in Debian since 2001; Debian Developer since 2003.
- Member of the Debian *pkg-perl* group since its formation, recently joined *pkg-ruby-extras*

Why am I presenting this? And why here?

- Programmers want their work to be useful — Or at least, used
- Free Software is one of Perl's most successful *ecological niches*
- The best way for my modules to reach the final user will probably be via a *Free Software distribution*
- Given the way the Perl community works... I expect this talk to be like *preaching to the choir*
  - Perl has a mature, Unix-culture-compatible culture
  - Many of the tools pkg-perl uses actually derive from prominent Perl mongers
- I expect to share this with you to better work together, to input this experience to other packaging groups in Debian, and probably to Perl packaging teams in other distributions

Are we really that similar?

There might be some sociological reasons for this — That will be up to somebody else to research into. I have at least noticed that both communities:

- Value documentation, and require all packages/modules to be at least basically documented
- Value portability and correctness, even for *fringe architectures*
- Principle of the least surprise; comparatively slow pace of development (ensuring not to break the existing user base). Things *do* change, but changes are gradual and well-thought. Both communities have been (wrongly?) blamed for stagnating.
- Thousands of packages/modules available in the repositories. Many of them are already in a *mature* stage, with very low change volumes, although very seldom abandoned.

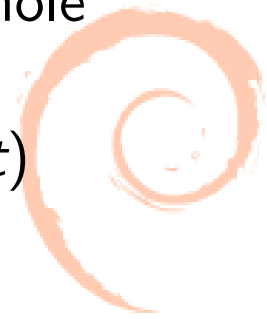
Contents

CPAN background

One of Perl's longest *standing selling* points is CPAN. It is among the largest repository of this kind in the Free Software world, and an immense source of resources for its programmers.

In the beginning, there was CPAN, and it was good

- The CPAN was born in October 1995
- As of May 2007, it hosts 11577 modules from 5856 distinct authors
- ...And it is a fast-growing list - During 10 days in May, 76 new modules appeared. Every day, more than 20 uploads were received.
- CPAN is in no small part responsible for Perl's success, giving the users a *huge* set of proven, ready libraries to work with.
- A very important characteristic of CPAN is that it is a *centralized* repository — A single entity. We can perform different operations (QA, searches, ...) throughout the whole module base.
- CPAN modules have mostly a homogeneous (well, *almost*) structure and tracking tools, which makes our work to integrate them *much* easier, technically and socially.



And then, there is CPAN.pm

- Having a centralized repository is good, but locally managing all the installed modules, versions and dependencies quickly becomes *hellish*.
- In February 1996, Andreas König creates a client program (and Perl module) to automatize CPAN downloads, dependencies and versions: `CPAN.pm`. For practical purposes, this solved the problem for users.
- In 2001, Jos Boumans starts working in CPANPLUS/CPAN++, redesigning and modularizing CPAN.pm. It seemed for some time as a natural replacement for CPAN - But in the end its ideas ended up flowing back, and CPAN still reigns.
- They basically cover the *Perl community*'s needs on managing installation, updates and dependency information.

Contents

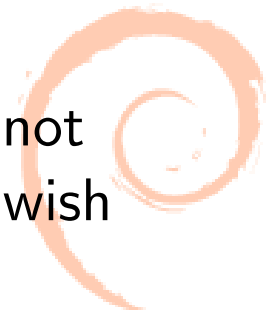
The bigger picture: End users' experiences

But all in all, the use CPAN.pm is -for the end-user- quite overrated.

- Users use *applications*, do not care about languages!
- They don't care about our beautifully crafted tools. Even worse, those tools look like hard-to-understand holes in reality.
- Every language commumity has its favorite tool: PEAR, Gems, Cheese Shop... Does the user really need to know how to use them?
- Our role, as *distribution maintainers*, is to generate a unified toolset for our users to manage their *whole* system coherently — Say, `aptitude install pkgname` (or its point-and-click equivalent).

Free Software distributions: Different support tiers

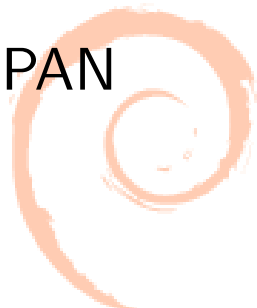
- The Free Software distributions exist to provide users with an integrated, unified system, and take care of passing communication between them and upstream authors as needde.
- Nowadays Debian is the largest FS distribution (20,000 source packages, with strict policies regarding QA and software freedom
- Users can file bugs through the Debian BTS, and the *package maintainer* filters the bug: Is it Debian-specific? Pacakging-related? Was it filed for the right package? *Should we push it "upstream"?*
- The distribution works as a *safety net*, making life easier not only for the user, but for the developer as well. Or so we wish and even believe ;-)



The other way around: Perl supporting the Linux distributions (1)

But to make this attractive to you: What do Perl Mongers gain from us?

- How can you track your modules across many different distributions?
- How can you contact their maintainers *everywhere* to warn them about a big, nasty API change or reorganization?
- What's the best way to notify about disruptive changes *before* bugs are filed (and users pissed)?
- How does each distribution stand up regarding Perl and CPAN up-to-dateness and completeness?



The other way around: Perl supporting the Linux distributions (2)

Sadly, right now I don't have the answers to those questions. But we are working on them.

- The CPAN ports page's section on Linux gave up 2004 (and IMHO should be just removed)
- Gabor Szabo gathered a list of moduless across distributions, which is reasonably up to date, but presents several interesting problems (even what seems as trivial as *what's the current version number of X::Y in Z?*)
- Maintainers from *different distributions* have no contact between each other — So the same problems are likely to appear uncoordinately... And swarm the developer.
  ...Exactly what we wanted to avoid in the first place

A word about propietary systems

We Debianers are known for being passionate about our committment to Free Software ideology; some BSDs and Linux distributions are like us. Some are known for being just the opposite — openly pragmatic. However, we all at least *all* share a Free Software background and worldview. But...

- Some people have to (or –gasp– want to) use a closed operating system, such as Windows, MacOS or the historic Unixes
- Propietary OSs don't (and probably won't) offer this first level of support — At least, not for their non-core components
- If you are in this group... May `$deity` bless you.
- Stick to `CPAN.pm`, and expect no miracles from above.

...My deepest sympathy to you guys.

Contents

How Debian maintainership works (1)

- Debian is a *social project with a technical result*. Around 1000 official developers, hundreds of unofficial maintainers, everybody participating strictly on a voluntary basis
- Each developer is responsable for his own packages. Others might *touch* them, but it is not considered polite to just overtake them.
- When you leave the project, you kindly announce your packages become *orphan*, so that other people take them over — or they can be deleted from the distribution if they have no significant user base.
- And yes, we do track our user base! (material for another talk, I know)

How Debian maintainership works (2)

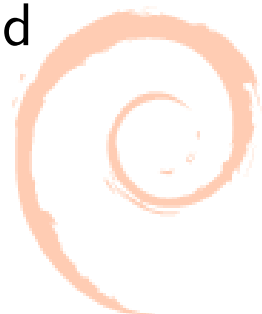
- Being a voluntary project, every now and then people leave the project... Sometimes without saying. To counter this fact, the *Debian QA Team* was born. They take care for the undermaintained packages, and have some tools for following who is active and who is *MIA*.
- It is also common to seek group maintainership for our packages — That decreases response time for any detected bugs, increases the number of people involved in each package, and makes it less likely for packages to become orphaned.

The pkg-perl group

- In late 2003, the `debian-perl@lists.debian.org` mailing list was formed
- In early 2004, Joachim Breitner sent out an announcement inviting Perl modules' maintainers to join an Alioth (GForge) project to start coordinating and avoid duplicating work in packaged Perl modules.
- **No, CPAN is not –and will not be– autopackaged**. Only the modules we personally use or are requested to (and interested in) packaging are.
- Our work is carried out on a public SVN repository at alioth.debian.org, and coordinated via the `debian-perl@lists.perl.org` mailing list and `#debian-perl` channel in `irc.debian.org`.

Current pkg-perl numbers and tools

- As of today, the pkg-perl group maintains 370 of the 1303 Perl module packages in the Debian Unstable branch
- Its growth rate is quite strong — In late May, we were handling only 327 packages!
- Formally we have almost 50 members registered, but the number of *active* members is closer to 10-15
- We are certainly not bug-free, but –as of July 19, 2007– we had only 61 open bugs, two of which were release-critical. Eight of those bugs were reported against Mime-tools, and most are tagged as "waiting for upstream to decide".



DEHS: Keeping track of upstream versions

- One of the most important points in a distribution is *how to track new upstream versions*. In Debian, we devised a mechanism via the magical `debian/rules` file
- ...But debian/watch was meant to be run *locally* by each developer (as it takes *stupid* amounts of time to run).
- Then came the DEHS (`http://dehs.alioth.debian.org/`) — And Alioth makes all the gruntwork! And it is now integrated into all of our maintainer pages (`http://qa.debian.org/developer.php?login=gwolf&comaint=yes`)
- Weaknesses? Yes, of course. It relies on the ability to check periodically over the network. And it often chokes on the silliest timeouts. And reports back nasty errors. So we cannot base our work only on DEHS.

Evaluating and following the repository as a whole

- Until some months ago, we went by with only what you saw so far. But we have adopted hundreds of modules — You can only expect us to start writing some tools!
- Our current scripts were started by Gregor Hermann, and are part of our SVN tree. The scripts basically cover:

build_logs We periodically build all of our packages. This script reports the full logs, sorted by build result

maintainers Checks our SVN directory to check if any of our modules does not have an updated maintainer/uploader information

versions Compares the versions of the modules in our tree, in Debian unstable and in CPAN. Of course, it has the same good and bad points of DEHs

wnpp Track new requests for packaging by our users, orphaned packages, etc.

dh-make-perl

- We will not blindly package all of CPAN... But we want to provide our users with tools so they can still benefit from an unified packaging system
- Back in October 2000, Paolo Molaro made the first upload of `dh-make-perl`. Starting in October 2006, the pkg-perl group adopted it. It is a program that takes a standard Perl module and generates a Debian package for it, including:
  - Name and version
  - Short and long description
  - Infrastructure needed for building (i.e. modules used for tests)
  - Dependencies for using this module
  - Information regarding the current base Perl language and modules installed
- It is messy and not pretty... But it is an invaluable tool :)

Questions? Comments? Whatever?

Thank you very much!

Gunnar Wolf — gwolf@gwolf.org