LCU14-112: The Philosophy of Open Source Development

Wookey, Al Stone, Leif Lindholm, Mon-15-Sep, 2:00pm

The philosophy of open source

4 Freedoms:

1. Run
2. Copy
3. Modify
4. Share

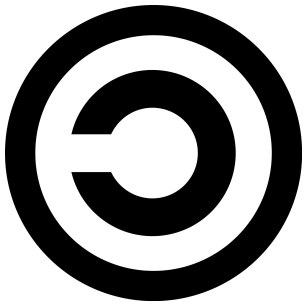
- FLOSS is Free, Libre and Open Source software.
- Both a more efficient development model, and a philosophical movement.
- It's an ecosystem – not planned: survival of the fittest

What is open source?

- The term *Open Source* can be used to mean several different things.
  - In general it refers to software being developed in the open, with source code available for people to build their own modified version.
  - Strictly means things licenced under an OSI (Open Source Initiative)-approved licence
  - Sometimes used for things that aren't actually FLOSS
    - visible, but not redistributable source ('Shared source')
    - 'Open-core' projects
  - Same concept in other fields: 'Open Source Hardware', 'Open Source Buildings'
  - It is not 'public domain' or 'No copyright'.

What is open source?

- There are two dominating philosophies:
  - 'Open Source' emphasises the development model and practical advantages
  - 'Free Software' (Libre Software) emphasises the 4 freedoms as an intrinsic good
  - But it's actually all the same software. All Free Software is Open Source. All Open Source is Free Software. (Obscure exceptions exist).
- Two main licencing models
  - Permissive licenses
    - BSD, ISC, Apache, ...
  - Copyleft
    - GPL, ShareAlike, ...



Permissively licensed software

- Permissively licensed open source projects put few to no restrictions on how a recipient of the source code can use it.
- This tends to be preferred by commercial operators since they can then choose freely how to handle the software in the future.
  - No requirement to 'give back' changes/improvements
  - This also makes it possible to include it in proprietary software, without many implications.
- Permissively licensed software can often be relicensed under a Copyleft license. (But the other way around is not possible.)

Copyleft software

- A copyleft license requires that any later recipient of the software retains the same rights that the code was initially supplied with (reciprocal rights)
  - In source form or binary, standalone or installed in a hardware device
  - This also apply to any derivative works.
- Often less favoured by commercial entities.
  - Forces them to decide up front that they will need to keep the code public for all future.
  - But the requirement comes into play only once the code has been intentionally contributed. (No need to stay away from for confidentiality reasons.)
- Sometimes negatively referred to as “viral”.
  - But there does not tend to be any complaints about the software being available at no cost...

What are the practical differences

- Copyleft software needs source, or an offer of source, with binaries.
  - Ensure you ship source with binaries, or keep a copy (3 years). VCS works.
  - Using a copylefted library may make your code 'derived' and thus copyleft too
  - Permissive code can usually be easily used with copyleft or proprietary software
- Not all licences are compatible.
  - Be careful of mixing incompatibly-licenced code -
  - Interpretations vary somewhat (e.g GPL+OpenSSL, Apache + GPL) are both common but Debian avoids them and lawyers will disagree on permissibility.

Not like proprietary development

- No fixed schedules
- You can't tell other people what to do, just ask/persuade
- Roadmaps are organic, not decreed

- Your manager will not understand
  - 'Which week will it be upstream?'

- Your responsibility for the code does not end when upstream
  - People will come back and ask you to fix it if it breaks, or expect you to help out if they want to change it.

Historical example - GCC/EGCS

- The GNU Compiler Collection (GCC) started its public development in 1987.
  - Being a Copyleft project, all users could modify the source code.

- Developers found it difficult to influence the central development, so created a *fork* of the project and called it EGCS.
  - EGCS ran from August 1997 until July 1999. The existing GCC project accepted that EGCS had a more productive community and scalable development process.
  - EGCS ended up “supplanting” the existing GCC, the projects merged back together, and what had been EGCS became released as GCC 2.95.

More examples

- XFree86
  - Essentially replaced by Xorg
- OpenOffice.org
  - Competing forks: Libreoffice and Apache Openoffice
- FFmpeg
  - Libav attemped replacement fork, but FFmpeg continued.

So why do we do it?

- Open source development lets us pool our resources together to do the things everyone need.
  - Without preventing anyone from specializing in areas where they think there is commercial opportunity.
- Having more commonly useful software out there lowers the barrier of innovation for everyone.
- Lets you just get stuff done: you don't have to wait for another person/company to fix/do something.

What does open source give you

- Expect the unexpected.
  - Catering for only the scenarios you predict means that only those things are likely to happen.
  - Who knows what crazy thing some bored student can come up with when given full opportunity to play around?
  - Software that doesn't come with a load of aggravation like flexlm licence servers/nagware/adware/bundling.
- Allows innovation: free software made google, amazon, facebook, yahoo, wikipedia, etc possible
  - Eben Moglen - “Innovation under Austerity”

What do you want from the community?

- Their work!
  - Code, ideas, bugfixes, testing
- Their effort
  - So you need to show them that you are worth it.
- Their expertise
  - They know more about their software than you do (usually!)
  - Upstream can often implement something in hours that would take you days or weeks. Build good relationships

What does the community want from you?

- Work that benefits the community.
  - Patches that fit into the existing codebase.
  - And do not break existing code (e.g other architectures)
  - Remember: *nobody has to take your stuff!*
- Interesting stuff
  - Featured they had not thought of
  - Work no-one else knows how to do (or it would take them a lot longer)
- Information
  - NDA-only documents are a hostile act.
  - Registration-only docs limit audience/impede adoption

The anatomy of an open source project

- Because projects are run by different individuals, with different goals and opinions, they work differently.
  - Mailing lists
    - Some want all patches to be sent as attachments.
    - Some absolutely do not want patches to be sent as attachments.
  - Communications
    - Some use mailing lists exclusively
    - Some use mailing lists, forums and irc
    - Some run almost entirely on IRC
- Find out how to interact before blundering in.
  - Allow a little time to discover this

Rules of engagement - IRC

- Do not ask if you can ask a question - just ask the question
- Do not ask individual people, unless you *know* that person is the only one who has the answer. Especially do not ask in private chat.
- Try to stay on-topic for the channel
- Rules on linaro- channels are a lot more loose - these are effectively our virtual office.

Rules of engagement - mailing lists

- Mailing lists can differ a lot in how they want communication to happen
  - Some lists want all patches sent as attachments, some want patches to never be sent as attachments, some want only patches sent with git-format-patch / git-send-email.
  - Most lists loathe HTML formatted email.
  - Some lists refuse email with automatically added legal disclaimers.
  - If there is time, subscribe to the mailing list and read the postings for a few weeks, to learn the etiquette of that particular list.
    - Failing that, read through some of the list archives.
- But, like irc, do not contact individuals directly off-list
  - *Even* if you know they are the only one to have the information you are after. Questions answered in the conversation will be logged in an archive if on the list.

Rules of engagement - common

- Always search for the answer first (google, baidu, bing)
  - Even if the hits are not relevant, or you do not understand them, showing people that you have made that effort makes them a lot more likely to want to help.
- Provide lots of information
  - Preferably as a Short, Self Contained, Correct (Compilable), Example - http://sscce.org/
  - People who are helping you for free will not put a lot of effort into trying to find out what you are actually asking.

How can we help you?

If you are in any way uncertain about how to act in a given situation, experienced Linaro people are available to help out.

#linaro-mentors IRC channel

linaro-dev@lists.linaro.org mailing list

More about Linaro Connect: http://connect.linaro.org
More about Linaro: http://www.linaro.org/about/
More about Linaro engineering: http://www.linaro.org/engineering/
Linaro members: www.linaro.org/members