Open Source Fundamentals

Wookey

2021.07.12

Who is Wookey?

Wrote first Free Software in 1991 (survex, GPL2)

Armlinux 1999
Debian Developer 2000
Guide to ARMLinux for developers 2001
Emdebian 2004-2014
Debconf 2005

Linaro at startup 2010
AArch64 bootstrap

The philosophy of free software

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
  - Same concept in other fields: 'Open Source Hardware', 'Open Source Buildings'
  - It is **not** 'public domain' or 'No copyright'.

- Sometimes used for things that aren't actually FLOSS
  - visible, but not redistributable source ('Shared source')
  - 'Open-core' projects (Gitlab, Elastic, MongoDB)

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: e.g NASA OS Agreement)

https://www.gnu.org/philosophy/open-source-misses-the-point.en.html

https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licences#Approvals

History

- 60s. All software effectively free
- 70s,80s Commercial Unices
- 1984 GNU project
- 1985 Free Software Foundation
- 1991 Linux kernel
- 1993 Slackware, Debian, Red Hat
  https://en.wikipedia.org/wiki/File:Linux_Distribution_Timeline.svg
- 1994 Debian Free Software Guidelines
- 1997 ‘Open Source’
- 1998 OSI, DFSG->OSI definition

- 20+ years of acceptance and adoption, but some ructions in last couple of years (ethics, open core)

‘Open Source’ origins

‘Open Source’ term coined by Christine Petersen (Foresight Institute) late 1997.

Group wanted less confusing business-friendly name. (Gratis vs Libre)

Todd Anderson, Tim Oreilly, Red-Hat Founders

Controversial at the time.

OSI (Open Source Institute) set up by Eric S Raymond and Bruce Perens

https://opensource.com/article/18/2/coining-term-open-source-software

Open Source Definition

1) Free Redistribution
2) Source Code
3) Derived Works
4) Integrity of The Author's Source Code
5) No Discrimination Against Persons or Groups
6) No Discrimination Against Fields of Endeavor
7) Distribution of License
8) License Must Not Be Specific to a Product
9) License Must Not Restrict Other Software
10) License Must Be Technology-Neutral

- https://opensource.org/osd-annotated
- Essentially Debian Free Software Guidelines + #10

So what’s the point?

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

How does open source work?

- It’s a community, not a company
- You can’t tell other people what to do, just ask/persuade
- Roadmaps are organic, not decreed - ‘do-ocracy’
- Project managers do not understand

  ‘Which week will it be upstream?’

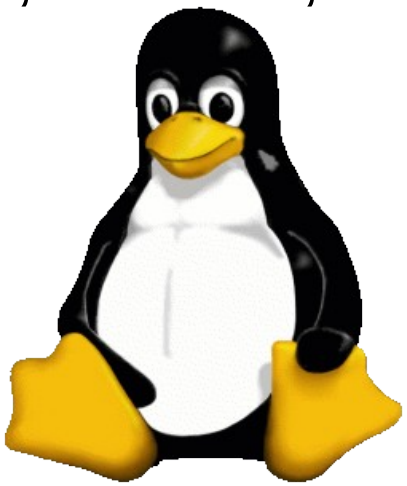
- Your responsibility for the code does not end when Upstream

  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.

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?
  - Wide range of codepaths tested
- 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’



Understanding the community

- A community is composed of:
  - Academics (students, scientists, teachers)
  - Hobbyists
  - Workers (from companies or freelance)

- The community takes predominance over individuals
  - → Consensus is the goal
  - Collaboration is the cornerstone of Open Source

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/other architectures.
  - Remember: *nobody has to take your stuff!*
- Interesting stuff
  - Features 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.

What do you want from the community?

- Their work!
  - Code, ideas, testing, bug reports, bugfixes
- 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
  - Building standing and relationships is effective – knowing who to find on an IRC channel can be worth months of corporate negotiation.

Collaboration

Collaboration relies on tools

- Development
  - Distributed version control system: git
  - Compilation tools : gcc, make
  - Debugging: gdb
  - Etc …
- Communication
  - Discussion: Mailing lists, Discourse
  - Chat: IRC, Matrix/Riot/Element
  - Text sharing: pastebin
  - Bug trackers: BTS, Bugzilla, Trac, Gitlab, Github
- Find out how to interact before blundering in

Don’t be scared

- Find out how to interact before blundering in
- Understand the actors in the Project
- Be prepared then send your first patch

Don’t be offended

- Comments are always a good thing, that means the changes raised some interest
- Comments can be tough: stay factual, stick to the technical aspect and give numbers to support your position
- Comments can spot an issue or a misdesign you missed
- Maintability is the priority, you may be asked to redesign everything

Don’t be demanding

- Comments can take some time: be patient
- There is no schedule / no deadline
- The community may be busy
- There is no obligation to merge the change

Don’t be selfish

- The changes must be designed as part of the community, not as an individual
- Changes for the purpose of one group of persons or a company have 100% chance to fail to be merged
- Working in the Open Source, is working as part of a community

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 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 and top-posting
  - 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 - General

- Always search for the answer first (DDG, google, baidu)
  - 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.

Forking

Always an option.

- Usually born of frustration
  - Slowness, Direction, Process
- Can dissipate effort/resources, but sometimes effective/necessary.

Examples

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

Forking - Historical example - GCC/EGC

- 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.