



# “Free” software: a look back, a look ahead

---

*By Randal L. Schwartz,  
Stonehenge Consulting Services  
<merlyn@stonehenge.com>  
<http://www.stonehenge.com/merlyn/>  
version 1.4 at 2 Jun 05*

# What do we mean, free?

---

- *Unencumbered by what you can do with it: “free” as in “freedom”*
- *Often also for zero cost: “free” as in “free beer”*
- *Generally “open source”*
- *Not necessarily “public domain”*

# Common “free” licenses

---

- *GPL*
- *LGPL*
- “*Anything but \$evil\_purpose*”
- *BSD-style*
- *Artistic License*
- “*public domain*”

# The GPL “lock in”

---

- *The GPL means you get free software*
- *But whatever you do with that also has to remain free*
- *So, some people can't touch it, since they want to leverage, but not return back*
- *Hint: I'm not a hardliner FSF guy*
- *I believe that there's a useful compromise*
- *And that means **more** free software!*

# The LGPL's loophole

---

- *LGPL is generally for libraries*
- *You can link your private stuff*
- *But if you modify the library, you still need to distribute that!*

# Specialized “no evil” licenses

---

- *You can use this “except for commercial”*
- *Or “except for military”*
- *Or “except for competing with me”*
- *Problem: can’t include this in more generally licensed distros!*

# BSD-style: a fair compromise

---

- *Use this as you wish*
- *Just give us credit somewhere*
- *And the license is sticky*
- *Good deal for most applications, including locked-up applications*
- *Upsets the FSF “hardliners”*

# Aside: the “unix source license”

---

- *Permits Unix Source License holders to share patches with other USL holders*
- *Forbidden to share with non-USL!*
- *Generally university faculty and students*
- *Interesting model that probably inspired other “open” source licenses*

# Perl's Artistic License

---

- *Use this as you wish*
- *But if you change this, don't call it "Perl" anymore — We don't want to confuse folks*
- *About as close to "public domain" as you can get*
- *In fact, Artistic 1.0 has been called so loose that it's unenforceable!*

# Public domain

---

- *Use this as you wish*
- *That's it*
- *Even re-license it and call it your own*
- *Oddly enough, very few things are strictly in the public domain*
- *No copyright or license = copyrighted!*

# Back when I was a kid...

---

- *You bought your computer*
- *You bought things to run on your computer*
- *That was it, mostly*

# Along came “shareware”

---

- *Smart people hacking their own stuff*
- *Not selling it, but “giving it away”*
- *Occasionally with restrictions*
  - *Not for commercial use*
  - *Send \$5 to \$my\_charity*

# Shareware - how it worked

---

- *Uploaded to “BBS systems”*
- *Downloaded to your machine—very slowly*
- *Great way to get little gadgets out there*
- *Sometimes used as a demo for commercial apps*

# The GNU project

---

- *My first awareness: GNU Emacs*
- *Creating social change about software through a politically pressuring license*
- *Founded by Richard Stallman (RMS)*
- *Goal: produce a complete OS (Hurd) and tools that could be used freely*

# An aside: the name of Linux

---

- *I've met Richard Stallman and Linus Torvalds on several occasions*
- *Linus created Linux*
- *Richard is wrong to be trying to rename it to GNU/Linux*
- *The name GNU/Linux offends Linus*
- *Please respect Linus: he's the nicer guy of the two, and can call Linux what he wants*

# The rn newsreader

---

- *The first significant large “free” software I used on a regular basis*
- *Larry Wall wrote it for himself in 1981 because he needed it, and gave it away*
- *Mostly because he wanted to, and could*

# The significance of 'rn'

---

- *Many people switched to rn because it was “free” (zero cost) and powerful*
- *'rn' became the most popular news reader from 1981 to 1990*
- *Larry Wall got status and power from that*
- *Even though he didn't make 'rn' for that*

# Patch

---

- *Created by Larry, again to fill a niche*
- *No easy way to fix 'rn' without shipping entire new source*
- *This was expensive over slow lines*
- *Again, de facto standard created from necessity and distribution of free stuff*

# Perl

---

- *Again, created by Larry Wall to fill a personal need, but given away*
- *Initially under 'rn' license (similar to Artistic license), but later added to GPL*
- *Enabled a large community to develop rapidly, and deployment to many sites*

# How free software appears

---

- *Someone needs something*
- *They write it for themselves*
- *They share it with others for free, presuming others might have the same needs*
- *Others reply by contributing back*
- *Popular software develops communities*

# How having source helps

---

- *Don't have to worry about company disappearing: you have automatic escrow*
- *Understand badly documented APIs*
- *Fix bugs, adapt the code to your needs*
- *Pay others to do same*

# How a contribution happens

---

- *Ask the developers to add some feature*
- *Ask a third party to do it*
- *Pay developers (or third party)*
- *Code it yourself, submit the patch*
- *Code it yourself, apply your patch to each release*

# Practical example: pp.el

---

- *I wrote a pretty-printer for GNU Emacs Lisp: I wanted it, and it hadn't been done*
- *I posted it on Usenet*
- *RMS liked it, asked me to release it under GPL and assign copyright to the FSF*
- *Now it's part of GNU Emacs*

# Practical example: Perl list slice

---

- *As a member of the Perl 3.0 alpha test team, I wanted `(LIST)[SUBSCRIPTS]` to work*
- *I asked for it on the mailing list*
- *Larry Wall wrote the code, and put it into the final release*

# Practical example: Perl coderef deref

---

- *Perl 5.003 required `&$coderef(@args)`*
- *Perl 5.004 needed `$coderef->(@args)`*
- *I bet someone I could get it added  
between late-beta and the final release*
- *I told Chip that “Larry wanted it”*
- *Chip added it, just as I had planned*
- *I won the bet!*

# Practical example: perlboot

---

- *I wrote some courseware about Perl objects, and turned it into two magazine articles*
- *I repackaged it as a “perldoc” and contributed it to the Perl distro*
- *Now everyone can read it!*

# Practical example: File::Finder

---

- *I saw “File::Find::Rule” and wanted a better evaluation engine and more compatibility with the find command*
- *I wrote “File::Finder” and submitted it to the CPAN*
- *I got paid to write a magazine article about the engine*

# Practical example: CGI::Prototype

---

- *One of my clients needed a web app*
- *I created a framework*
- *I got permission to contribute it publicly*
- *I reused it on other projects*
- *Other people are using it now too!*
- *I got paid to write free software, but only because some parts of it aren't free*
- *This is why FSF hardline can hurt*

# The CPAN

---

- *In the spirit of Perl's source code*
- *Contributions can happen*
- *Results can be leveraged*
- *Truly a "community potluck" where everyone can give what they can, take what they need*

# Other ways of contributing

---

- *Answer questions online*
- *Submit doc patches*
- *Submit bug fixes*
- *Run test harnesses (smoke tests)*
- *Contribute money to good organizations*

# Community communications

---

- *Mailing lists*
- *Web-based discussion software*
- *Usenet groups (public or private)*
- *IRC channels*
- *Face-to-face meetings at user groups and conferences*

# Importance of a community

---

- *Knowing you're not the only person using the software*
- *Getting questions answered*
- *Sharing results and insights*
- *Answering questions for others (take a little, give a little)*

# Open source and security

---

- *Public scrutiny reveals bugs better*
- *Both for the good guys and bad guys!*
- *Bug fixes can be coded in-house rapidly*
- *Third parties can create bug fixes*
- *Vendors sometimes hide fixes because they are embarrassing or lawsuit provoking*

# Open source and the future

---

- *There'll always be open source*
- *People have needs, and will write code to solve those needs*
- *People want to share what they've done*
- *People need to modify what others have done to suit their own situation*
- *Locked-up software is more expensive to create, maintain, or use*

# What you should do

---

- *Find ways to promote open source*
- *Find ways to contribute to open source projects*
- *Look for “compromise” projects (part open, part private) for funding*
- *Encourage your government organizations to use open source and open data standards*

# In summary

---

- *Free software has been around for a long time*
- *Free software is here to stay*
- *Free software is better for many applications*
- *I'm here to help. How about you?*

# How to contact me

---

- *Email: [merlyn@stonehenge.com](mailto:merlyn@stonehenge.com)*
- *Website: [www.stonehenge.com/merlyn/](http://www.stonehenge.com/merlyn/)*
- *Phone: +1 (503) 777 0095*