Five years of pdfTEX – lessons learned

Martin Schröder

ConTEXt 2008, 20th August—24th August 2008, Bohinj

Why this talk?

- From 2003 till 2008 I was the maintainer of pdfTEX
- I want to share some of the things I learned about managing an open source project over the years
- So you have an idea for a project – what next?

Guidelines

- Be as professional as possible
- Have fun!
- Release early and often
- “All labor that uplifts humanity has dignity and importance and should be undertaken with painstaking excellence.” (Martin Luther King, Jr.)

License

- One of the first things you should decide for your project is the license
- Two basic flavours:
  - GPL. “Free”, and it makes sure that the code stays “free”. Two incompatible subflavours: GPLv2 and GPLv3
  - BSD. Free. “Do with it what you want, but don’t claim you wrote it.”
- For TEX macros there’s of course the LPPL

Website

- Your project needs a website; it is the first thing Google will find or people will search for
- Keep it up to date and organize it well
- Consider setting up a blog

Revision control

- Your project needs a revision control system; either host it yourself or use public services (TUG, sourceforge, foundry, savannah, . . .)
- A public web interface is very useful
- Either centralized (CVS, subversion, perforce) or distributed (git, darcs, mercurial)

Tracker

- Set up a public bug tracking system
- There is a multitude of software, but beware of being locked-in; an export function is a must

Mailing list

- Set up a mailing list
- Direct commit messages of the revision control system and mails from the tracker to this mailing list
- Later on in your project you might set up a dedicated mailing list for the developement of your project and use the former list for user support

Code hygiene

- Every source file must have a license statement
- Every source file should have a © statement
- If your revision control system supports them, use its keywords
- Do not let any compile time warnings slip through; if you are using gcc, `-Wall` is your friend
- Make sure that you can identify the source code revision that led to a given binary
- Agree on a code style early on and enforce it, if possible with software like `indent`
- Free coding styles like GNU or OpenBSD are very helpful
- Try hard to keep your software portable. GNU is *not* Unix

Documentation

- User documentation is needed and must be kept up to date
- Include a readme, a change log, a news file and installation instructions
- Proper release annoucements are also a must

Socials

- If you are on your own, you can skip this section
- Consider the motivations of your team members: They are in it either just for the fun of it, for the merits or because they get payed for it. Either way they can at any time just drop off the earth or fork the project
- Decide upon one model for your project: Benovelent dictator, meritocracy or democracy
- Decide upon a clear goal for your project and maintain some kind of road map
- Decide upon rules for conflict solving, even if you think you will not need them. You will
- If your team members are not all close friends of yours, make some effort on team building
- If there are changes of responsibility, make them clear to everyone involved and document them

Meetings

- Try to get the team together in meatspace from time to time
- Again: Be as professional as possible: Set up an agenda and have a protocol. Or just drink beer and have fun
- Hackathlons are very useful

Public relations

- Your project needs some PR; after all you want people to use it
- Your foremost tool is your website
- A blog helps
- Maintain entries for your project at sites like freshmeat and wikipedia
- Enlist and participate on related mailing lists and blogs
- Conference talks and papers are also useful