# FSEM 1111 Computer Security – from a Free Software Perspective

Christian Grothoff
christian@grothoff.org

http://grothoff.org/christian/

# Overview

- Free Software Philosophy – What does society need?
- Open Source Philosophy – What do developers need?

# What does society need?

# What does society need?

- It needs information that is truly available to its citizens.
- Non-free software is a black box that we can't study or change.

# What does society need?

- Society also needs freedom.
- When a program has an owner, the users lose freedom to control part of their own lives.
- Free software is a matter of freedom, not price.

# What does society need?

- Cooperation is more important than copyright.
- Copyright restricts cooperation (you may not be allowed to give a copy to a friend).
- Underground, closet cooperation does not make for a good society.

$\Rightarrow$ Society needs to abandon copyright.

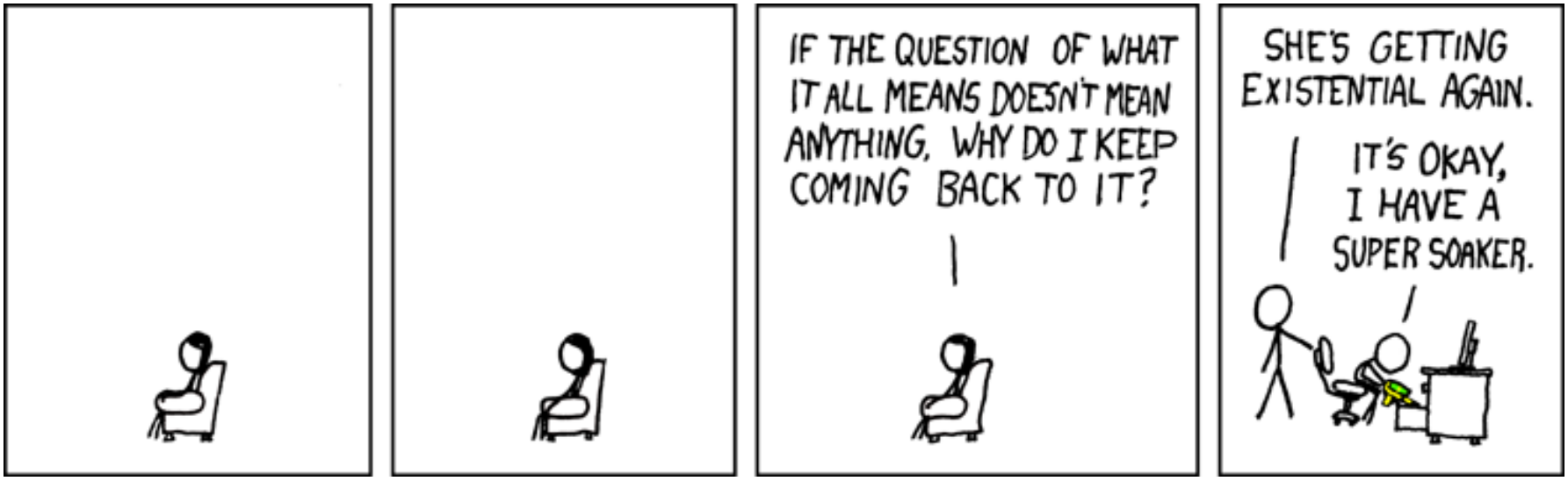
# Free Software in Education

- Free software is cheaper (including hidden costs for upgrades and licensing costs outside of school)
- Free software permits students to learn how software works.
- Teach civil behavior: If you bring software to school, you must share it with the others.
- School should teach students ways of life that will benefit society as a whole.

$\Rightarrow$ Help society escape domination by megacorporations.

# RMS

# Philosophy[1]




[1] Thanks to xkcd.org

# Requirements for Software Development

- Every good work of software starts by scratching a developer's personal itch.
- Good programmers know what to write. Great ones know what to rewrite (and reuse).
- If you have the right attitude, interesting problems will find you.
- When you lose interest in a program, your last duty to it is to hand it off to a competent successor.

# Requirements for Software Development

- Treating your users as co-developers is your least-hassle route to rapid code improvement and effective debugging.
- Release early. Release often. And listen to your customers.
- Somebody finds the problem, and somebody else understands it. And I'll go on record as saying that finding it is the bigger challenge.
- Users with access to the source code produce better problem reports.

# Requirements for Software Development

- The next best thing to having good ideas is recognizing good ideas from your users.
- A security system is only as secure as its secret. Beware of pseudo-secrets.
- To make the bazaar model work, it helps if you have skill at charming people.
- Provided the development coordinator has Internet access, and knows how to lead without coercion, many heads are inevitably better than one.

# Traditional Software Development

The role of managers is

- To define goals and give everybody direction
- To monitor and make sure crucial details don't get skipped
- To motivate people to do boring but necessary drudgework
- To organize the deployment of people for best productivity
- To marshal resources needed to sustain the project

# Anatomy of a GNU Project: GNU libextractor

- Started with 2 students to address a need for GNUnet
- Concrete, simple goal: *extract metadata from files*
- Minimal prototype after a day: *handle ID3 tags for MP3*
- First public release done after within a week
- Instant user base: quickly integrated with GNUnet

# Growth

- Design allowed incremental growth: add new file formats
- Over the next 3 years, formats were added roughly at the rate of one per month
- After 3 years (in 2005) libextractor becomes an official GNU project
- Dozens of people contribute improvements (bugfixes, translations, support for additional formats)

# Today

- Mature, well-tested package available for many platforms
- Used by a dozen other projects
- Development has slowed down:
  - Low-hanging fruit have been picked
  - Most desired features are available
  - Core developers busy with other projects

# Lessons obeyed that helped

- Scratch a personal need
- Make sure there is no other solution out there!
- Keep it as simple as possible
- Release early, release often
- Be lazy – encourage others to contribute

# Non-obvious lessons

Say no:

- To buggy code contributions (or clean them up)
- To patches that do not fix issues
- To maintain binary distributions for various platforms (if you are the project maintainer)
- To feature requests that are clearly outside of the scope of the project

But be *very* polite about it!

# Questions

?