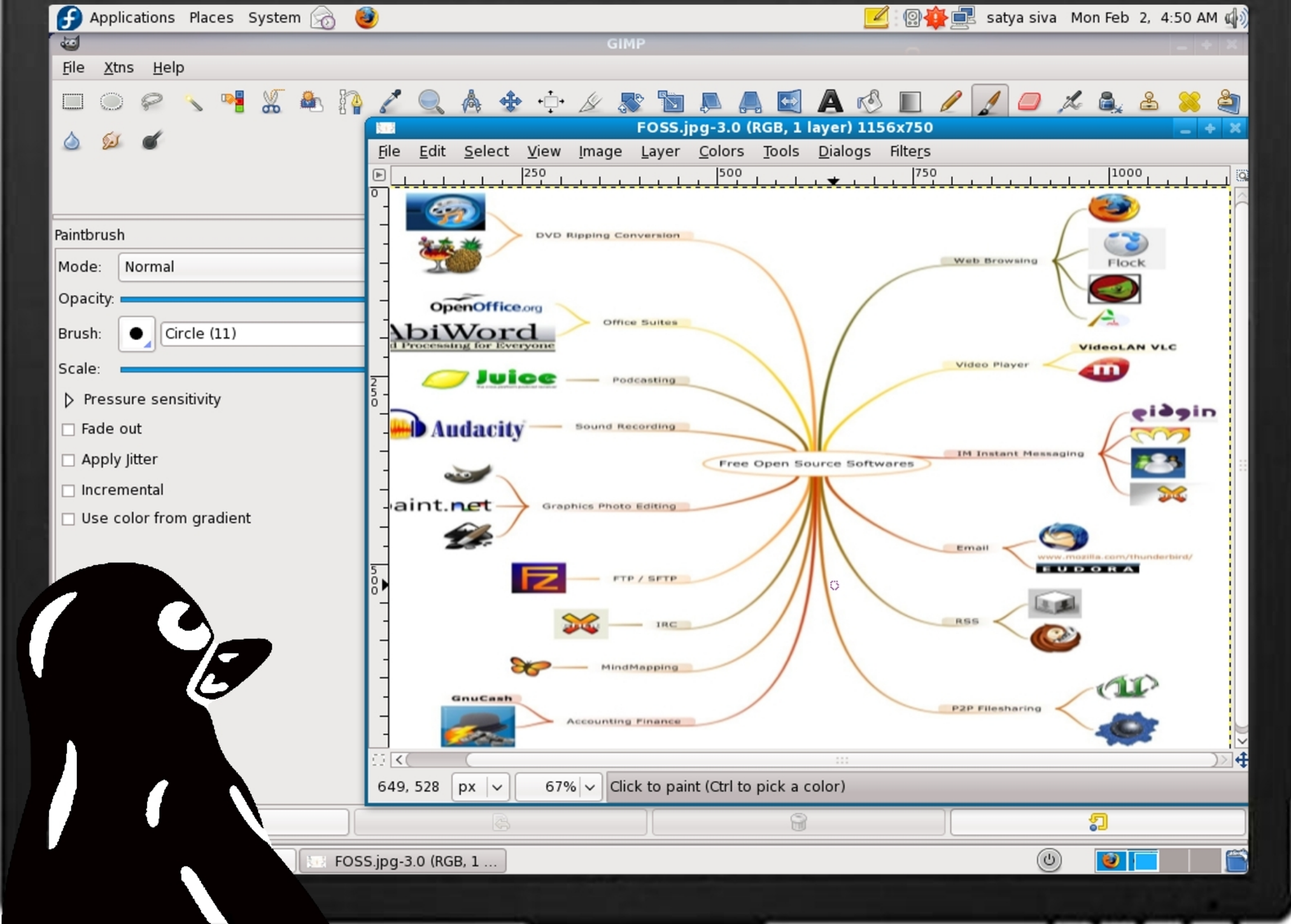
How to Contribute to FOSS

Satya Komaragiri


Applications Places System
satya siva Mon Feb 2, 4:50 AM
GIMP
File Xtns Help
Paintbrush
Mode: Normal
Opacity:
Brush: Circle (11)
Scale:
Pressure sensitivity
Fade out
Apply Jitter
Incremental
Use color from gradient
FOSS.jpg-3.0 (RGB, 1 layer) 1156x750
File Edit Select View Image Layer Colors Tools Dialogs Filters
Free Open Source Softwares
DVD Ripping Conversion
Office Suites
Podcasting
Sound Recording
Graphics Photo Editing
FTP / SFTP
IRC
MindMapping
Accounting Finance
Web Browsing
Video Player
IM Instant Messaging
Email
RSS
P2P Filesharing
649, 528 px 67% Click to paint (Ctrl to pick a color)
FOSS.jpg-3.0 (RGB, 1 ...

The Talk

1. The Philosophy of FOSS

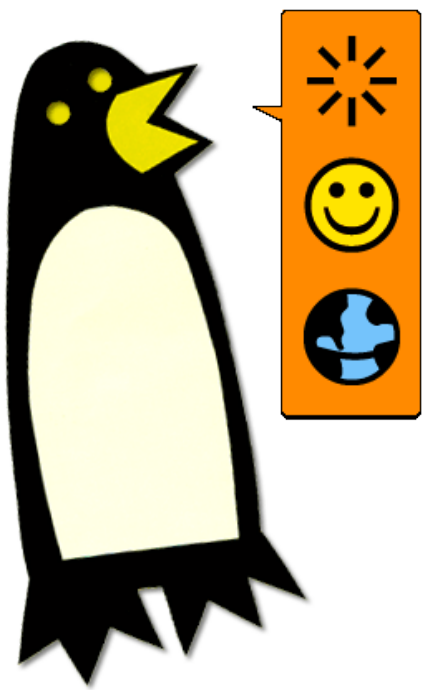
2. Why Should One Contribute to FOSS

3. What it Takes to be a FOSS Hacker

4. How to contribute?
   i. By coding
   ii. Without coding

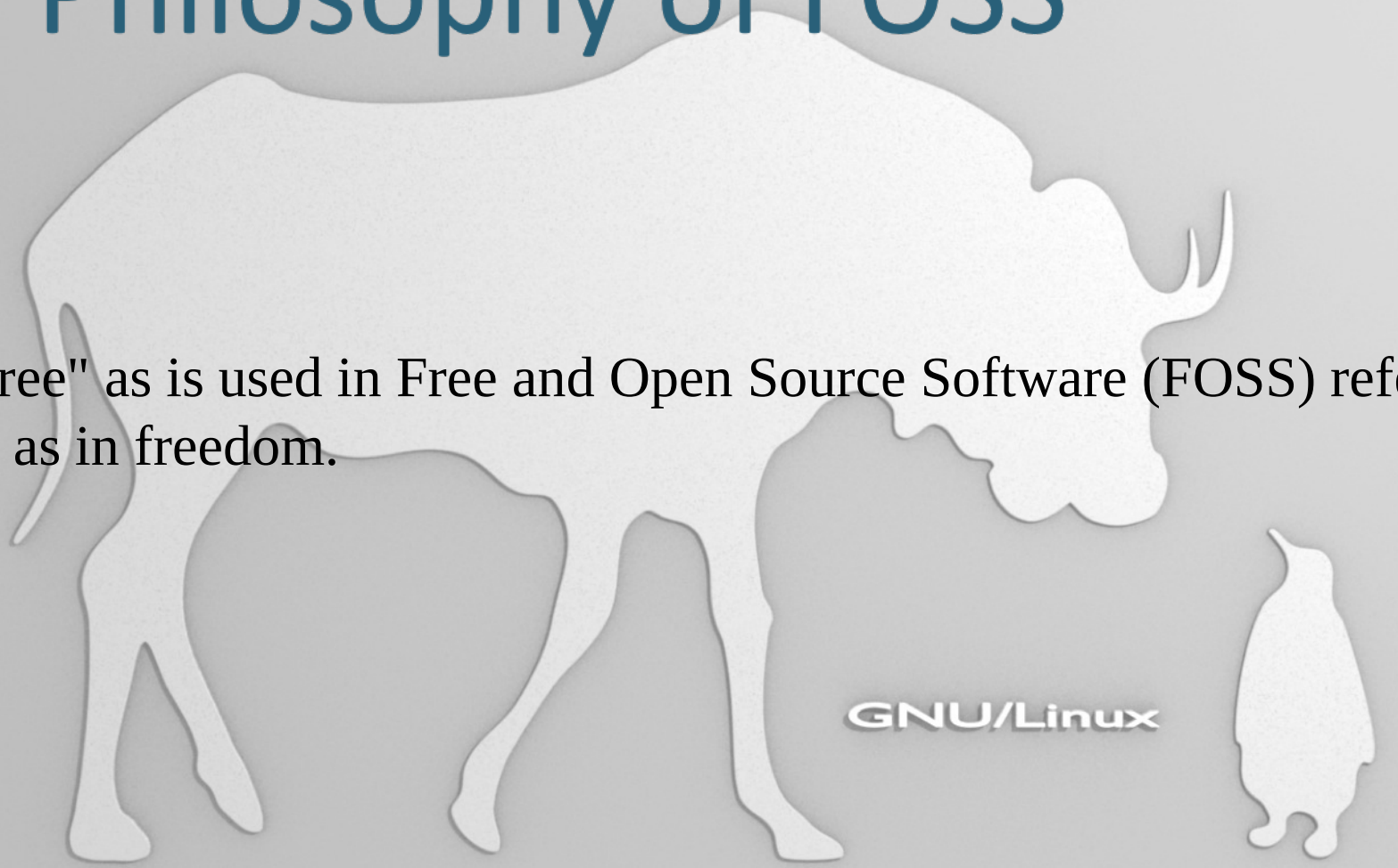
5. Some examples.
   i. Fedora
   ii. OLPC

6. Questions and Answers



The Philosophy of FOSS

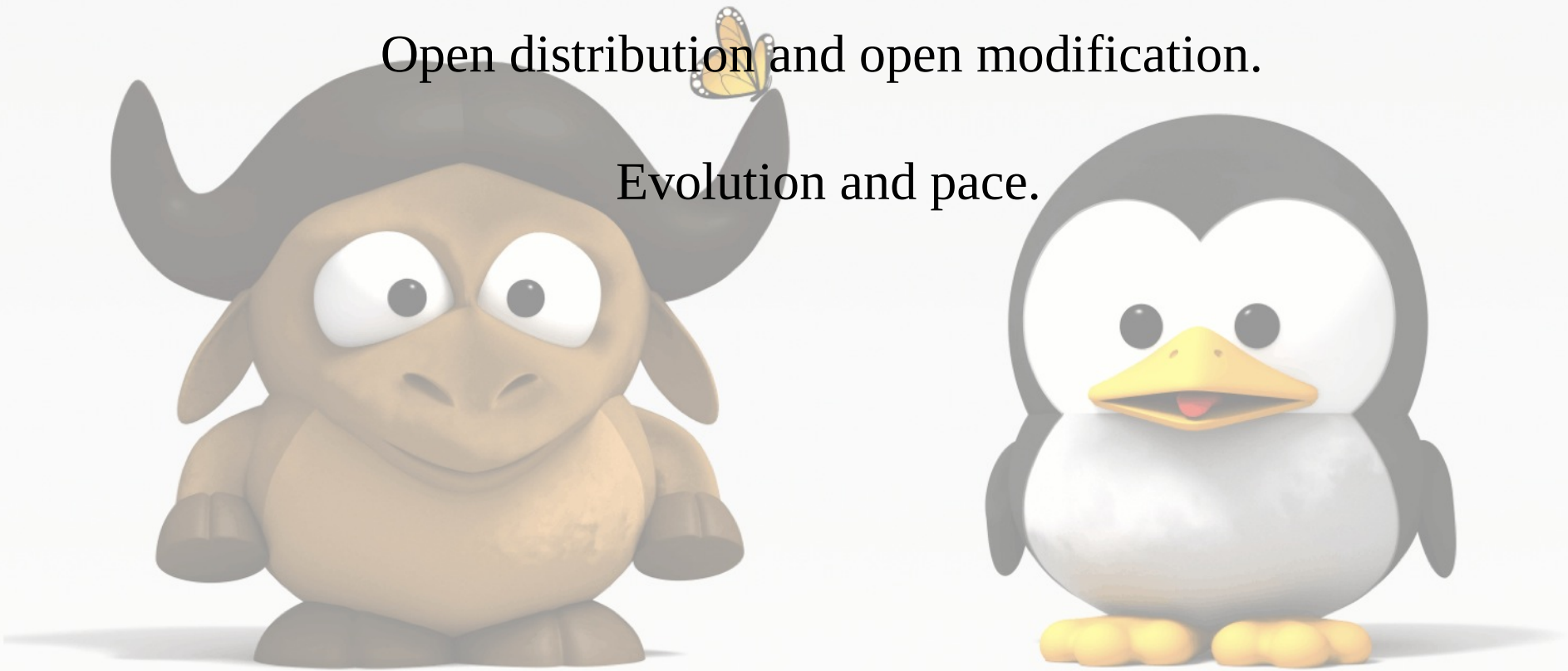
The "free" as is used in Free and Open Source Software (FOSS) refers to free as in freedom.



The Philosophy of FOSS (cont.)

Open distribution and open modification.

Evolution and pace.



Why to Contribute to FOSS

- Gaining Knowledge.
- New Challenges.
- Introduction to Real Life Coding.
- Opportunity to Gain Expertise.
- Opportunity to Experiment.
- Collective Learning.
- Self Assurance.

Why to Contribute to FOSS (Cont.)

Recognition.

Communication Skills.

Market Value.

Giving Back.

What it takes to become a FOSS Hacker

Have the right attitude.

The world is full of fascinating problems waiting to be solved.

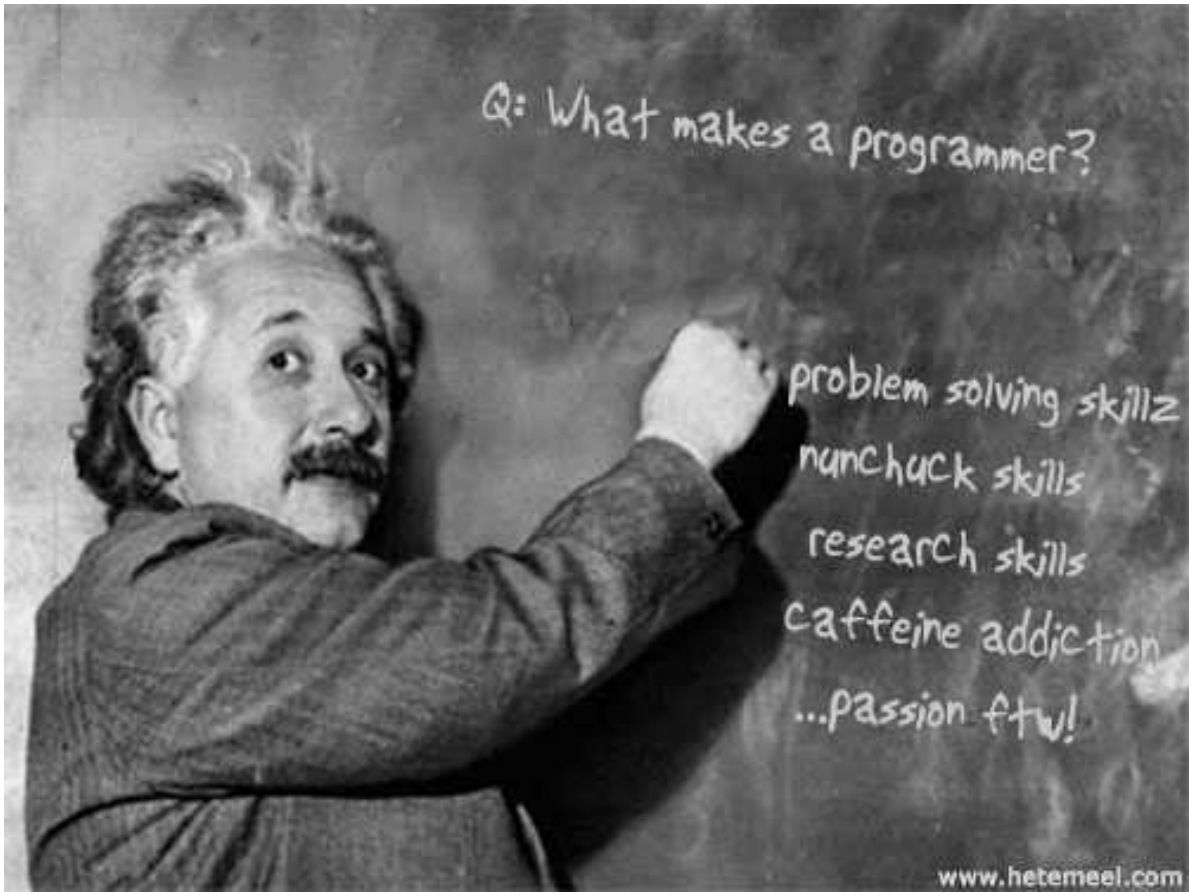
No problem should ever have to be solved twice.

Boredom and drudgery are evil.

Freedom is good.

Attitude is no substitute for competence.

Voluntary mutual help is good.



What it takes to become a FOSS Hacker (Cont.)

Be motivated to put in that effort.

Derive thrill from solving problems, sharpening your skills, and exercising your intelligence.

Have faith in your own learning capacity.

You are never to young to start.

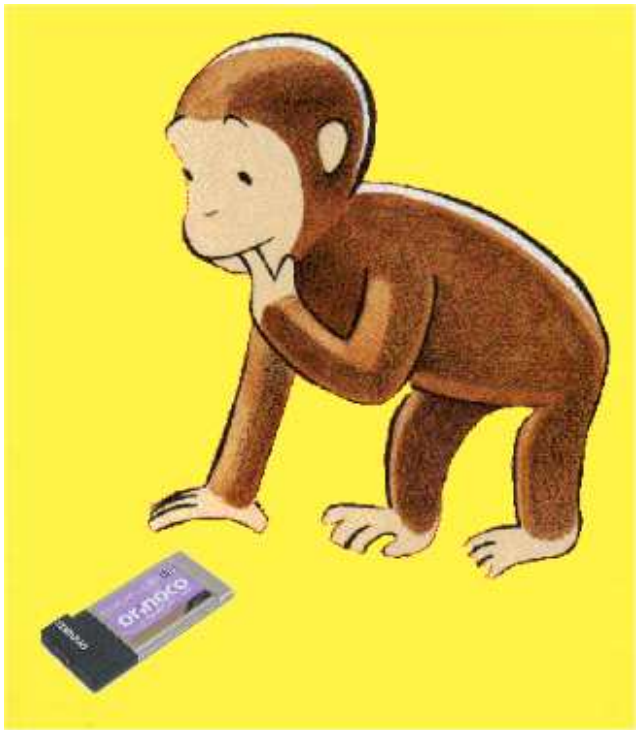
How to Contribute

Everyone can be a contributor.

A project needs people with varied skills.

For the purpose of organizing this talk, contribution is categorized as

i. by coding
ii. Without coding



Contributing by Coding

Contributing Code :

- Contributing to an existing project.
- Starting your own project.

Contributing by Coding (Common Tips)

- Use different open source projects.
- Read a lot of code, and learn from that.
- Learn how to create and apply patches.
- Learn how to make use of GNU Autotools (most projects use them).
- Learn how to use revision control systems.
- Become familiar with open source licenses.
- Learn how to participate in public mailing lists, IRC etc. Netiquettes.

Choosing a Project

For your first project, you may want to choose a project which:

- Uses the programming language you know.
- Is active, with recent releases.
- Already has a team of more experienced developers.
- Has some part you think you can immediately start. implementing without modifying the existing code too much.
- Has active discussion lists and bug reports, receives and implements requests for enhancement etc.

Contributing to an existing project

- Learn the specific tool chain used by the project before contributing.
  - the build system
  - harness for unit tests
  - bug tracking software
  - version control system
  - communication channels used by the developers and users.

- When reading code, consult include files for info on library functions.

- Start off commenting existing code where it needs it.

- Write some documentation on the architecture of the program.

- Write your own small programs just to learn the language and libraries.

- Experiment by making changes to your local copy of the code.

Contributing to an existing project

- Adhere to the maintainer's coding and formatting standards
- Start small, with one-line changes to existing programs but gather the changes and submit as one clean-up patch.
- Test your code thoroughly before you submit it.
- Each project has its own distinct methods for and submitting contributions. Follow them.
- Join the mailing lists. Respect and maintain discussions.
- Respond and send feedback.
- Don't get discouraged when your patches are rejected (they will be!)

Contributing to an existing project

- Add features to the software rather than change in the beginning.
- Work on something no one is working on at the moment.
- Search documentation and archives before asking doubts.
- Do your own research.
- Don't give up

Starting your own project

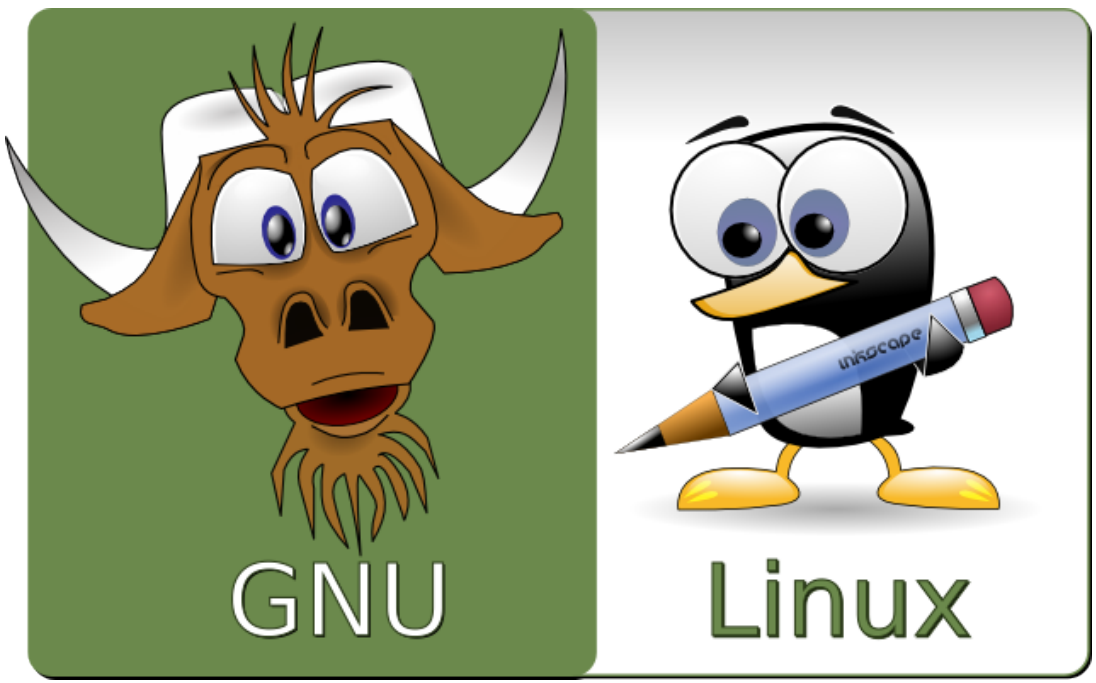
- Pick an existing and approved Open Source license .
- Make sure your website states that your project is "Open Source" and under which license it is.
- Host it on free hosting sites for open source projects like SourceForge.
- Publicize it on Forums or your blog etc. To attract more contributors.
- Encourage contributions.
- Give other contributors a proper infrastructure:
    - webspace for documentation,
    - setting up a mailing list,
    - a bug database with web interface
- a revision control system.

Starting your own project

- Do the releases time to time.
- Do not try to invent a different license.
- If you use some components of another Open Source project, be sure to respect its license.
- Be professional and mature in handling differences of opinions among contributors.
- Avoid private communications on the developments.

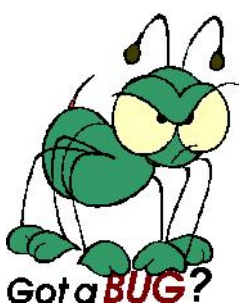
Contributing without Coding

- Contribute Quality
- Contribute Documentation
- Contribute Support
- Contribute Publicity



Contributing Quality

- Submit bug reports
- Triage
- Suggest new features, options and ways to improve the framework
- Create artwork (icons, backgrounds, logos)
- Help maintain a web site for an Open Source project
- Design a better user interface for your favourite Program (GLADE and Qt Designer are great for mocking up a new UI)

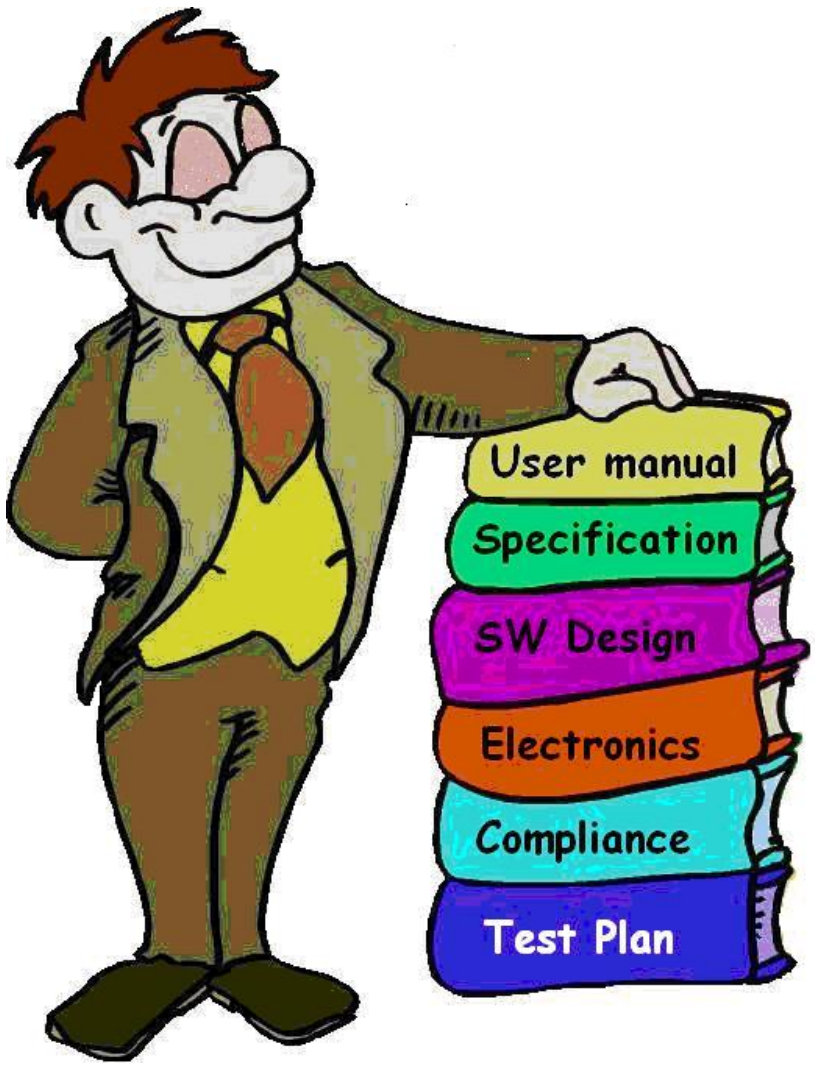


Contributing Quality (Cont.)

- Run usability studies
- Create validation or regression test cases
- See how a program handles streams of random data
- Get the program to compile on a new platform
- Read relevant standards and make sure the program follows them

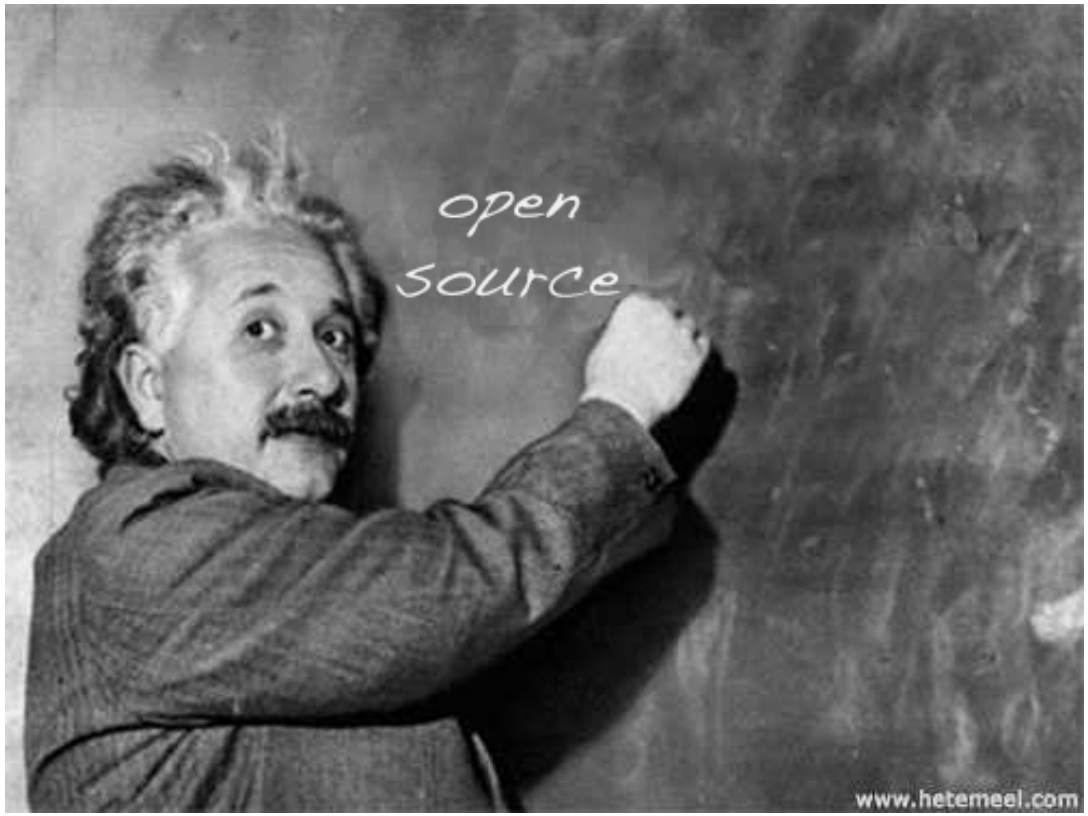
Contributing Documentation

- Help write good documentation
- Translate the documentation (and program text) into another language
- Read existing documentation, follow the examples and make corrections
- Create diagrams, screen-shots, and graphics for documentation
- Develop spelling and grammar style conventions for documentors
- Build a glossary of technical terms
- Convert documentation into more useful formats.



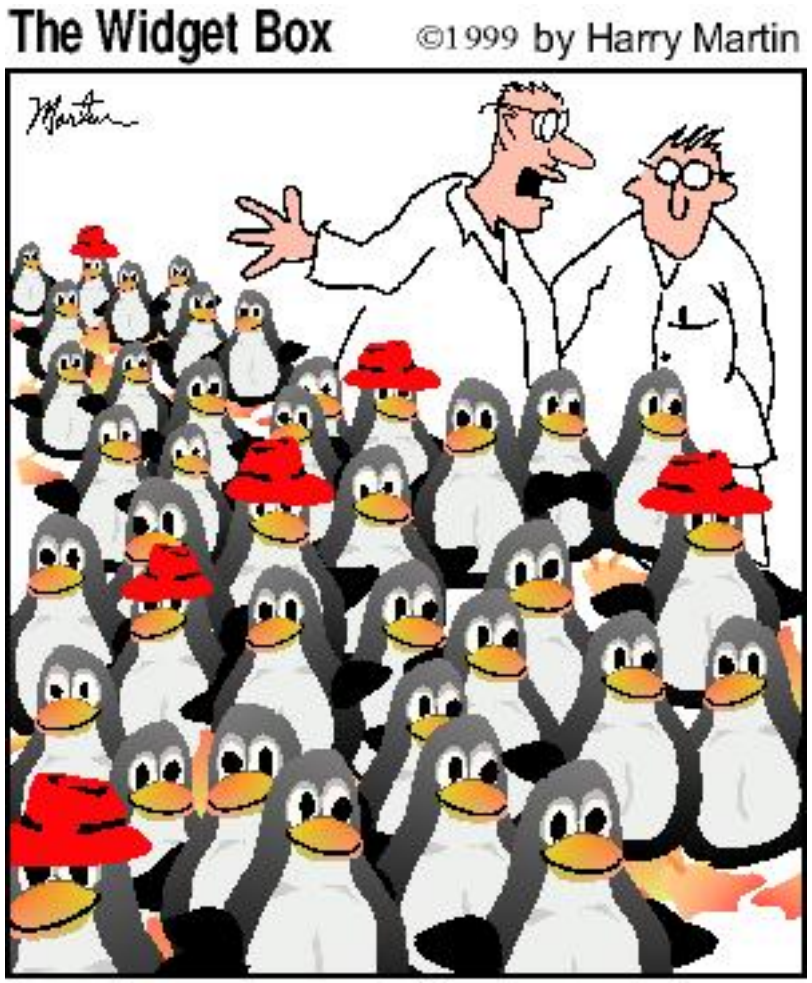
Contributing Support

- Answer questions on forums, mailing lists or IRC channels
- Contribute to (or start) an online support group
- Help other people learn how to use the program (or programming library)
- Maintain a FAQ or HOWTO document
- Buy a Free Software product, or associated products
- Buy products from companies that Support Free Software



Contributing Publicity

- Package the application for a particular Linux distribution (or other OS)
- Convince people to chose Open Source products when possible
- Write articles, reviews, critiques and books
- Write about new ways of using an Open Source program
- Write up case studies of successful Open Source implementations
- Help organize LUG events, including InstallFests, BugFests, and DocFests
- Provide training to new Linux users



"There's no telling how many Linux users are out there...they're breeding faster than rabbits."

Examples: The Fedora Project

Content writer
People person
Translator
OS developer
Designer
Web developer and administrator

Examples: OLPC

Activities

Epistemological impact

Fun

Quality

Sugarized

FOSS

Extensible

Uniqueness

Expectations

Discoverable

Core systems support

Question and Answers