Open Source Software *for fun and profit*

Russell Doty
Red Hat

disclaimer

- This presentation does not constitute advice or legal advice.
- It represents my personal observations and insights.
  - I’m not speaking for Red Hat.
- If you have questions, seek legal advice.
- If you are tempted to follow my suggestions blindly, seek professional help!
- The goal is to help you get started in your own personal journey with Open Source software.

What's in it for me? - Fun

- Work on something interesting
- Do the parts you want to do
- Access to the entire project
- Learn! Tools, technologies, disciplines, skills
- Control your own destiny
- Contribute to something bigger than you
- Make a difference
- Associate with some of the brightest people on the planet

What’s in it for me? - Profit

- *“My resume is on Google”*
  - This is the code I’ve written
  - These are the things I’ve designed
  - These are the problems I’ve solved
  - These are the people I’ve helped
  - These are the complex systems I’ve contributed to
  - This is where I’ve done the entire software lifecycle
  - This is how I interact with people
  - It’s public. It’s good. It’s me.

What is Open Source?

Open Source

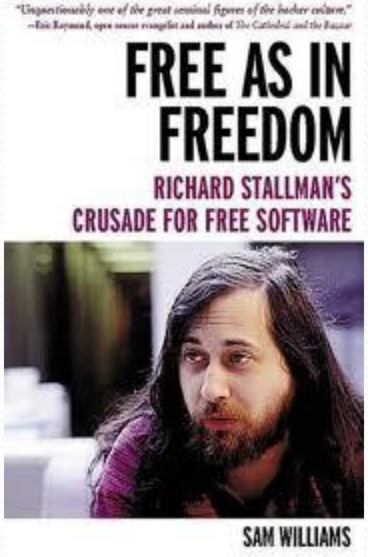
- Source code available
- Can be used for any purpose
- Can be modified
- Can be freely redistributed
- Under an Open Source license
    - **Not** public domain

Proprietary

- Generally binary only
    - Source code restricted
- Controlled by owner
    - Only owner can make changes
    - Owner can set terms of use
- Generally can’t be redistributed
- Generally can’t be modified

Richard Stallman: Four Freedoms for Free Software

- Freedom to run the program for any purpose
- Freedom to study how the program works and adapt it to your needs
- Freedom to redistribute copies so you can help your neighbor


"Unquestionably one of the great seminal figures of the hacker culture."
—Eric Raymond, open source evangelist and author of The Cathedral and the Bazaar
FREE AS IN FREEDOM
RICHARD STALLMAN'S CRUSADE FOR FREE SOFTWARE
SAM WILLIAMS

- Freedom to improve the program and release your modifications to the public

Open Source License Differences

Changes Returned

- All changes must be returned to the community as a pre-requisite for redistribution
- Example: GPL

No Changes Returned

- Changes do not have to be returned to the community as a pre-requisite for redistribution
- Example: BSD

Note: seek legal advice if you have any questions on licensing

Changing/Improving Software

Proprietary

- The owner can change it
  - Only the owner
  - Can change it in any way at any time
  - Can refuse to change it!
  - Can discontinue it
  - No recourse

Open Source

- The upstream maintainer can change it
- Distributions can change it
- You can change it
- You can hire or convince someone to change it
- Anyone can change it!
- **AND**
  - No one is forced to accept changes

Rule #1 for Open Source: *No one has to accept your changes*

- You have the right to make changes
  - You have the obligation to make the changes available under certain conditions
- You do not have the power to **make** anyone *accept* your changes
  - No matter how important it is to you
- You have to *persuade* people to accept your changes
  - This acceptance based on influence and participation, not command and control

How does Open Source work?

- Projects
- Communities
- Maintainers, Contributors, and Participants
- Upstream/Downstream/Users

Open Source Projects

- *Projects* are the building blocks of Open Source
  - A unit of software that can be installed to perform a task or function
  - Examples: Evolution, The GIMP, Libre Office, Linux kernel

LibreOffice
The Document Foundation

- Three Legs of an Open Source Project
  - Source Code Control repository
  - Web Site
  - Mailing List & irc channel

Open Source Community

- *Community* – the group of people supporting a project
  - Maintainers
  - Contributors
    - Code
    - Test Cases
    - Documentation
    - Packaging
    - Testing
    - Community Work (web site, community voice, etc.)
  - Users
    - Provide use support, bug reports,

Community Participation is Voluntary

Streams

- *Upstream*
  - The person, organization, or group which maintains & enhances a project. Final say on what is included in the official project.
- *Downstream*
  - An organization or mechanism for integrating and delivering applications and modules
    - Debian, Red Hat, SuSE, Canonical, etc.
- Users
  - People or organizations consuming software

Roles

- Maintainer or Upstream Maintainer
  - Commit access to repository
  - Final say about what goes into the official project
- Committer
  - Commit access to repository
- Contributor
  - Contributes – code, documentation, testing, support, etc.
- Supporter
  - Engaged but doesn’t directly contribute
- Consumer or User
  - Uses the software without contributing

"Push it Upstream"

- The ***process*** of getting changes ***accepted*** into a project
- Push it Upstream is not:
  - Slap an open source license on some code
  - Throw it up on SourceForge
  - Demand that a project accept it

We will tell you how to get changes accepted after covering how to make them

Mechanics of Open Source

Components

- Web sites
- Source Code Repositories
- Mailing Lists
- Patch Submission
- Project Specific Procedures & Processes

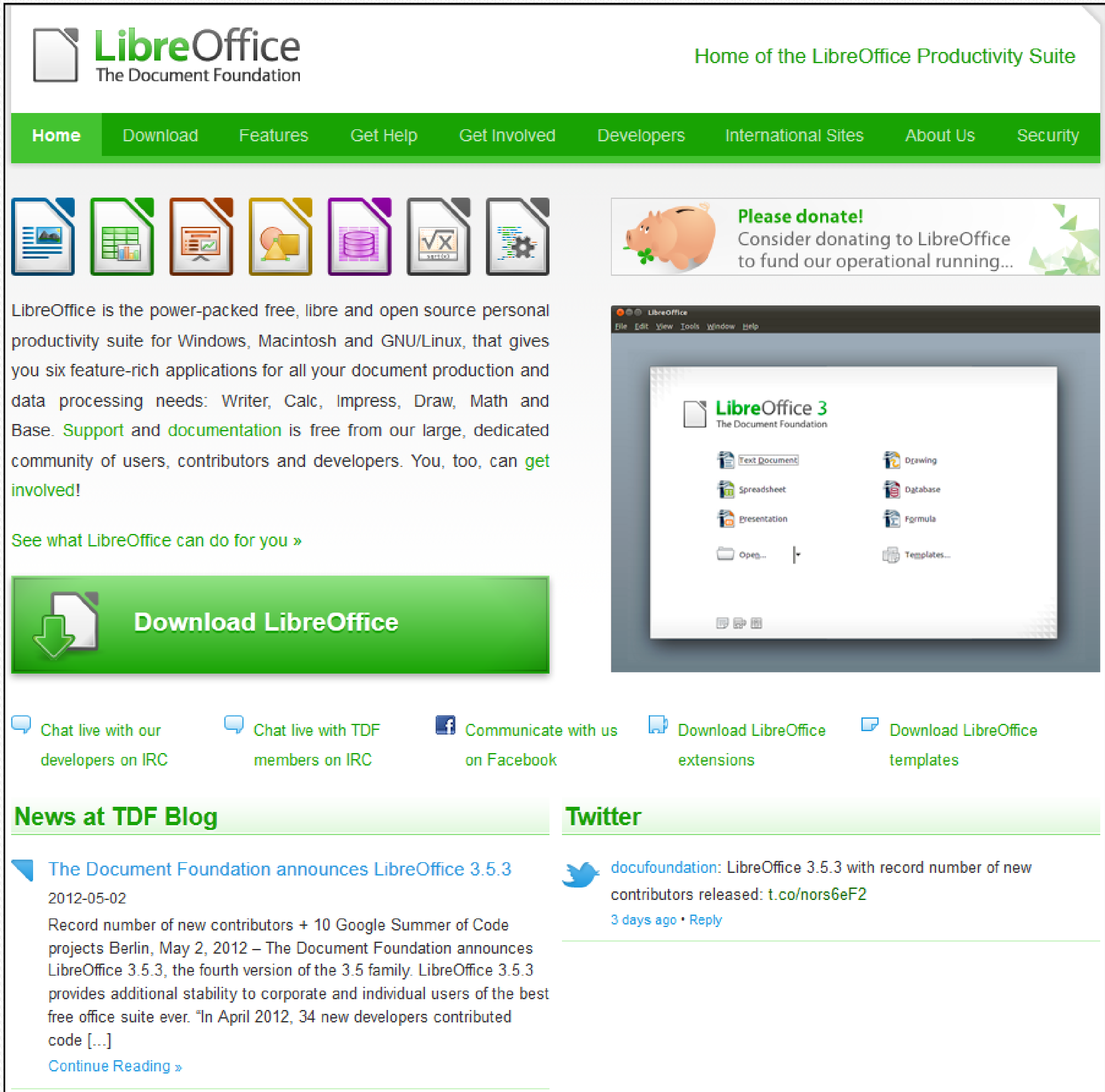
Process

- *Find out how the Project works*
- Develop Code
- Release under proper license
- Push the code upstream
- Maintain the code

Web Sites

Core of Project

- Overview
- Features
- Download
- Documentation
- Plans/Schedule
- Bug Reporting



Home of the LibreOffice Productivity Suite

Please donate!
Consider donating to LibreOffice to fund our operational running...

LibreOffice is the power-packed free, libre and open source personal productivity suite for Windows, Macintosh and GNU/Linux, that gives you six feature-rich applications for all your document production and data processing needs: Writer, Calc, Impress, Draw, Math and Base. Support and documentation is free from our large, dedicated community of users, contributors and developers. You, too, can get involved!

See what LibreOffice can do for you »

Chat live with our developers on IRC

Chat live with TDF members on IRC

Communicate with us on Facebook

Download LibreOffice extensions

Download LibreOffice templates

News at TDF Blog

The Document Foundation announces LibreOffice 3.5.3

2012-05-02

Record number of new contributors + 10 Google Summer of Code projects Berlin, May 2, 2012 – The Document Foundation announces LibreOffice 3.5.3, the fourth version of the 3.5 family. LibreOffice 3.5.3 provides additional stability to corporate and individual users of the best free office suite ever. "In April 2012, 34 new developers contributed code [...]

Continue Reading »

Twitter

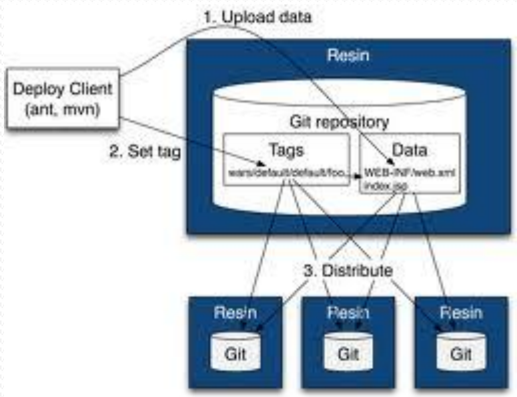
docufoundation: LibreOffice 3.5.3 with record number of new contributors released: t.co/nors6eF2

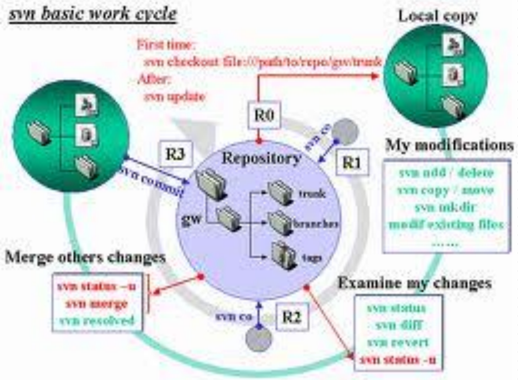
3 days ago • Reply

Source Code Repositories

Git, SVN, CVS, etc.

- Source Code
- History of Changes
- Versions
- Configuration
- Checkin/Checkout
- Branching
- Merging
- Access Control





You need to learn to use the source code control system

Mailing Lists

Primary Vehicle for Work

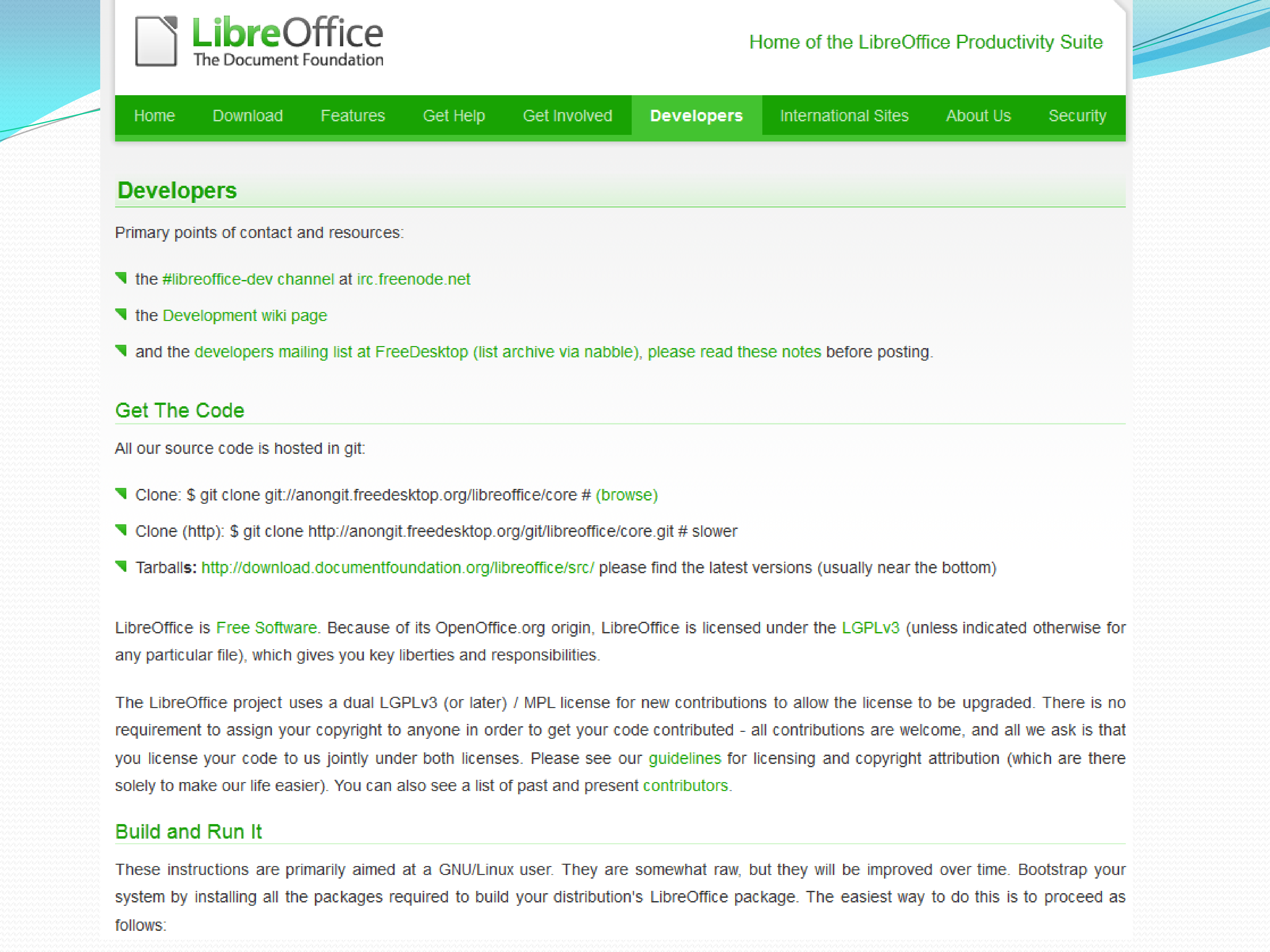
- Questions
- Discussions
- Proposed Changes
- Feedback
- Patches
- ACK/NAK
- Archives
- Flame wars…

Using the Mail List

- Join the project mail list
- Search the Archives
- Follow the project mail list
- Learn protocol, conventions & personalities of the list
- Contribute to the list
- Ask questions on the list
- Review patches
- Submit patches

Project Processes & Guidelines

- Learn how ***this*** project works
  - Project Web site is a great place to start
  - Differs from project to project
- Follow the process
  - Even if you don’t like it
- Let’s look at LibreOffice as an example:



Developers

Primary points of contact and resources:

- the #libreoffice-dev channel at irc.freenode.net
- the Development wiki page
- and the developers mailing list at FreeDesktop (list archive via nabble), please read these notes before posting.

Get The Code

All our source code is hosted in git:

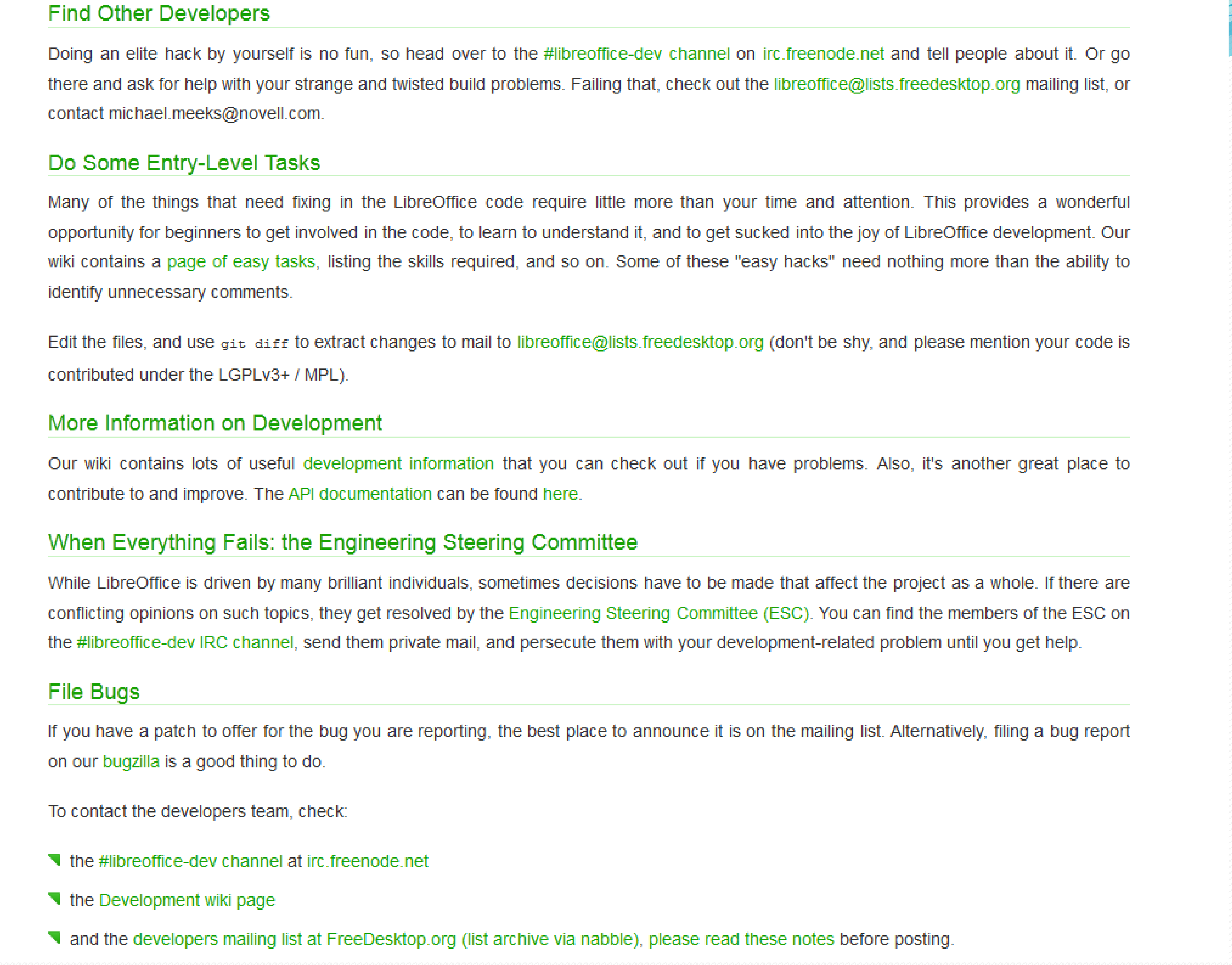
- Clone: $ git clone git://anongit.freedesktop.org/libreoffice/core # (browse)
- Clone (http): $ git clone http://anongit.freedesktop.org/git/libreoffice/core.git # slower
- Tarball**s:** http://download.documentfoundation.org/libreoffice/src/ please find the latest versions (usually near the bottom)

LibreOffice is Free Software. Because of its OpenOffice.org origin, LibreOffice is licensed under the LGPLv3 (unless indicated otherwise for any particular file), which gives you key liberties and responsibilities.

The LibreOffice project uses a dual LGPLv3 (or later) / MPL license for new contributions to allow the license to be upgraded. There is no requirement to assign your copyright to anyone in order to get your code contributed - all contributions are welcome, and all we ask is that you license your code to us jointly under both licenses. Please see our guidelines for licensing and copyright attribution (which are there solely to make our life easier). You can also see a list of past and present contributors.

Build and Run It

These instructions are primarily aimed at a GNU/Linux user. They are somewhat raw, but they will be improved over time. Bootstrap your system by installing all the packages required to build your distribution's LibreOffice package. The easiest way to do this is to proceed as follows:



Find Other Developers

Doing an elite hack by yourself is no fun, so head over to the #libreoffice-dev channel on irc.freenode.net and tell people about it. Or go there and ask for help with your strange and twisted build problems. Failing that, check out the libreoffice@lists.freedesktop.org mailing list, or contact michael.meeks@novell.com.

Do Some Entry-Level Tasks

Many of the things that need fixing in the LibreOffice code require little more than your time and attention. This provides a wonderful opportunity for beginners to get involved in the code, to learn to understand it, and to get sucked into the joy of LibreOffice development. Our wiki contains a page of easy tasks, listing the skills required, and so on. Some of these "easy hacks" need nothing more than the ability to identify unnecessary comments.

Edit the files, and use `git diff` to extract changes to mail to libreoffice@lists.freedesktop.org (don't be shy, and please mention your code is contributed under the LGPLv3+ / MPL).

More Information on Development

Our wiki contains lots of useful development information that you can check out if you have problems. Also, it's another great place to contribute to and improve. The API documentation can be found here.

When Everything Fails: the Engineering Steering Committee

While LibreOffice is driven by many brilliant individuals, sometimes decisions have to be made that affect the project as a whole. If there are conflicting opinions on such topics, they get resolved by the Engineering Steering Committee (ESC). You can find the members of the ESC on the #libreoffice-dev IRC channel, send them private mail, and persecute them with your development-related problem until you get help.

File Bugs

If you have a patch to offer for the bug you are reporting, the best place to announce it is on the mailing list. Alternatively, filing a bug report on our bugzilla is a good thing to do.

To contact the developers team, check:

- the #libreoffice-dev channel at irc.freenode.net
- the Development wiki page
- and the developers mailing list at FreeDesktop.org (list archive via nabble), please read these notes before posting.

Developing Code

- Not going to say much – you should know what you are doing here
- Follow the guidelines of the project
    - Language
    - Formatting (white space & tabs matter)
    - Conventions, variable naming, etc.
    - Test cases
    - Testing
    - Base on latest upstream version
    - Release early / Release often!

Packaging Code

- Diffs
- Small, self contained
    - Reviewer should understand in 5-10 minutes
- Bi-sectable
- Applies cleanly
- Include Explanation
- In-line in email (not attached)
- Goal is to *obtain review and feedback*
- **Make it easy for reviewer – or they will move on!**

Gaining Upstream Acceptance

- Submit patchsets on mailing list
- Follow up quickly on questions and feedback
- Make suggested changes
  - Unless compelling reason not to
- Need to get ACKs
  - And no NAKs
- Stay positive and proactive
- Remember: **you have to justify acceptance** – *upstream doesn't have to justify rejection*
- Play by the rules or be ignored

Gaining Upstream Acceptance

- Remember: **you have to justify acceptance** – *upstream doesn’t have to justify rejection*

Notes on Feedback

- There is a stereotype that Open Source is a bunch of Alpha Geeks with Asperger's Syndrome and no social skills
  - There is a kernel of truth to this…
- **Consider:**
  - Debate is good – for learning and making progress
  - Directness or Bluntness is considered a virtue
  - Many maintainers are pressed for time
  - Maintainers reluctant to invest time in people unless they believe there will be benefit and return
  - Reputation and Influence are **earned**

More on Feedback

- Don't take it personally
- Feedback is against your *code* – not against *you*
- You get feedback because someone *cared enough to spend the time*; listen to what they have to say
- Much of the debate is valuable and you learn a lot
- **However:**
  - There are jerks out there
  - If a project is dominated by jerks, find another project
    - Life is too short to waste dealing with jerks
  - Don't be too sensitive / Don't put up with too much crap

Feedback…

NAK

Code poorly structured. Needs indenting and white space. Variable names not meaningful. Fails at 2pi; needs input validation. Needs better and more robust error handling.

Consider using Heimdall-Riemann for better performance.

Feedback 2

Who do you think you are, wasting our time with this drivel. This code is such crap you must be a total moron. Get off our mailing list loser and climb back under that rock in your mother's basement.

Feedback 3

ACK.

Committed to repository, commit as34gcd878ddlefy.

Please register yourself in the maintainers file.

Finding a Project/Community

- What are you interested in?
- What projects are in this area?
- How active is the project?
- Who are the key players?
- What is the interaction style?
- What is the general feel of the project?
- Is it something you think you can contribute to?
  - Note: expertise is something you *build*, not something you have to *start with*.

Joining a Project/Community

- Read the Web site/follow the mail list & irc
- Check out the code and get familiar with it
- Look for starter tasks:
  - Bug triage
  - Simple bug fixes
  - Code cleanup/refactoring
  - Documentation
  - Test cases
- Review patches on the mailing list

How to Succeed/How to Fail

Succeed

- Join the community & Contribute
- Offer changes of value
- Package Patches for Review
- Respond to Feedback
- Make suggested changes
- Work with the community
- RTFM
- Listen!

Fail

- Doesn't match project
- Patch Bombs
- Code doesn't build
- Code doesn't work
- Don't listen to reviewers
- Approach community with an attitude
- Demand your changes be accepted

Your Code is Accepted! Now What?

- Celebrate!
- Let the world know
- Look for feedback from people using it
    - Praise – bask in it!
    - Bugs – fix ‘em
    - Valid RFEs – consider them
    - Whining – ignore
- Maintain
    - You brought it into the world; your responsibility
- Enhance! What are you going to do next?

Go Forth and Benefit!

- Are you having fun?
  - Yes – keep going!
  - No – re-evaluate
- Build your reputation
- Build your online presence
- Look for more opportunities to participate
  - Gatherings, hackathons, meetups
- Mercenary? Look for ways to monetize
- Mentor
- Be ready for *incoming opportunities*

What are you waiting for?