Free Software and Debian
20 years later... and the next 10

Roberto Di Cosmo
roberto@dicosmo.org

Université Paris Diderot et INRIA

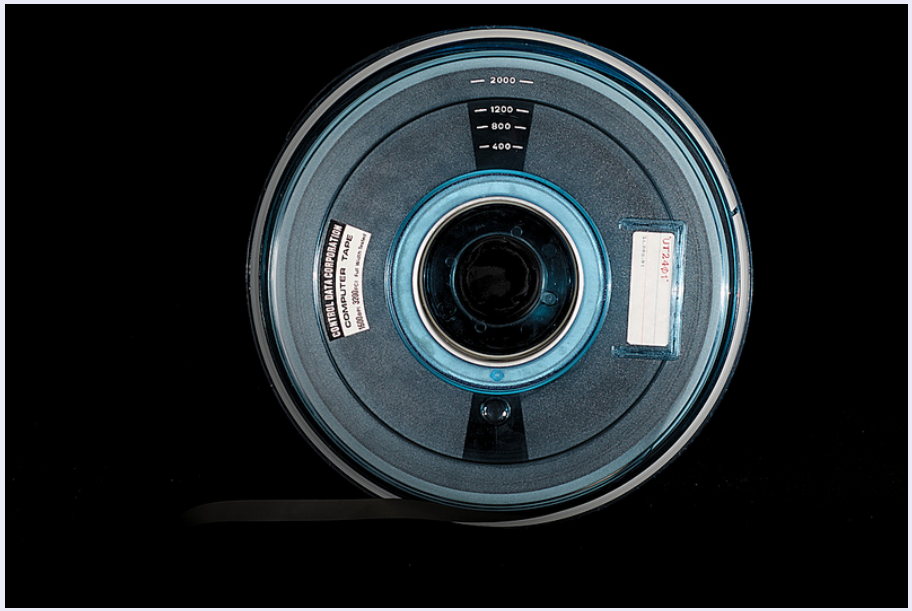
24 November 2012
Mini Debconf
Paris

Outline

The 80's

Code distribution

Slow ! ☹



The 80's

Code documentation

Good ! Literate programming was popularised by Donald E. Knuth.

See this excerpt from the original diff code (file `io.c`)

```
/* Discard lines from one file that have no matches in the other file.

   A line which is discarded will not be considered by the actual
   comparison algorithm; it will be as if that line were not in the file.
   The file's 'realindexes' table maps virtual line numbers
   (which don't count the discarded lines) into real line numbers;
   this is how the actual comparison algorithm produces results
   that are comprehensible when the discarded lines are counted.

   When we discard a line, we also mark it as a deletion or insertion
   so that it will be printed in the output.  */

static void
discard_confusing_lines (struct file_data filevec[])
```

We had a lot of time to kill waiting for the tape to arrive ☺

The 80's

Collaboration

Some : a lot of time available, but spent working with lousy tools

- Ethernet, 10Mb/s.
- NFS mounted partitions
- RCS ☹☹
- CVS ☹

All this enabled local collaboration, but non-local collaboration was based on IRC, e-mail, ftp, bulletin boards, news...

The 80's

Philosophy

Emerging : GNU Manifesto, GPL (1988/1989)

GNU GENERAL PUBLIC LICENSE
Version 1, February 1989

Copyright (C) 1989 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.

Preamble

The license agreements of most software companies try to keep users at the mercy of those companies. By contrast, our General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. The General Public License applies to the Free Software Foundation's software and to any other program whose authors commit to using it. You can use it for your programs, too.

The 80's

Visibility

Marginal ☹

The 90's

Free software projects

Growing ! ☺

- Linux
- Gnome, Kde
- OpenOffice
- Mozilla
- ...

The 90's

Code distribution

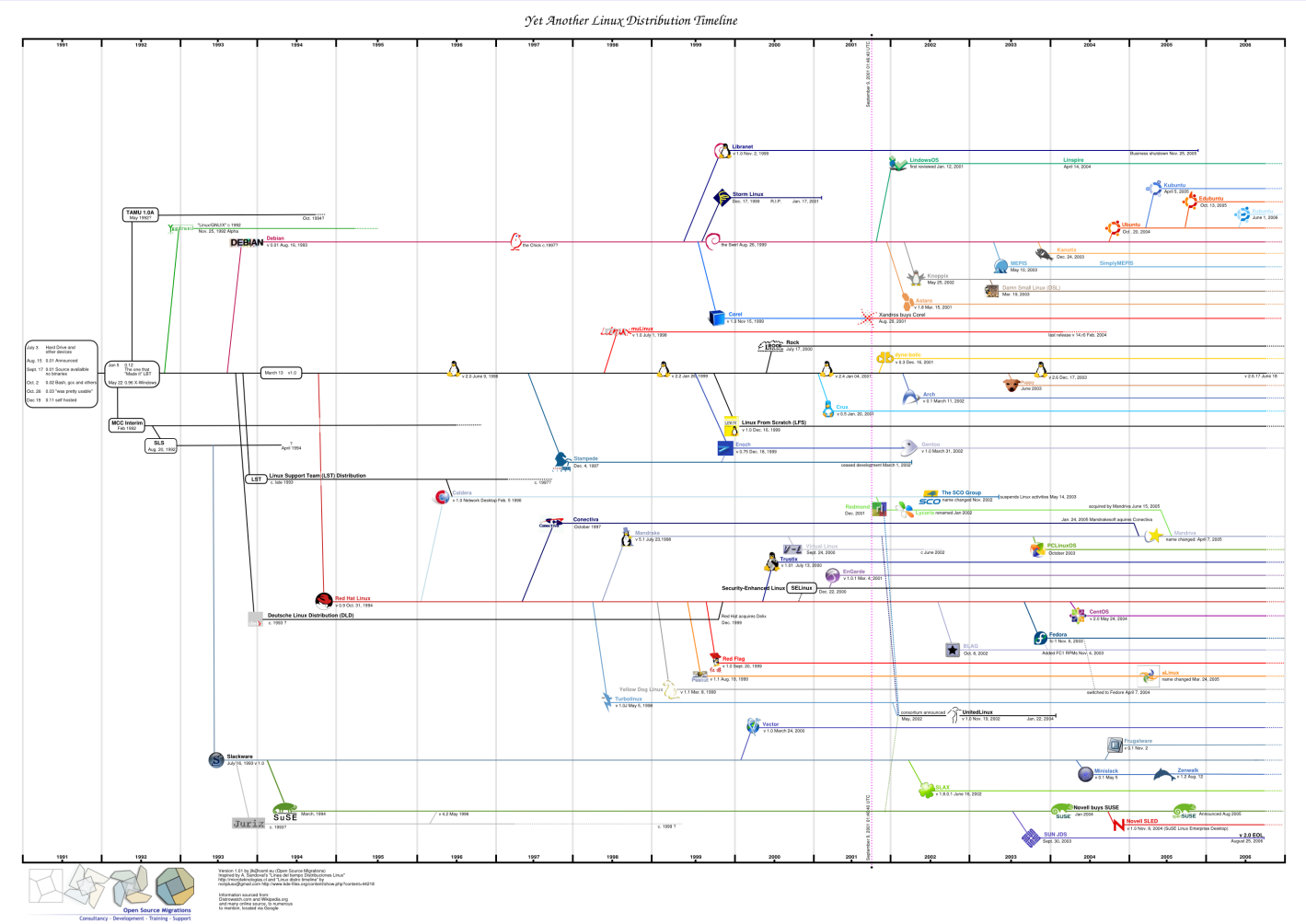
Getting (slowly) up to speed ! ☺

- World Wide Web
- Internet access from home (1995 is a turning point)
- ADSL arrives (1Mb/s at home)

But installing software becomes hard... we needed help !

The 90's

Distributions, a very successful idea !



The 90's

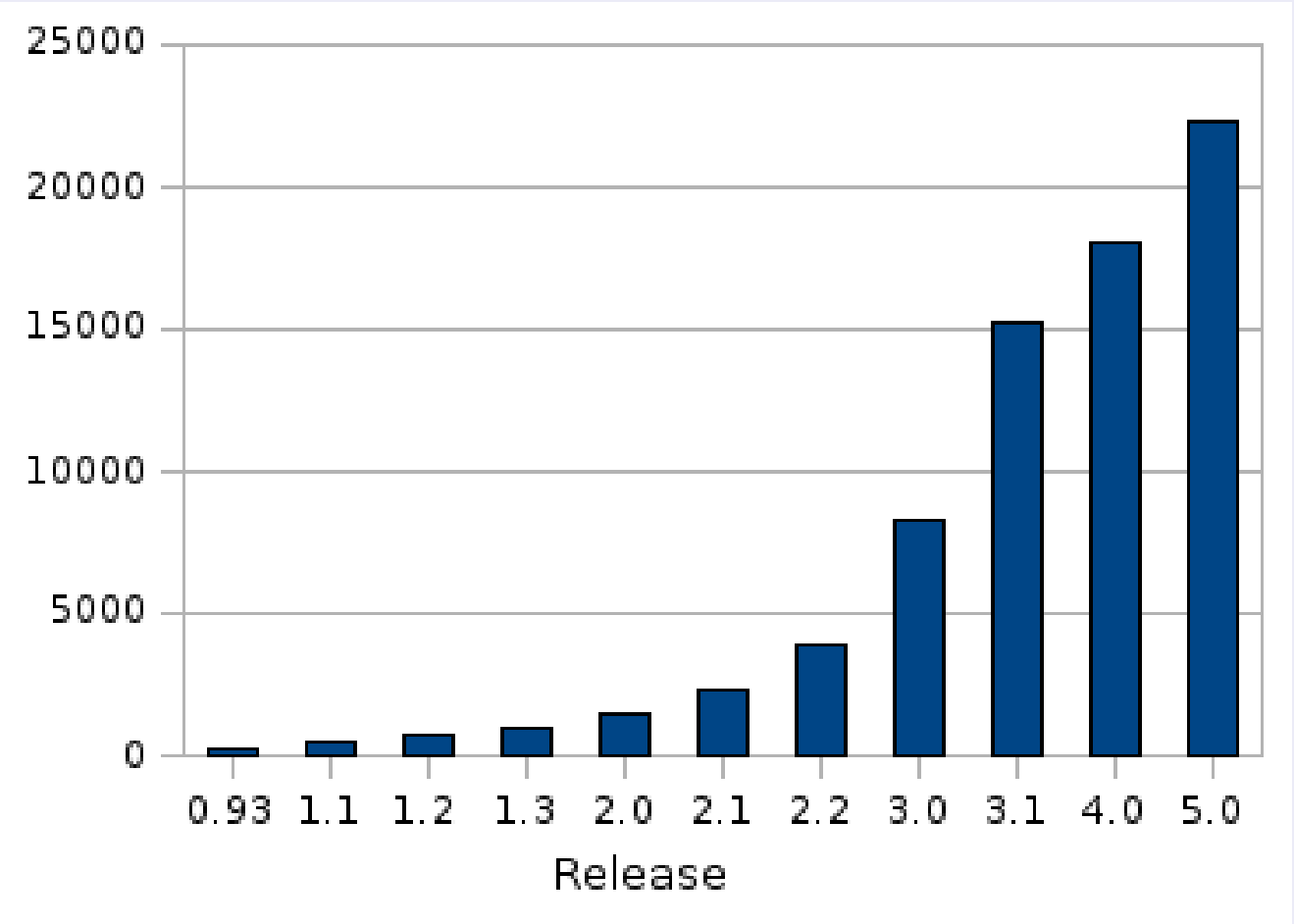
Visibility

Good ☺

The 00's

Free software projects

Exponential growth



The 00's

Visibility

Business notices ! Free software everywhere ...☺
But too often, under the hood ...

- iPhone
- ADSL "boxes"
- all kind of devices (routers, etc.)

that's not always visible ☹

The 00's

Collaboration

- software forges (SourceForge, Savannah)
- distributed VCS (Mercurial, Bazaar, Git)
- ...

The 00's

So, everything is fine ?

After all...

- fast network
- a lot of free software
- free software everywhere
- nice tools
- social frenzy
- free software economy (2.5Billions FY 2011 in France)
- ...

Outline

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Newcomers with different goals

The very success of free software may be its greatest danger : we need to bring up to speed a wealth of newcomers that do not necessarily share (yet) the same culture

- code drop, open-core, freemium, etc. : innovations we may have lived without
- "social coding" frenzy : very appealing marketing, with a huge entropy risk
- "who needs comments", "just copy/paste it", etc. moods : bad habits creeping in
- "who cares if package P disappears" mood : need to cater for all users, even non developers
- ...

Mentoring the new generation is now more important than ever. Telling our story to the newcomers is essential.

The landscape has changed

Dematerialization of the personal computer and the data center

- cloud environments may generalize the use of free software as a commodity
- TPM-enabled hardware may render free software unusable
- BYOC, BYOD, and remote hyperconnected virtualization may render free software irrelevant on our devices
- cloudified storage puts our privacy at a risk

We need to rise our standards : free software and open standards are no longer enough

The landscape has changed

Dematerialization of the personal computer and the data center

- cloud environments may generalize the use of free software as a commodity
- TPM-enabled hardware may render free software unusable
- BYOC, BYOD, and remote hyperconnected virtualization may render free software irrelevant on our devices
- cloudified storage puts our privacy at a risk

We need to rise our standards : free software and open standards are no longer enough

The landscape has changed

Dematerialization of the personal computer and the data center

- cloud environments may generalize the use of free software as a commodity
- TPM-enabled hardware may render free software unusable
- BYOC, BYOD, and remote hyperconnected virtualization may render free software irrelevant on our devices
- cloudified storage puts our privacy at a risk

We need to rise our standards : free software and open standards are no longer enough

The landscape has changed

Dematerialization of the personal computer and the data center

- cloud environments may generalize the use of free software as a commodity
- TPM-enabled hardware may render free software unusable
- BYOC, BYOD, and remote hyperconnected virtualization may render free software irrelevant on our devices
- cloudified storage puts our privacy at a risk

We need to rise our standards : free software and open standards are no longer enough

The landscape has changed

Dematerialization of the personal computer and the data center

- cloud environments may generalize the use of free software as a commodity
- TPM-enabled hardware may render free software unusable
- BYOC, BYOD, and remote hyperconnected virtualization may render free software irrelevant on our devices
- cloudified storage puts our privacy at a risk

We need to rise our standards : free software and open standards are no longer enough

The landscape has changed

Free software is everywhere !

In particular embedded : quality concerns are on the rise

- aerospace (of course)
- automotive (it's coming !)
- health care (a clear need)
- ... you name it

We need to rise our standards : access to the code is no longer enough

The landscape has changed

There is a huge amount of free software !

This growing complexity poses huge challenges, for quality, maintenance, documentation, organization, governance, user participation.
Our concerns

- the software component
- the collection of components
- the process
- the licence
- the collaboration
- ...

We need to rise our standards : guidelines are no longer enough, we need new tools

Outline

Let's work together

Researchers look for interesting problems

Let's convince them to look at ours!
See `coccinelle.lip6.fr`, `mancoosi.debian.net`, `coinst.irill.org` for example of what this collaboration can produce.

Educators and students look for new ways of learning

Let's give them an occasion to participate.

Modularity is the key

Back to the basics

To cope with complexity in Computer Science, we use abstraction and modularity. This does not apply only to software !

We need to apply the same principle to community, software, economy, legal framework and organization.

Conclusions

Debian matters now more than ever

A living example of a community collaborating on a common goal, with a strong quality focus on all aspects : technical, legal and social.

Now it's time to go to the next level.