Free Software

Philosophy, history and practice

Luca Saiu
positron@gnu.org
http://ageinghacker.net

GNU Project

Screencast version
Paris, October 2014

Introducing myself

I'm a computer scientist living and working somewhere around Paris...

...and a *GNU* maintainer.

I'm also an associate member of the *Free Software Foundation*, a fellow of the *Free Software Foundation Europe* and an *April* adherent.

So I'm **not** an impartial observer.

Contents

1. Introduction (and some quick reminders)
2. History and philosophy
   - The hacker community
   - The GNU Project and the Free Software movement
   - Linux and the Open Source movement
3. Legal aspects
   - Copyright
   - Free Software licenses

Reminders about software — source code vs. machine code

Source code vs. machine code

Quick demo

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

- Programs *are linked to* libraries
  - static libraries
  - shared libraries

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

- Programs *are linked to* libraries
  - static libraries
  - shared libraries
- Libraries (or programs) request services to the *kernel*

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

- Programs *are linked to* libraries
  - static libraries
  - shared libraries
- Libraries (or programs) request services to the *kernel*
- Programs *invoke* with other programs

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

- Programs *are linked to* libraries
    - static libraries
    - shared libraries
- Libraries (or programs) request services to the *kernel*
- Programs *invoke* with other programs
- Programs *communicate* with other programs...
    - ...on the same machine
    - ...over the network

Reminders about software — linking

In practice, programs don't exist in isolation.

Remember `printf()`? It's a library function.

- Programs *are linked to* libraries
  - static libraries
  - shared libraries
- Libraries (or programs) request services to the *kernel*
- Programs *invoke* with other programs
- Programs *communicate* with other programs...
  - ...on the same machine
  - ...over the network

We're gonna see that this has legal implications.

History of the Free Software movement — hackers

This story begins at the end of the 1970s at the *Artificial Intelligence Lab* of the Massachusetts Institute of Technology...

It was a community of *hackers*.

- By the way, the word "hacker" has been misused by the media: it does **not** imply breaking over network security. Hacking is "*playful cleverness*" [RMS]:
    - Finding unusual, creative solutions to computing problems
    - *"Hacker: A person who enjoys exploring the details of programmable systems and stretching their capabilities, as opposed to most users, who prefer to learn only the minimum necessary."* [the Jargon file]
    - An intriguing culture and mentality, not limited to computing. See for example [2]

The MIT Artificial Intelligence lab in the 1960-1970s

A friendly, informal community...

- ...even external visitors were allowed to work on the system
- *No passwords* for a long time
- Software was normally shared, *in source form*.

The MIT Artificial Intelligence lab in the 1960-1970s

A friendly, informal community...

- ...even external visitors were allowed to work on the system
- *No passwords* for a long time
- Software was normally shared, *in source form*.
    - No copyright notices
    - Anybody was expected to be able to make copies
    - It was just the way it was. *It was normal.*

The MIT Artificial Intelligence lab in the 1960-1970s

A friendly, informal community...

- ...even external visitors were allowed to work on the system
- *No passwords* for a long time
- Software was normally shared, *in source form*.
  - No copyright notices
  - Anybody was expected to be able to make copies
  - It was just the way it was. *It was normal.*

Lots of now-famous people did important work there:

- Marvin Minsky, John McCarthy (for a while), Harold Abelson, Gerald Sussman, Guy Steele...

The MIT Artificial Intelligence lab in the 1960-1970s

A friendly, informal community...

- ...even external visitors were allowed to work on the system
- *No passwords* for a long time
- Software was normally shared, *in source form*.
  - No copyright notices
  - Anybody was expected to be able to make copies
  - It was just the way it was. *It was normal.*

Lots of now-famous people did important work there:

- Marvin Minsky, John McCarthy (for a while), Harold Abelson, Gerald Sussman, Guy Steele...

Richard Stallman was one of the hackers hired to work on the operating system.

The MIT AI lab in the 1960-1970s — hardware and software

The computing environment at the AI lab:

- DEC *PDP-10* computers; pretty powerful machines for the time: 36 bit (yes, *36*), about 1MB RAM

Figure: a PDP-10 computer

The MIT AI lab in the 1960-1970s — hardware and software

The computing environment at the AI lab:

- DEC *PDP-10* computers; pretty powerful machines for the time: 36 bit (yes, *36*), about 1MB RAM

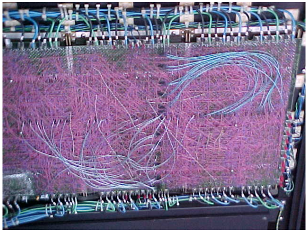


Figure: PDP-10 backplane

The MIT AI lab in the 1960-1970s — hardware and software

The computing environment at the AI lab:

- DEC *PDP-10* computers; pretty powerful machines for the time: 36 bit (yes, *36*), about 1MB RAM
- *ITS* operating system, written in assembly *by the lab hackers themselves*: multi-task, multi-user

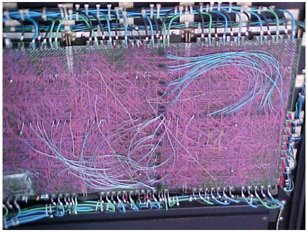


Figure: PDP-10 backplane

The MIT AI lab in the 1960-1970s — hardware and software

The computing environment at the AI lab:

- DEC *PDP-10* computers; pretty powerful machines for the time: 36 bit (yes, *36*), about 1MB RAM
- *ITS* operating system, written in assembly *by the lab hackers themselves*: multi-task, multi-user
- Much user software written *in Lisp*

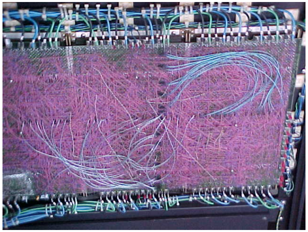


Figure: PDP-10 backplane

The MIT AI lab in the 1960-1970s — hardware and software

The computing environment at the AI lab:

- DEC *PDP-10* computers; pretty powerful machines for the time: 36 bit (yes, *36*), about 1MB RAM
- *ITS* operating system, written in assembly *by the lab hackers themselves*: multi-task, multi-user
- Much user software written *in Lisp*
  - SHRDLU
  - Macsyma
  - Scheme
  - ...

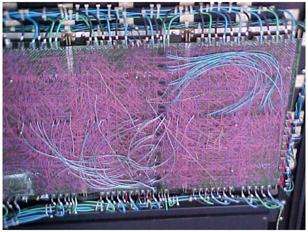


Figure: PDP-10 backplane

Stallman encounters Non-Disclosure Agreements

End of the 1970s: Xerox donates one of the first laser printers to the lab...

- the driver is proprietary, and has problems. Stallman tries to get a copy of the source to fix it...

Stallman encounters Non-Disclosure Agreements

End of the 1970s: Xerox donates one of the first laser printers to the lab...

- the driver is proprietary, and has problems. Stallman tries to get a copy of the source to fix it...
- the answer he gets is "I've promised *not* to share the source".

Stallman encounters Non-Disclosure Agreements

End of the 1970s: Xerox donates one of the first laser printers to the lab...

- the driver is proprietary, and has problems. Stallman tries to get a copy of the source to fix it...
- the answer he gets is "I've promised *not* to share the source".
- Stallman feels the community is being harmed and is very enraged.

Stallman encounters Non-Disclosure Agreements

End of the 1970s: Xerox donates one of the first laser printers to the lab...

- the driver is proprietary, and has problems. Stallman tries to get a copy of the source to fix it...
- the answer he gets is "I've promised *not* to share the source".
- Stallman feels the community is being harmed and is very enraged.

(the driver problem was never solved)

Lisp machines

Lisp is powerful, but hard to efficiently implement. It felt “slow” on the machines of the time.

The idea: make *special computers* with hardware support for Lisp

Lisp machines

Lisp is powerful, but hard to efficiently implement. It felt "slow" on the machines of the time.

The idea: make *special computers* with hardware support for Lisp

- the *Lisp Machine* was designed at MIT by Tom Knight and Richard Greenblatt, two lab hackers

Figure: a Lisp Machine

Lisp machines

Lisp is powerful, but hard to efficiently implement. It felt "slow" on the machines of the time.

The idea: make *special computers* with hardware support for Lisp

- the *Lisp Machine* was designed at MIT by Tom Knight and Richard Greenblatt, two lab hackers
- *two* spinoffs to build and sell the machines

Figure: a Lisp Machine

Lisp machines

Lisp is powerful, but hard to efficiently implement. It felt "slow" on the machines of the time.

The idea: make *special computers* with hardware support for Lisp

- the *Lisp Machine* was designed at MIT by Tom Knight and Richard Greenblatt, two lab hackers
- *two* spinoffs to build and sell the machines
  - Lisp Machines Incorporated (Greenblatt)
  - Symbolics (Noftsker)

Figure: a Lisp Machine

Early 1980s: the community crumbles — LMI vs. Symbolics

- LMI: sell hardware, and contribute the software back to MIT for everybody to use

Early 1980s: the community crumbles — LMI vs. Symbolics

- LMI: sell hardware, and contribute the software back to MIT for everybody to use
- Symbolics: the new software is proprietary (can be used at MIT, but not redistributed)

Early 1980s: the community crumbles — LMI vs. Symbolics

- LMI: sell hardware, and contribute the software back to MIT for everybody to use
- Symbolics: the new software is proprietary (can be used at MIT, but not redistributed)
- many lab hackers leave MIT for Symbolics

Early 1980s: the community crumbles — LMI vs. Symbolics

- LMI: sell hardware, and contribute the software back to MIT for everybody to use
- Symbolics: the new software is proprietary (can be used at MIT, but not redistributed)
- many lab hackers leave MIT for Symbolics
- Richard Stallman feels betrayed: he stays at MIT, and tries to *independently re-implement* Symbolics's software modifications for everybody to share, alone for two years (1982-1983)

Figure: Richard Stallman (in 2008)

The PDP-10 becomes obsolete

DEC retires the PDP-10, which has become too limited.

- new machines (the PDP-11, then the VAX) are not compatible

The PDP-10 becomes obsolete

DEC retires the PDP-10, which has become too limited.

- new machines (the PDP-11, then the VAX) are not compatible
- the ITS operating system, written in assembly, becomes practically useless

The PDP-10 becomes obsolete

DEC retires the PDP-10, which has become too limited.

- new machines (the PDP-11, then the VAX) are not compatible
- the ITS operating system, written in assembly, becomes practically useless
- all the operating systems for the new machines are proprietary

The PDP-10 becomes obsolete

DEC retires the PDP-10, which has become too limited.

- new machines (the PDP-11, then the VAX) are not compatible
- the ITS operating system, written in assembly, becomes practically useless
- all the operating systems for the new machines are proprietary

It's the final blow to what remains of the lab community.

Birth of a new system

Stallman refuses to write proprietary software, and wants to rebuild a community like the MIT AI hackers.

A new operating system will be the first thing needed.

Birth of a new system

Stallman refuses to write proprietary software, and wants to rebuild a community like the MIT AI hackers.

A new operating system will be the first thing needed.

- available to anybody in source form

Birth of a new system

Stallman refuses to write proprietary software, and wants to rebuild a community like the MIT AI hackers.

A new operating system will be the first thing needed.

- available to anybody in source form
- portable

Birth of a new system

Stallman refuses to write proprietary software, and wants to rebuild a community like the MIT AI hackers.

A new operating system will be the first thing needed.

- available to anybody in source form
- portable
- compatible with Unix

Birth of a new system

Stallman refuses to write proprietary software, and wants to rebuild a community like the MIT AI hackers.

A new operating system will be the first thing needed.

- available to anybody in source form
- portable
- compatible with Unix
    - good for technical reasons, and to make it easily accepted
    - very different from ITS
    - *no political reason for this choice*; Unix was proprietary

1983: the GNU Project

Stallman decides to call the new system “GNU”:

GNU stands for “GNU’s Not Unix”: a *recursive acronym*, following a hacker naming tradition.

1983: the GNU Project

Stallman decides to call the new system “GNU”:

GNU stands for “GNU’s Not Unix”: a *recursive acronym*, following a hacker naming tradition.

- 1983, September 27: Stallman announces the birth of the project on `net.unix-wizards` and `net.usoft`, and calls for help.

1983: the GNU Project

Stallman decides to call the new system “GNU”:

GNU stands for “GNU’s Not Unix”: a *recursive acronym*, following a hacker naming tradition.

- 1983, September 27: Stallman announces the birth of the project on `net.unix-wizards` and `net.usoft`, and calls for help.
- some people help, but most think the project is impossible. Stallman simply ignores the naysayers and goes on.

1983: the GNU Project

Stallman decides to call the new system "GNU":

GNU stands for "GNU's Not Unix": a *recursive acronym*, following a hacker naming tradition.

- 1983, September 27: Stallman announces the birth of the project on `net.unix-wizards` and `net.usoft`, and calls for help.
- some people help, but most think the project is impossible. Stallman simply ignores the naysayers and goes on.
- He quits MIT and starts to work on GNU full-time. He sells copies of the already-written software and does consultancy work.

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention
- 1984: *GNU Emacs* (a powerful text editor scriptable in Lisp) is an immediate success

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention
- 1984: *GNU Emacs* (a powerful text editor scriptable in Lisp) is an immediate success
- 1987: the *GNU C Compiler* is another success

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention
- 1984: *GNU Emacs* (a powerful text editor scriptable in Lisp) is an immediate success
- 1987: the *GNU C Compiler* is another success
- People want to run GNU programs (on proprietary Unix systems) because they are technically better, even before the GNU system is complete

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention
- 1984: *GNU Emacs* (a powerful text editor scriptable in Lisp) is an immediate success
- 1987: the *GNU C Compiler* is another success
- People want to run GNU programs (on proprietary Unix systems) because they are technically better, even before the GNU system is complete
- People use the software and are introduced to Stallman's ethical vision. The project gets momentum.

Early development of GNU

Unix is made of many small programs, which can be developed independently by replacing proprietary Unix components *piece by piece*.

- Early GNU software gets little attention
- 1984: *GNU Emacs* (a powerful text editor scriptable in Lisp) is an immediate success
- 1987: the *GNU C Compiler* is another success
- People want to run GNU programs (on proprietary Unix systems) because they are technically better, even before the GNU system is complete
- People use the software and are introduced to Stallman's ethical vision. The project gets momentum.
- *The Free Software Foundation* is created

The Free Software definition

A piece of software is free software for you if you have:

The Free Software definition

A piece of software is free software for you if you have:

0.

the freedom to run it, for any purpose

The Free Software definition

A piece of software is free software for you if you have:

0.

the freedom to run it, for any purpose

1.

the freedom to study how it works, and change it to make it do what you wish.

The Free Software definition

A piece of software is free software for you if you have:

0.

the freedom to run it, for any purpose

1.

the freedom to study how it works, and change it to make it do what you wish.

2.

the freedom to redistribute copies so that you can help your neighbor.

The Free Software definition

A piece of software is free software for you if you have:

0.

the freedom to run it, for any purpose

1.

the freedom to study how it works, and change it to make it do what you wish.

2.

the freedom to redistribute copies so that you can help your neighbor.

3.

the freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits.

The Free Software definition

A piece of software is free software for you if you have:

0.

the freedom to run it, for any purpose

1.

the freedom to study how it works, and change it to make it do what you wish. **[Access to the source code is a precondition for this]**

2.

the freedom to redistribute copies so that you can help your neighbor.

3.

the freedom to improve the program, and release your improvements (and modified versions in general) to the public, so that the whole community benefits. **[Access to the source code is a precondition for this]**

"Free speech, not free beer"

The word "free" is ambiguous in English.

"Free speech, not free beer"

The word "free" is ambiguous in English.

Free software does *not* mean "non-commercial":

“Free speech, not free beer”

The word “free” is ambiguous in English.

Free software does *not* mean “non-commercial”:

- Selling copies is fine.

“Free speech, not free beer”

The word “free” is ambiguous in English.

Free software does *not* mean “non-commercial”:

- Selling copies is fine.
- Being paid for services or development is fine

“Free speech, not free beer”

The word “free” is ambiguous in English.

Free software does *not* mean “non-commercial”:

- Selling copies is fine.
- Being paid for services or development is fine
- Not everybody is a programmer: if the software is free, **any** programmer can make custom modifications for anybody else

“Free speech, not free beer”

The word “free” is ambiguous in English.

Free software does *not* mean “non-commercial”:

- Selling copies is fine.
- Being paid for services or development is fine
- Not everybody is a programmer: if the software is free, **any** programmer can make custom modifications for anybody else
    - Lock-in doesn’t happen with free software

“Free speech, not free beer”

The word “free” is ambiguous in English.

Free software does *not* mean “non-commercial”:

- Selling copies is fine.
- Being paid for services or development is fine
- Not everybody is a programmer: if the software is free, **any** programmer can make custom modifications for anybody else
    - Lock-in doesn’t happen with free software
    - (“anti-features” are also *much* easier to prevent, by the way)

"Free speech, not free beer"

The word "free" is ambiguous in English.

Free software does *not* mean "non-commercial":

- Selling copies is fine.
- Being paid for services or development is fine
- Not everybody is a programmer: if the software is free, **any** programmer can make custom modifications for anybody else
    - Lock-in doesn't happen with free software
    - ("anti-features" are also *much* easier to prevent, by the way)

Now there are several successful examples of commercial free software companies.

GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

- the *GNU Emacs* editor
- *GCC*, now a world-class compiler, plus the *GNU C Library*

GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

- the *GNU Emacs* editor
- *GCC*, now a world-class compiler, plus the *GNU C Library*
- the *GNU binary utilities*: assemblers, linkers, ...
- the *Bison* parser generator
- the *Bash* shell
- command-line utilities

GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

- the *GNU Emacs* editor
- *GCC*, now a world-class compiler, plus the *GNU C Library*
- the *GNU binary utilities*: assemblers, linkers, ...
- the *Bison* parser generator
- the *Bash* shell
- command-line utilities
- already **several millions lines of code (!)**

GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

- the *GNU Emacs* editor
- *GCC*, now a world-class compiler, plus the *GNU C Library*
- the *GNU binary utilities*: assemblers, linkers, ...
- the *Bison* parser generator
- the *Bash* shell
- command-line utilities
- already **several millions lines of code (!)**
- ...plus other non-GNU free software: the X window system, TEX, ...

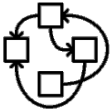
GNU at the beginning of the 1990s

At the beginning of the 1990s the GNU system has many contributors, is nearly complete and some components are widely popular:

- the *GNU Emacs* editor
- *GCC*, now a world-class compiler, plus the *GNU C Library*
- the *GNU binary utilities*: assemblers, linkers, ...
- the *Bison* parser generator
- the *Bash* shell
- command-line utilities
- already **several millions lines of code (!)**
- ...plus other non-GNU free software: the X window system, TEX, ...
- *no GNU kernel yet*

Problems with the GNU Hurd

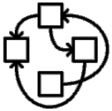
1991: the GNU kernel, *"the Hurd"* is started by Michael (later Thomas) Bushnell



- A very ambitious *multi-server architecture* based on the Mach micro-kernel

Problems with the GNU Hurd

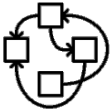
1991: the GNU kernel, *"the Hurd"* is started by Michael (later Thomas) Bushnell



- A very ambitious *multi-server architecture* based on the Mach micro-kernel
- The project has always been plagued by technical and personality problems. It's now improving but still slow, missing drivers and incomplete [getting some momentum again, lately]

Problems with the GNU Hurd

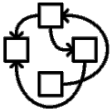
1991: the GNU kernel, *"the Hurd"* is started by Michael (later Thomas) Bushnell



- A very ambitious *multi-server architecture* based on the Mach micro-kernel
- The project has always been plagued by technical and personality problems. It's now improving but still slow, missing drivers and incomplete [getting some momentum again, lately]
- Stallman pushed for using Mach, thinking it would made the development simpler. However:

Problems with the GNU Hurd

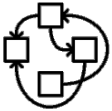
1991: the GNU kernel, *"the Hurd"* is started by Michael (later Thomas) Bushnell



- A very ambitious *multi-server architecture* based on the Mach micro-kernel
- The project has always been plagued by technical and personality problems. It's now improving but still slow, missing drivers and incomplete [getting some momentum again, lately]
- Stallman pushed for using Mach, thinking it would made the development simpler. However:
  - Mach has serious flaws, not yet recognized at the time

Problems with the GNU Hurd

1991: the GNU kernel, *"the Hurd"* is started by Michael (later Thomas) Bushnell



- A very ambitious *multi-server architecture* based on the Mach micro-kernel
- The project has always been plagued by technical and personality problems. It's now improving but still slow, missing drivers and incomplete [getting some momentum again, lately]
- Stallman pushed for using Mach, thinking it would made the development simpler. However:
  - Mach has serious flaws, not yet recognized at the time
  - a multi-server operating system is *very* hard to debug

Linux

- 1991, Helsinki: Linus Torvalds writes *Linux*, a kernel for the i386 processor. He releases it as free software in 1992

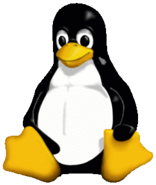


Figure: The Linux logo

Linux

- 1991, Helsinki: Linus Torvalds writes *Linux*, a kernel for the i386 processor. He releases it as free software in 1992
    - Traditional monolithic design...
    - ..but well-written and very fast

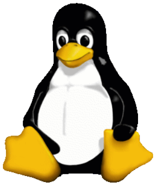


Figure: The Linux logo

Linux

- 1991, Helsinki: Linus Torvalds writes *Linux*, a kernel for the i386 processor. He releases it as free software in 1992
    - Traditional monolithic design...
    - ..but well-written and very fast
- Linus has a friendly personality and is good at managing the project over the Internet
    - many contributions from external developers

Figure: Linus Torvalds

Linux

- 1991, Helsinki: Linus Torvalds writes *Linux*, a kernel for the i386 processor. He releases it as free software in 1992
    - Traditional monolithic design...
    - ..but well-written and very fast
- Linus has a friendly personality and is good at managing the project over the Internet
    - many contributions from external developers
- Linux works with the GNU system: a complete free software operating system now finally exists...

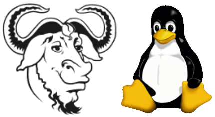


Figure: GNU + Linux

Linux and GNU

- ...but Linus refuses to acknowledge the role of the GNU Project, de-emphasizing the political message. *He calls "Linux" the complete system.*

Figure: Linus Torvalds

Linux and GNU

- ...but Linus refuses to acknowledge the role of the GNU Project,
  de-emphasizing the political message.
  He calls "Linux" the complete system.
- He doesn't care about the ethics message of free software: he clams he does it *"just for fun"*.

Figure: Linus Torvalds

Linux and GNU

- ...but Linus refuses to acknowledge the role of the GNU Project, de-emphasizing the political message. *He calls "Linux" the complete system.*
- He doesn't care about the ethics message of free software: he clams he does it *"just for fun"*.
- Linus a good communicator, and his project becomes much more visible than GNU.

Figure: Linus Torvalds

Linux and GNU

- ...but Linus refuses to acknowledge the role of the GNU Project, de-emphasizing the political message. *He calls "Linux" the complete system.*
- He doesn't care about the ethics message of free software: he clams he does it *"just for fun"*.
- Linus a good communicator, and his project becomes much more visible than GNU.
- The GNU/Linux system is a success, but people are not exposed to the political message of free software any more. Most new users even ignore GNU's existence.

Figure: Linus Torvalds

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a "killer application" and becomes a central part of the Internet revolution of the 1990s.

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a “killer application” and becomes a central part of the Internet revolution of the 1990s.

- Several companies “package” the GNU/Linux system on CDs making it easy to install.

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a "killer application" and becomes a central part of the Internet revolution of the 1990s.

- Several companies "package" the GNU/Linux system on CDs making it easy to install.
- New projects (*KDE* and then *GNOME*, from the GNU Project) make the system easier to use for non-technical people

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a "killer application" and becomes a central part of the Internet revolution of the 1990s.

- Several companies "package" the GNU/Linux system on CDs making it easy to install.
- New projects (*KDE* and then *GNOME*, from the GNU Project) make the system easier to use for non-technical people
- Many new users and developers are exposed to the system (usually just as "Linux")
    - an estimated 20,000,000 installed base by the end of the decade.

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a "killer application" and becomes a central part of the Internet revolution of the 1990s.

- Several companies "package" the GNU/Linux system on CDs making it easy to install.
- New projects (*KDE* and then *GNOME*, from the GNU Project) make the system easier to use for non-technical people
- Many new users and developers are exposed to the system (usually just as "Linux")
  - an estimated 20,000,000 installed base by the end of the decade.
- GNU/Linux becomes a strong competitor against proprietary operating system

1990s: The GNU/Linux system spreads

The GNU/Linux system is efficient and reliable, particularly strong as a *server* operating system.
The *Apache* web server is a "killer application" and becomes a central part of the Internet revolution of the 1990s.

- Several companies "package" the GNU/Linux system on CDs making it easy to install.
- New projects (*KDE* and then *GNOME*, from the GNU Project) make the system easier to use for non-technical people
- Many new users and developers are exposed to the system (usually just as "Linux")
  - an estimated 20,000,000 installed base by the end of the decade.
- GNU/Linux becomes a strong competitor against proprietary operating system
- *Proprietary applications* are ported to the system.

1998: The Open Source Movement (1)

- 1998: Eric S. Raymond, long time Emacs contributor, urges people to rename “Free Software” into “Open Source”.
- *No references to freedom*: Stallman’s strongly political message “scares away investors”

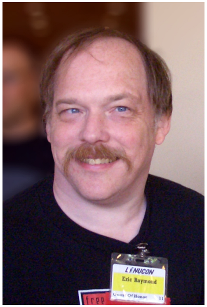


Figure: Eric Raymond

1998: The Open Source Movement (1)

- 1998: Eric S. Raymond, long time Emacs contributor, urges people to rename "Free Software" into "Open Source".
- *No references to freedom*: Stallman's strongly political message "scares away investors"
- The Open Source movement does not consider proprietary software an ethical problem. Open Source software is preferred *just because of practical reasons*:
  - It's flexible and tends to be of better quality
  - It's frequently gratis

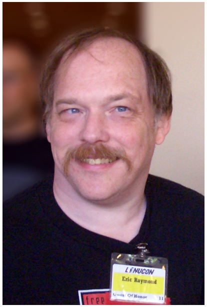


Figure: Eric Raymond

1998: The Open Source Movement (2)

- Raymond writes *"The Cathedral and the Bazaar"* analyzing the development models making Open Source/Free Software a success

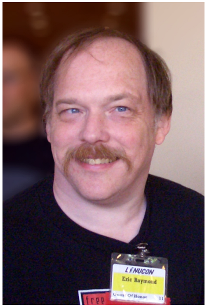


Figure: Eric Raymond

1998: The Open Source Movement (2)

- Raymond writes *"The Cathedral and the Bazaar"* analyzing the development models making Open Source/Free Software a success
- Raymond and other Open Source advocates contribute a lot to "software engineering" in the distributed model of the Internet; yet they don't see proprietary software as a problem

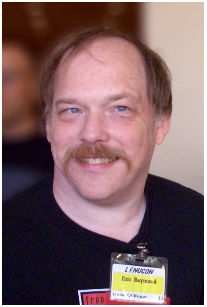


Figure: Eric Raymond

1998: The Open Source Movement (2)

- Raymond writes *"The Cathedral and the Bazaar"* analyzing the development models making Open Source/Free Software a success
- Raymond and other Open Source advocates contribute a lot to "software engineering" in the distributed model of the Internet; yet they don't see proprietary software as a problem
- Stallman and other Free Software advocates strongly react to such an a-political stance and distance themselves, but they have become a minority

Figure: Richard Stallman

Free Software and Open Source

- The two movements consider Free Software / Open Source essentially *the same set of programs*
  - the Open Source definition is formulated differently, but in practical terms it describes almost the same set of software as the Free Software definition
  - almost the same set of licenses!
- Very different philosophies, but there is frequent cooperation
  - often contributors *in the same project* have different views
- The Free Software movement is regaining visibility (but it needs your support)
- Free/Open Source projects continue to grow in number

Legal aspects

Warning: I'm not a lawyer!

"Intellectual property"

In the Free Software movement we're against using the term "Intellectual property": it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark
- Copyright
- Patent

"Intellectual property"

In the Free Software movement we're against using the term "Intellectual property": it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark

  a distinctive sign or indicator used by an individual, business organization, or other legal entity to identify that the products or services to consumers with which the trademark appears originate from a unique source, and to distinguish its products or services from those of other entities.

- Copyright

- Patent

"Intellectual property"

In the Free Software movement we're against using the term "Intellectual property": it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark

  a distinctive sign or indicator used by an individual, business organization, or other legal entity to identify that the products or services to consumers with which the trademark appears originate from a unique source, and to distinguish its products or services from those of other entities.
- Copyright

  exclusive right for a certain time period in relation to that work, including its publication, distribution and adaptation, after which time the work is said to enter the public domain.
- Patent

"Intellectual property"

In the Free Software movement we're against using the term "Intellectual property": it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark

  a distinctive sign or indicator used by an individual, business organization, or other legal entity to identify that the products or services to consumers with which the trademark appears originate from a unique source, and to distinguish its products or services from those of other entities.

- Copyright

  exclusive right for a certain time period in relation to that work, including its publication, distribution and adaptation, after which time the work is said to enter the public domain.

- Patent

  a set of exclusive rights granted by a state (national government) to an inventor or their assignee for a limited period of time in exchange for a public disclosure of an invention

“Intellectual property”

In the Free Software movement we’re against using the term “Intellectual property”: it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark

  a distinctive sign or indicator used by an individual, business organization, or other legal entity to identify that the products or services to consumers with which the trademark appears originate from a unique source, and to distinguish its products or services from those of other entities.

- Copyright

  exclusive right for a certain time period in relation to that work, including its publication, distribution and adaptation, after which time the work is said to enter the public domain.

- Patent **[Software must not be patentable. Period]**

  a set of exclusive rights granted by a state (national government) to an inventor or their assignee for a limited period of time in exchange for a public disclosure of an invention

"Intellectual property"

In the Free Software movement we're against using the term "Intellectual property": it confuses very different aspects and laws, and wrongly suggests that abstract entities should be treated like material objects.

- Trademark

  a distinctive sign or indicator used by an individual, business organization, or other legal entity to identify that the products or services to consumers with which the trademark appears originate from a unique source, and to distinguish its products or services from those of other entities.

- Copyright [We're concerned about this today]

  exclusive right for a certain time period in relation to that work, including its publication, distribution and adaptation, after which time the work is said to enter the public domain.

- Patent **[Software must not be patentable. Period]**

  a set of exclusive rights granted by a state (national government) to an inventor or their assignee for a limited period of time in exchange for a public disclosure of an invention

Copyright, in general (1)

- Moral rights (only in France and some other countries)
- Patrimonial rights

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights
  - *«Les droits patrimoniaux assurent à l'auteur un monopole d'exploitation économique sur ses œuvres. L'auteur a le pouvoir d'autoriser ou d'interdire toute communication, reproduction ou adaptation de ses créations.»*

    The author can concede a license to somebody to exempt him/her from his monopoly [in France: patrimonial rights only]

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights
  - *«Les droits patrimoniaux assurent à l'auteur un monopole d'exploitation économique sur ses œuvres. L'auteur a le pouvoir d'autoriser ou d'interdire toute communication, reproduction ou adaptation de ses créations.»*

    The author can concede a license to somebody to exempt him/her from his monopoly [in France: patrimonial rights only]

In practice the legislation is relatively uniform all around the world (1886 Berne convention, TRIPS).

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights
  - *«Les droits patrimoniaux assurent à l'auteur un monopole d'exploitation économique sur ses œuvres. L'auteur a le pouvoir d'autoriser ou d'interdire toute communication, reproduction ou adaptation de ses créations.»*

    The author can concede a license to somebody to exempt him/her from his monopoly [in France: patrimonial rights only]

In practice the legislation is relatively uniform all around the world (1886 Berne convention, TRIPS).

In France copyright lasts for the author's lifetime + 70 years for physical persons (**longer** in some special cases).

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights
  - *«Les droits patrimoniaux assurent à l'auteur un monopole d'exploitation économique sur ses œuvres. L'auteur a le pouvoir d'autoriser ou d'interdire toute communication, reproduction ou adaptation de ses créations.»*

    The author can concede a license to somebody to exempt him/her from his monopoly [in France: patrimonial rights only]

In practice the legislation is relatively uniform all around the world (1886 Berne convention, TRIPS).

In France copyright lasts for the author's lifetime + 70 years for physical persons (**longer** in some special cases). Then the work becomes public domain

Copyright, in general (1)

- Moral rights (only in France and some other countries)
  - the paternity of a work must be recognized
- Patrimonial rights
  - *«Les droits patrimoniaux assurent à l'auteur un monopole d'exploitation économique sur ses œuvres. L'auteur a le pouvoir d'autoriser ou d'interdire toute communication, reproduction ou adaptation de ses créations.»*

    The author can concede a license to somebody to exempt him/her from his monopoly [in France: patrimonial rights only]

In practice the legislation is relatively uniform all around the world (1886 Berne convention, TRIPS).

In France copyright lasts for the author's lifetime + 70 years for physical persons (**longer** in some special cases). Then the work becomes public domain (in theory: they could retroactively change the law *again*)

Copyright, in general (2)

Copyright is obtained *automatically*. Writing a copyright notice like

Copyright © 2012, Jacques Lefevre

may make the situation more clear, but is no longer mandatory.

In France and some other countries you can *register* your work at a government agency just to make it easier to prove your autorship in the future, but it is *not required*.
[Perversely, in France this agency is a *private* entity. Look for "APP", Agence de Protection des Programmes]

Copyright for software

Software is treated *like a literary work*. In Computer Science terms we're concerned about copies of some "string of text", be it source or binary, of some significant length (often one says over 15 source lines, but that's just a guideline).

In practice how complex the code is, or the algorithm employed, doesn't matter: we speak about "text".

In France by default a work's copyright is held by *the author's employer* if the work is part of the author's job.

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is “making a derived work”.

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is “making a derived work”.
- **Linking** two pieces of software together is “making a derived work”:

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is “making a derived work”.
- **Linking** two pieces of software together is “making a derived work”:
    - Static linking
    - Dynamic linking

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is "making a derived work".
- **Linking** two pieces of software together is "making a derived work":
  - Static linking
  - Dynamic linking
  - Invoking an external program doesn't count as linking

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is “making a derived work”.
- **Linking** two pieces of software together is “making a derived work”:
    - Static linking
    - Dynamic linking
    - Invoking an external program doesn’t count as linking
    - Network communication doesn’t count as linking

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is "making a derived work".
- **Linking** two pieces of software together is "making a derived work":
  - Static linking
  - Dynamic linking
  - Invoking an external program doesn't count as linking
  - Network communication doesn't count as linking
  - Writing two programs on the same physical medium doesn't count as linking

Copyright for software: derived works and linking

- Of course, making a modification to a piece of software is "making a derived work".
- **Linking** two pieces of software together is "making a derived work":
  - Static linking
  - Dynamic linking
  - Invoking an external program doesn't count as linking
  - Network communication doesn't count as linking
  - Writing two programs on the same physical medium doesn't count as linking
- So *as a practical, a posteriori guideline* the address space seems to be the "barrier" (calling the kernel is not "linking", for example).

Licenses

With a **license** an author permits somebody else to perform some activities on which he/she has a monopoly by default (for example, making copies), at some conditions.

License notices in source files tend to look like:

```
/* Copyright (C) 2012, Jacques Lefevre
   This work is licensed under the Foo license.
   See the file COPYING for the full license text. */
```

Within comments, at the beginning of files.

License compatibility

When you link two pieces of software, you have to respect both their licenses.

If one license requires to do something forbidden by the other one, *you can't link the two pieces of software*.

License compatibility

When you link two pieces of software, you have to respect both their licenses.

If one license requires to do something forbidden by the other one, *you can't link the two pieces of software*.

Some licenses are *incompatible*

License compatibility

When you link two pieces of software, you have to respect both their licenses.

If one license requires to do something forbidden by the other one, *you can't link the two pieces of software*.

Some licenses are *incompatible*

- That's **one** reason why inventing new licenses tends to be a bad idea

Free Software licenses

Very simply, a piece of sofware is free software for you if its license grants you all four freedoms 0..3.

Why do I say “for you”?

What happens when you receive a software with a free sofware license allowing you to redistribute it under a different license?

Why do I say “for you”?

What happens when you receive a software with a free sofware license allowing you to redistribute it under a different license?

- It’s still free software **for you**

Why do I say “for you”?

What happens when you receive a software with a free sofware license allowing you to redistribute it under a different license?

- It’s still free software **for you**
- Not necessarily for who receives it from you.

Why do I say “for you”?

What happens when you receive a software with a free sofware license allowing you to redistribute it under a different license?

- It’s still free software **for you**
- Not necessarily for who receives it from you.

- Many commonly-used licenses allow you to do that

Copyleft

A *copyleft* license is a free software license requiring that derived works maintain the same license.

There are two “varieties” of copyleft:

Copyleft

A *copyleft* license is a free software license requiring that derived works maintain the same license.

There are two “varieties” of copyleft:

- *Strong copyleft*: the license must be kept equal in *all* derived works (modifications and linking)

Copyleft

A *copyleft* license is a free software license requiring that derived works maintain the same license.

There are two "varieties" of copyleft:

- *Strong copyleft*: the license must be kept equal in *all* derived works (modifications and linking)
- *Weak copyleft*: the license must be kept equal in modified versions; *not necessarily* for the result of linking the software with something else

Copyleft

A *copyleft* license is a free software license requiring that derived works maintain the same license.

There are two “varieties” of copyleft:

- *Strong copyleft*: the license must be kept equal in *all* derived works (modifications and linking)
- *Weak copyleft*: the license must be kept equal in modified versions; *not necessarily* for the result of linking the software with something else

How does copyleft work, legally?

- With copyright!

Copyleft

A *copyleft* license is a free software license requiring that derived works maintain the same license.

There are two "varieties" of copyleft:

- *Strong copyleft*: the license must be kept equal in *all* derived works (modifications and linking)
- *Weak copyleft*: the license must be kept equal in modified versions; *not necessarily* for the result of linking the software with something else

How does copyleft work, legally?

- With copyright!
- A subversive hack on the legal system [RMS]

Examples of free software licenses

- Strong copyleft: GNU GPL
- Weak copyleft: GNU LGPL
- No copyleft: X11, BSD (both versions, but the older one is GPL-incompatible so please don't use it)

Examples of free software licenses

- Strong copyleft: GNU GPL **(our preferred choice)**
- Weak copyleft: GNU LGPL
- No copyleft: X11, BSD (both versions, but the older one is GPL-incompatible so please don't use it)

A final word

Thank you.

And thanks to the hosting organization for the opportunity of giving this slightly subversive talk.

In case you're interested in contacting me:

positron@gnu.org
http://ageinghacker.net

I made some changes suggested by Ludovic Courtès and José Marchesi in Summer 2012. Thanks!

For more information I

- Richard M. Stallman
  Free Software, Free Society: Selected Essays of Richard M. Stallman
  GNU Press, Boston, 2002
  also freely downloadable from the Net
- Steven Levy
  Hackers: Heroes of the Computer Revolution
  ISBN 0-385-19195-2, Anchor Press/Doubleday, 1984
  about the early history of hacker culture and communities, written by an outsider

Image credits I

- http://www.gnu.org/graphics/heckert_gnu.html
  Author: Aurelio A. Heckert
  GFDL 1.3, the Free Art License, or under CC-BY-SA 2.0. It's also a trademark but the FSF permits use without permission when speaking of GNU in a supportive and accurate way.
- https://en.wikipedia.org/wiki/File:NicoBZH_-_Richard_Stallman_%28by-sa%29_%2810%29.jpg
  Author: NicoBZH from Saint Etienne - Loire, France
  This file is licensed under the Creative Commons Attribution-Share Alike 2.0 Generic license.

Image credits II

- `https://en.wikipedia.org/wiki/File:PDP-10_1090.jpg`
  Author: Michael L. Umbricht, The Retro-Computing Society of RI. The original uploader was Sun-collector at `en.wikipedia` CC-BY-SA-2.5.
- `https://en.wikipedia.org/wiki/File:KL10-backplane.jpg`
  Author: Shieldforyoureyes, Dave Fischer
  This file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license.

Image credits III

- `https://en.wikipedia.org/wiki/File:Linus_Torvalds.jpeg`
  Author: Unknown photographer; the copyright holder is linuxmag.com
  GFDL. Permission of Martin Streicher, Editor-in-Chief, linuxmag.com.
- `https://en.wikipedia.org/wiki/File:Eric_S_Raymond_portrait.jpg`
  Authors: jerone2, Bilby
  This file is licensed under the Creative Commons Attribution-Share Alike 2.0 Generic license.

Image credits IV

- https://en.wikipedia.org/wiki/File:Tux.svg
  Author: Larry Ewing, Simon Budig, Anja Gerwinski
  The copyright holder of this file allows anyone to use it for any purpose, provided that the copyright holder is properly attributed. Redistribution, derivative work, commercial use, and all other use is permitted.
- https://en.wikipedia.org/wiki/GNU_Hurd#mediaviewer/File:Hurd-logo.svg
  Original METAFONT by Stephen McCamant. Converted into hand-written SVG by Colin Leitner and Thomas Schwinge
  CC BY-SA 3.0

Image credits V

- `https://en.wikipedia.org/wiki/File:LISP_machine.jpg`
  Authors: Jszigetvari, Hydrargyrum
  Disjunctive licensing: GFDL 1.2+, CC BY-SA 1.0, CC BY-SA 2.0, CC BY-SA 2.5, CC BY-SA 3.0