EVIDENCE®

EMBEDDING TECHNOLOGY

Summary of open-source licenses

Claudio Scordino

Paolo Gai

Why open-sourcing ?

1. Code doesn't need to be written from scratch
   - Code from similar open-source applications can be reused

2. Free help and support from a development community

3. Positive image of the company
   - E.g. Google
   - Easier to hire talented developers

4. Better code:
   - Developers encouraged to write better code
   - Free code review by a high number of developers

Background: possible freedoms

- to use a software
    - Commercial (e.g. Windows)
- to distribute a software
    - Shareware/Freeware (e.g. WinZip)
- to modify and distribute a software w/out releasing source
    - BSD/MIT licenses (e.g. FreeBSD)
- to modify and distribute a software by releasing source
    - GPL licenses (e.g. Linux)
    - Common misconception: the modified code must be provided only to the "final recipient" (i.e. people receiving the binaries in whatever form). And only at their request.
    - Original authors keep IP rights and can re-license the provided code

Background: selling software

Open-source ≠ free (as in "free beer")

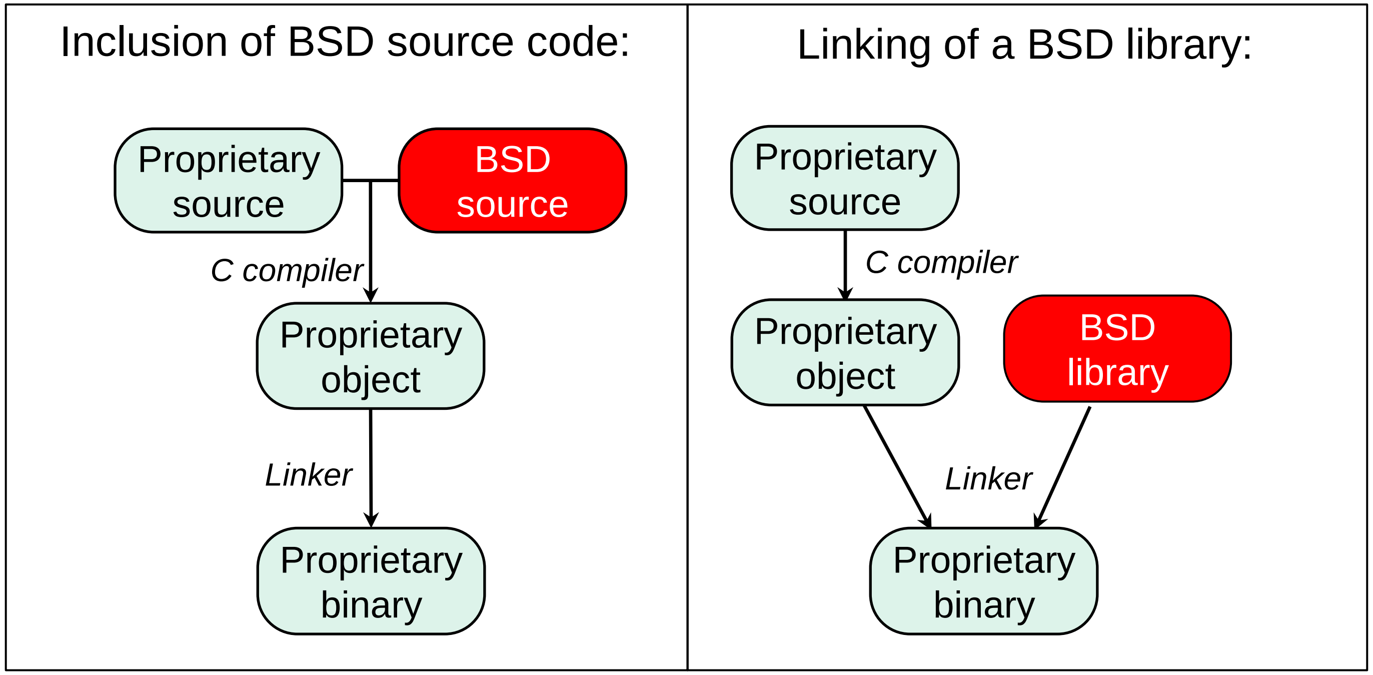
- Most open-source licenses allow to sell the software
- Even Microsoft allows access to some source code by subscription
- The point is more about *access to knowledge* than to gratuitousness

BSD/MIT

- Families of licenses
- Permissions/constraints:
  - Use/modify/distribute the binaries
  - Modified/linked code doesn't become BSD/MIT
  - No need to disclose the source code
  - Must acknowledge the original authors
- Most permissive licenses
  - "Industry-friendly": existing code can be re-used without disclosing our own source code
  - No "pollution" as the license doesn't affect other code, even in case of inclusion or linking
- Examples: FreeBSD, FreeRTOS

Summary of BSD/MIT

- Proprietary code never becomes BSD/MIT:

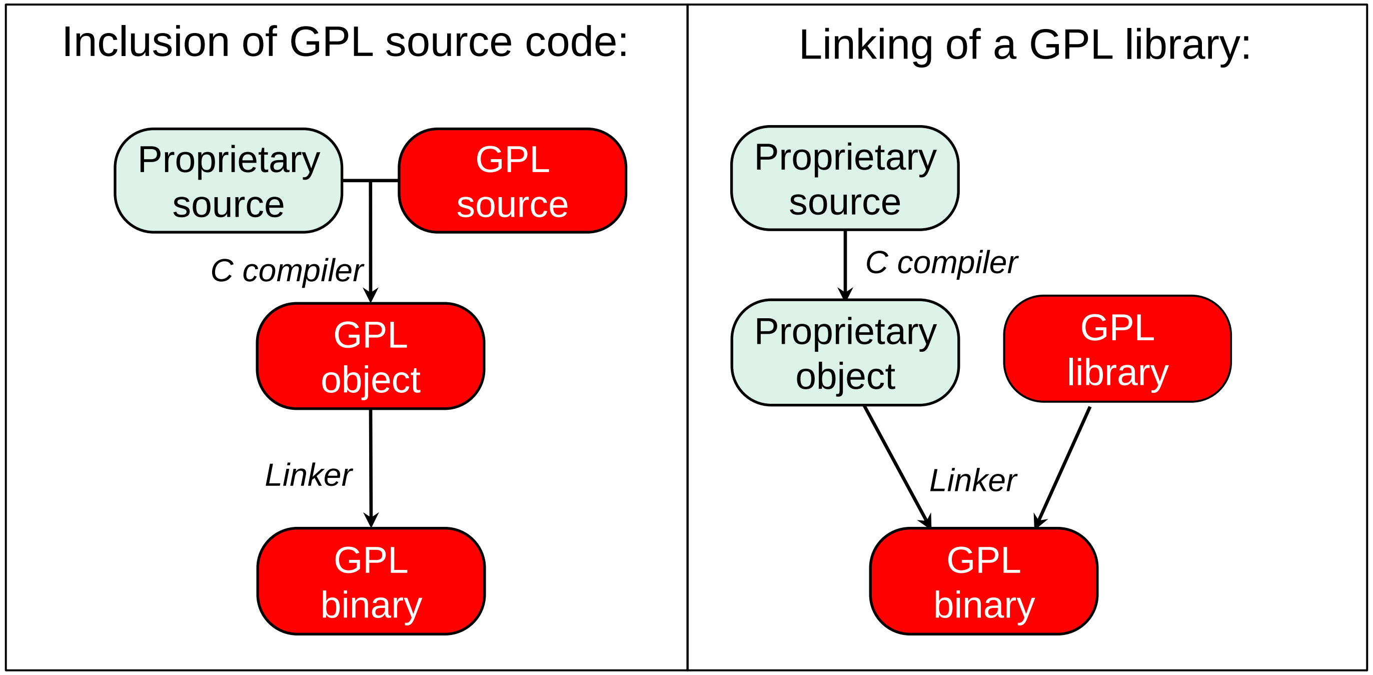


GNU General Public License (GPL)

- Published by the Free Software Foundation
- Permissions/constraints:
  - Use/modify/distribute the binaries as long as the recipient has access to the source code and maintains the same rights
  - Modified/linked code becomes GPL
  - "Pollution": changes are and remain under GPL
  - GPL code cannot be used in/linked with non-GPL code
- Examples: Linux kernel, Jailhouse hypervisor

GNU General Public License (GPL)

- Proprietary code becomes GPL if:
  - It includes a portion of GPL source code
  - It is linked againsy a GPL license



GPL version 3

- Version 3 of GPL added some extra restictions:
  - Patent protection: grants to recipients all patents needed for using/distributing the GPL software
  - No hardware-based lockdown mechanisms ("tivoization") can be implemented for preventing the user running a modified version of the GPL code.
    Unacceptable by most embedded manufacturers.
  - Cracking Digital Rights Management (DRM) included in GPL code is legal and cracked code can be redistributed

Linux and user-space

- On Linux, kernel and user-space applications are not linked (either at run-time)
  - Syscalls allow applications to invoke kernel services
  - Therefore, the kernel's GPL license doesn't affect the user-space.

- Different for embedded RTOSs:
  - Typically, the application code is linked against the RTOS
  - If the RTOS is under GPL, then the application code becomes GPL and <u>must be released</u>.

Linux and modules (an exception ?)

- Linux allows to implement Loadable Kernel Modules (LKM)
  - These modules are linked to the kernel at run-time

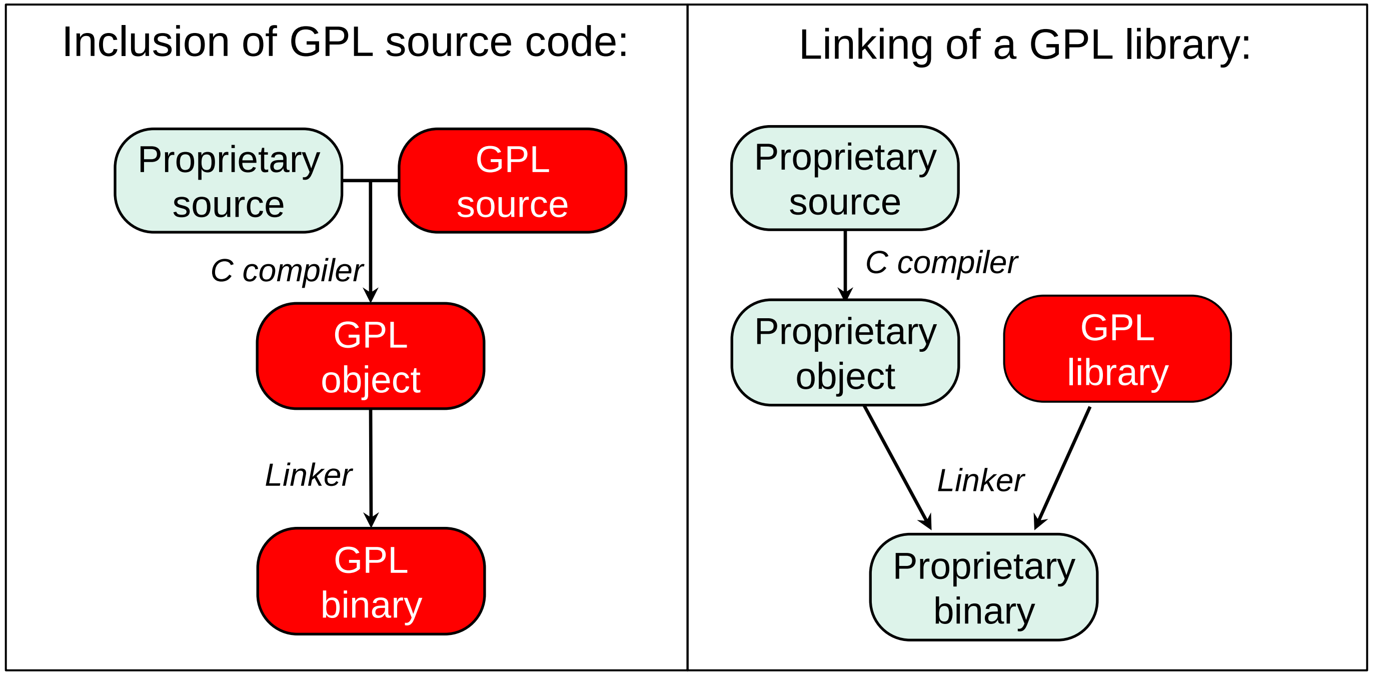
- Grey area: non-GPL modules are tolerated, however
  - They can't access all services provided by the kernel (some are available only to GPL modules)
  - When loading a non-GPL module, the kernel is marked as "tainted"

GNU Lesser General Public License (LGPL)

- Published by the Free Software Foundation
- Permissions/constraints:
  - Use/modify/distribute the binaries as long as the recipient has access to the source code and maintains the same rights
  - Modified code becomes LGPL
  - Use of LGPL libraries with proprietary code is permitted. However it must be possible to link the program against a newer version of the LGPL library (i.e. forces dynamic linking)
- Examples: Qt, GLIBC

GNU Lesser General Public License (LGPL)

- Proprietary code becomes LGPL if:
  - It includes a portion of LGPL source code



GPL with Linking Exception

- Also known as "Classpath"
- It is the GPL with an execption allowing linking between proprietary code and GPL code
- More permissive than LGPL: no need to allow linking to newer/modified versions of the LGPL library
- Examples: ERIKA version 2

Summary

	Inclusion of proprietary code	Linking of proprietary objects	Notes
BSD MIT	😀	😀	*Need to acknowledge the original authors.*
GPL	😡	😡	
LGPL	😡	🤔	*Only dynamic linking (allows to relink to a newer version of the LGPL library).*
GPL with Link. Except.	😡	😀	

Contacts

Evidence Srl

Via Carducci 56

56010 S.Giuliano Terme

Pisa - Italy

Web: http://www.evidence.eu.com

E-mail: info@evidence.eu.com

Phone: +39 050 99 11 122