Introduction to Free Software

Definition, History, and Licensing

Free Software Course

November 24, 2025

Table of Contents

Definition of Free Software

Free Software refers to software that respects users' freedom and community.

"Free" means **freedom**, not price.

Free software grants users the liberty to:

- Run the program for any purpose
- Study and modify the source code
- Redistribute copies
- Distribute modified versions

The Four Essential Freedoms

According to the Free Software Foundation (FSF):

- **Freedom 0:** The freedom to **run** the program as you wish, for any purpose
- **Freedom 1:** The freedom to **study** how the program works, and change it
- **Freedom 2:** The freedom to **redistribute** copies so you can help others
- **Freedom 3:** The freedom to **distribute copies of your modified versions** to others

Access to source code is a precondition for freedoms 1 and 3

Origins: The Early Days (1950s-1970s)

- Free software emerged in the early days of computing when sharing source code was common practice.
- In the **1950s and 1960s**, software was often bundled with hardware and freely shared among researchers.
- Universities and labs exchanged source code openly, fostering collaboration and innovation
- By the late 1970s, commercialization began: companies started selling proprietary software, restricting access to source code.

1980s:GNU Project and FSF

When software became closed, Richard Stallman started a movement to keep it free.

- **In 1983**, Richard Stallman announced the **GNU Project**, starting the effort to build a free operating system.
- To support this vision, **in 1985** he founded the Free Software Foundation (**FSF**), an organization to promote and protect software freedom.
- As part of this work, the FSF introduced the GNU General Public License (**GPL**), which guarantees users the freedom to use, study, change, and share software.

GNU = "GNU's Not Unix"

Linux and Open Source

- In 1991, Linus Torvalds released the Linux kernel, giving the missing piece that GNU needed.
- When the Linux kernel was combined with GNU tools, the result was a complete free operating system, known as GNU/Linux.
- Later, in 1998, the term “Open Source” was introduced to make this idea more business-friendly and easier to accept in industry.

2000s–Present: Mainstream Adoption

- Web servers (Apache, Nginx) run most of the internet websites.
- Operating systems: GNU/Linux, Android dominate servers and mobile.
- Collaborative platforms (GitHub, GitLab) expand participation.

Understanding Software Licenses

What is a Software License?

A legal instrument governing the use and redistribution of software

Purpose of Free Software Licenses:

- Grant the four essential freedoms
- Protect users' rights
- Define terms of distribution and modification
- Ensure software remains free (in some cases)

Categories of Free Software Licenses

Free software licenses fall into two main categories:

Copyleft Licenses

Require derivative works to be distributed under the same license (reciprocal)

Permissive Licenses

Allow derivative works to be distributed under different terms, including proprietary licenses

Copyleft Licenses

Copyleft uses copyright law to ensure software remains free

Key principles:

- Anyone can use, modify, and distribute the software
- **But:** Modified versions must be distributed under the same license
- Prevents "proprietization" of free software

"Copyleft is a general method for making a program free and requiring all modified and extended versions to be free as well" - Richard Stallman

We will examine two copyleft licenses: GP and LGPL

Example 1: GNU General Public License (GPL)

The most widely used copyleft license

Key features:

- Strong copyleft: derivative works must be GPL-licensed
- Source code must be made available
- Users receive all four freedoms
- **Current versions:** GPLv2 (1991), GPLv3 (2007)

Notable GPL software: Linux kernel, Git, and WordPress

Example 2: GNU Lesser General Public License (LGPL)

Weaker copyleft for libraries

Key features:

- Allows linking with proprietary software
- Only modifications to the library itself must be shared
- Designed for software libraries
- Encourages wider adoption in commercial software

Use case: When you want a library to be widely adopted, including in proprietary applications

Notable LGPL software: Qt, GNU C Library

Permissive Licenses

Key Principles of Permissive Licenses

Permissive licenses are characterized by:

- **Minimal restrictions:** Few requirements on how software can be used
- **Allow proprietary derivatives:** Modified versions can be closed-source
- **No reciprocity requirement:** No obligation to share modifications
- **Simple compliance:** Usually just require attribution and license notice
- **Business-friendly:** Encourage commercial adoption and integration
- **Maximum freedom for downstream users:** Choose their own licensing

We will examine two permissive licenses: MIT, BSD,

Example 1: MIT License

One of the most permissive licenses

Key features:

- Very short and simple license text
- Minimal restrictions on reuse
- Allows proprietary derivatives
- Only requires: copyright notice and license text in copies
- No copyleft provisions:Take MIT-licensed code, Modify it however you want, Keep your modifications private/secret, Sell it as proprietary (closed-source) software, Change the license of your modified version to anything you want, and Not share the source code with anyone
- No explicit patent grant

Use case: Maximum freedom and adoption, minimal legal complexity

Notable MIT software: Node.js, Ruby on Rails, jQuery, React

Example 2: BSD License (3-Clause)

Family of permissive licenses

Key features:

- Very permissive, similar to MIT
- **3-Clause addition:** Cannot use project name for endorsement
  Example: If you build software using code from "Project X," you cannot advertise your product as "Approved by Project X" or "Officially supported by Project X" unless the maintainers give you written permission.
- Prevents unauthorized use of contributors' names in marketing

Use case: Permissive licensing with name protection

Notable BSD software: FreeBSD, NetBSD, Nginx, Django

Conclusion

Key Takeaways:

- Free software is about **freedom**, not price
- The four freedoms define what makes software free
- Free software movement has shaped modern computing
- **Copyleft licenses** ensure software stays free
- **Permissive licenses** maximize adoption flexibility
- Choose a license based on your goals and values

Resources: fsf.org, opensource.org, choosealicense.com