

# **Software Agents and Electronic Commerce**

**CS486-25**

## ***Intellectual Property #2***

Chris Brooks

Department of Computer Science

University of San Francisco

# Distributing Digital Intellectual Property

---

- Once you've made an intellectual property, such as a program, you need to decide what to do with it.
- You need to decide the *terms* under which people can use your software.

## 25-1: Licensing

---

- A license specifies the terms under which a piece of software can be used. It can grant all of the terms that copyright does, or just a subset. It may specify:
  - rules about copying
  - use for profit
  - number of installations
  - ownership (do you own the software, or just use it?)
  - Responsibility for usage
  - technical support

## 25-2: Commercial Licenses

---

- These are the traditional licenses that come with most shrink-wrapped software.
- Legal contract: as copyright holder, the authors can choose whatever usage terms they like.
- They typically prohibit users from tampering with, modifying or redistributing the software.
- Often, ownership remains with the vendor; users are technically leasing the code.
- Typically limits liability; vendor is not responsible for damage.
- May also restrict usage, duration, number of machines, reverse engineering.

## 25-3: Public Domain

---

- The simplest way to freely distribute your program is to put it into the public domain.
- No copyright is retained; users can do whatever they want with your program.
- This includes selling it or converting it into a proprietary product and copyrighting the derivative work.

## 25-4: Copyleft

---

- Copyleft is more restrictive than putting something in the public domain.
- Copyleft is an idea promoted by the GNU foundation
- Copyleft requires that anyone who redistributes software, either original or changed, must release this software under copyleft.
- Note that you *can* charge for this new software, but you can't change the licensing terms.
- This provides developers with an incentive to develop free software

## 25-5: Copyleft

---

- Stallman: “If you will make your software free, you can use this code.”
- Example: The C++ front-end to gcc was developed in industry. GNU’s copyleft required it to be copylefted, since it used copylefted code.

# 25-6: Examples of Copylefted Software

---

- Emacs
- gcc
- Many unix tools
- Anything released under the GNU Public license

## 25-7: GNU Public License

---

- The GNU Public license is the most well-known implementation of the “Free Software” principles.
- Distribution terms: everyone has the right to use, modify and redistribute copylefted code as long as the rights to this code or any derived code are not changed.
- In other words, you can't weaken the use conditions (or strengthen restrictions) on copylefted code.

## 25-8: Free Software

---

- It's useful to think a bit about what “free software” really means.
- “Free as in speech, not as in beer.”
- In other words, you have the right to do what you want with it, not that you're obligated to get it without paying.

## 25-9: Free Software

---

- GNU software freedoms:
  - The freedom to run the program for any purpose
  - The freedom to study the program and how it works (i.e. source code access)
  - The freedom to redistribute copies
  - The freedom to modify the program and redistribute the modified version

## 25-10: The Free Software Foundation

---

- Started by Richard Stallman in 1985 as a response to the overwhelmingly proprietary nature of software at the time.
  - No free OSes (Unix was very expensive), few free tools.
- Goal: develop a free OS, along with a set of tools for that OS (editor, compiler, mail reader, etc)
- This system was referred to as GNU (Gnu's not Unix)
- When the linux kernel was developed in the early 1990s, GNU finally had a free OS to run on.

## 25-11: Lesser GPL

---

- GNU/FSF also offers something called the “Lesser GPL”
- Allows GPL’ed libraries to be linked into non-free code, which can then be proprietary.
- Goal: encourage adoption of a particular standard
- Or, the free library may do the same job as a widely-existing non-free library.
  - For example, the GNU C library.

## 25-12: Open Source

---

- *Open Source* is a broader term that covers a number of different possible licensing agreements.
- Basic criteria:
  - License must not prevent redistribution, and must allow free distribution.
  - Source code for the program must be publically available.
  - Derived works must be covered under the original license.
  - License may require modifications to be distributed a separate patches, or be clearly named or identified.

## 25-13: Open Source

---

- License may not discriminate against groups of users.
  - For example, countries may have export restrictions. An open source license may remind users to obey the law, but cannot explicitly incorporate these restrictions.
  - Also cannot restrict use to, for example, only non-commercial entities
- Cannot require that it be distributed only with other open-source works.

# 25-14: Examples of Open Source software

---

- Mozilla
- Apache
- Perl, Python, all GNU tools
- PHP
- openSSH
- Much more ...

## 25-15: Economics of Open Source

---

- A common objection to open source is: “But I need to eat!”
- If your program is freely available, how can you make money from it?
- Maintenance/development: Most programmers spend a large fraction of their time maintaining tools or modifying them for a particular business. this won't change if the underlying codebase is open source.

## 25-16: Economics of Open Source

---

- Support/consultation. Just because something is free doesn't mean there's no value for people who are experts at using it.
- Paid development. If a tool is widely accepted or deemed valuable, industry will subsidize its development, even if the end product is open source, because it's valuable to them.
  - e.g. Apache, g++, perl, ANTLR

# 25-17: Open Source as a Business Model

---

- Some managers may also be reluctant to adopt an open-source model.
  - “Why should we give away something that cost us lots of \$ to develop?”
  - Reliability: open source is arguably more reliable, as more people are using and testing the code.
  - Technical superiority: this is an open argument, but OS proponents argue that having more developers leads to a better product.
  - Rapid response. Faster turnaround times allow businesses to respond to customer needs more easily.

# 25-18: Open Source as a Business Model

---

- Market penetration. OS may allow a firm to add developments that would not be feasible otherwise. For example, ports to less popular operating systems.
- Remember, software has a “network effect” - getting a large user base makes it more valuable.

# 25-19: Open Source as a Business Model

---

- Potential Open-source business models:
- Support sellers (RedHat): Give away the product, sell support, after-sale service.
- Loss-leader. (Netscape, Java): You give away your software to help sales of related, closed software.

# 25-20: Open Source as a Business Model

---

- Product improvement. (SGI) A hardware company supports and ships open-source software (such as Samba) that improves their other products.
- Accessorizing. (O'Reilly, VA research) Selling systems or manuals that use or describe open-source systems.
- It's not *too* different a consideration from the discussion on how to sell information goods when copying is prevalent.

## 25-21: Creative Commons

---

- These are just a few of the potential ways in which you may want to control the use of your IP.
- Creative Commons is an organization started by IP and CS experts devoted to increasing people's ability to control access to their work.
- Provides a broad set of potential licenses for creators of IP.
- Goals: Increase amount and accessibility of online source material.
- Also provides a standard metadata encoding for these licenses using HTML and RDF.
  - This makes the licenses standardized and machine-readable.

## 25-22: Creative Commons

---

- Some CC licenses:
- Attribution: Anyone can use your work, but they must give you credit.
- Noncommercial. People can use and display your work for noncommercial purposes.
- No derivative works. People can use or copy your work, but not modify it.
- Share alike. People can distribute derivative works as long as original licensing terms remain.
- Full copyright. You retain all rights to how your work is used.

## 25-23: Summary

---

- There is a wide spectrum of possible ways in which IP can be distributed.
- There is no “one-size-fits-all” solution.
- There are cases in which open-source/GPL distributions make a lot of sense.
- There are also cases in which proprietary solutions make sense.
- Consider the specifics of your software, users, and developers.