Intellectual property

- protection mechanisms
  - trade secrets
  - trademarks
  - patents
  - copyrights
  - licenses
- standards and standardization
- open source / free software
- Warning: IANAL

Patents & copyrights

- US Constitution, Article 1, Section 8:

- "The Congress shall have Power ...
  To promote the Progress of Science and useful Arts, by securing for limited Times to Authors and Inventors the exclusive Right to their respective Writings and Discoveries;

- "Writings": copyright protects expression but not idea
  - you can't copy my program
  - but you can implement the same idea in some different form
- "Discoveries": patent protects an idea
  - you can't use my patented idea
  - but you can achieve the same effect in a different way

- the meaning of "different" is NOT usually clear

Patents

- **exclusive right to make, use or sell an invention in US**
- **valid for 20 years after filing**

- **requirements:**
  - statutory subject matter:
    process, machine, article of manufacture, composition of matter
  - novel
  - useful
  - unobvious to person having ordinary skill in the art
    at the time of filing

- **contents:**
  - abstract
  - drawings/diagrams
  - specifications (narrative description, preferred embodiment)
  - claims

Copyright

- protects expression, not idea
- duration used to be 17 years + one renewal
- now life + 70 years, or 95 years for commercial works
  - (the "Mickey Mouse Protection Act", 1998)
- "fair use" permits limited copying under some circumstances
  - criticism, comment, scholarship, research, news reporting, teaching
- uncertain what fair use really is -- case by case decisions
- considerations:
  - purpose and character of the use
  - nature of the copyrighted work
  - amount and substantiality of the portion used
  - effect of the use on potential market or value of the copyrighted work
- recent copyright laws may prevent some fair uses
  - can't decrypt to make excerpt for teaching or criticism
  - can't reverse engineer to make copies in different media

DMCA: Digital Millennium Copyright Act (1998)

- US copyright law: www.copyright.gov/title17, Chapter 12
- anticircumvention: illegal to circumvent a technological measure protecting access to or copying of a copyrighted work
  - limited exceptions for reverse engineering for interoperability, encryption research, security testing
- illegal to remove or alter copyright notices and management information
- "safe harbor": protects ISPs from copyright infringement claims if they follow notice and takedown procedures

Copyright issues in software

- code
  - theft in commercial setting
  - plagiarism in academic setting
- visual appearance, "look and feel", etc., of a program
- interfaces vs implementations
- reverse engineering?
  - clean room implementation
- copyright or patent?
  - which is appropriate to protect specific piece of software?

Licenses

- an agreement (e.g., contract) that allows a particular use of some software
  - that might otherwise be a violation of copyright, patent, etc.

- are shrinkwrap and clickwrap licenses valid and enforceable?
- is licensing replacing purchase?
- are warranty and liability disclaimers for software valid?

Open source / free software

- **source code: instructions in a readable programming language**
  - usually has significant commercial value
    e.g., Windows, Office, TurboTax, Photoshop, …
  - usually proprietary, secret, not revealed
    even if compiled version is given away (e.g., iTunes, Internet Explorer)

- **"open source": source code is available, can be use, copied and modified**
  - a reaction to restrictions on proprietary code
  - promoted by Free Software Foundation, other open source projects & groups
- **various kinds of licenses determine what can be done with it**
  - mainly concerned with keeping source code open enough that others can continue to build on it and improve it
  - prevents anyone from taking it private / proprietary
- **a viable threat to proprietary software in important areas**

Free Software Foundation (Richard Stallman, MIT, ~1985)

- plan to build an operating system and all supporting software
  - "GNU" -- "GNU's not Unix"
- started non-profit organization called the Free Software Foundation
- wanted source code to be released so that it could not be made proprietary, would remain free forever
  - "free" as in "free speech", not "free beer"
    - ok to charge for distribution, support, etc.

- source released under copyright agreement that requires that any subsequent distribution be covered by the same agreement
- GNU GPL (General Public License): "copyleft"
  - full permission to use, copy, modify, distribute modifications
  - copies, derivative works, etc., must have the same terms if distributed
  - copies, etc., must have the same license attached to them
  - NO permission to add further restrictions; explicitly forbidden
- source code has to be freely available
  - can't "take it private"

Real programs (warning: flaky numbers ahead)

- 6th Edition Unix 9,000 lines
- Linux: 24 M lines, 70K files, 47,200 source files (v 5.3.7, 10/19)
  – includes many drivers, tests, etc., not all needed, not all simultaneous

- first C compiler: about 2,700 lines
- GCC C/C++/… compiler: 4.8 M lines, 35,000 files (v 9.2.0, 10/19)

- Firefox: 7.6 M lines, 22,800 files C++ (v 63, 10/18)
  – plus 59,000 Javascript files with about 6 M lines

- Windows 98: 18 M lines (but what's included?)
- Windows XP: 38 M lines
- Windows Vista: 100 M lines?
- Windows 7, 8, 10: ?????

What's hard about big programs?

- lots of components with hidden or implicit connections
  - a change in one place has unexpected effects elsewhere

    software as spaghetti
  - most errors occur at interfaces between components
  - language features, software design, etc., devoted to reducing and controlling interconnections
- changes in requirements or environment or underpinnings
  - each change requires rethinking, adapting, changing -- a fresh chance to get something wrong
- constraints on performance, memory, schedule, …
  - force people to create more complicated code or cut corners
- coordination and cooperation among groups of people
  - management complexity grows rapidly with size of organization
  - Brooks's Law: Adding manpower to a late software project makes it later

Fundamental Software Ideas

- **algorithm: sequence of precise, unambiguous steps**
  - performs some task and terminates
  - based on defined basic / primitive operations
  - describes a computation independent of implementation details
- **programming language:**
  - grammar, syntax, and semantics for expressing computation
    - notation is important
- **program: algorithms implemented in a programming language**
- **compilers, interpreters: programs that convert from the high level language used by people to a lower level**
  - a compiler is a program that writes a program
  - an interpreter also acts as a computer so the program can be run
- **libraries and components: programs written by others**
  - packaged in a form that can be used in a new program
- **abstraction, layers, interfaces, virtualization**
  - hiding details, pretending to be something else
- **bugs: the need for absolute precision**
  - cover all cases, cope with failures and misuse