Software property topics

- **protection mechanisms**
  - trade secrets
  - trademarks
  - patents
  - copyrights
  - licenses

- **standards and standardization**
  - FRAND licensing
  - data formats

- **open source / free software**

Trade secrets

- **information is a secret held by its owner**

- **disclosed only under some kind of agreement**
- **basically a contract / binding legal agreement between two parties ("non-disclosure agreement" or NDA)**

- **no recourse if secrecy is lost**

- **often used as an argument about why information should not be made public**
  - voting machine technology
  - breathalyzer technology
  - ...

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;

- **copyright protects expression but not idea**
  - you can't copy my program
  - but you can implement the same idea in some different form
- **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**

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?

Patent issues for software

- **what is patentable? (statutory subject matter)**
  - software itself
  - business methods

    (whether implemented in software or not?)

- **what's novel?**
  - how new and unobvious does something have to be?

- **what should be patentable?**
- **is the term too long?**
  - policy questions

Patent reform?

- **patent system is showing real strain, if not actually broken**
  - large and growing number of applications each year
  - number of examiners is not growing nearly fast enough
  - standards for prior art and obviousness may be too low
    especially for software and business methods
  - new technologies are always a problem
- **lots of money at stake, lots of powerful interests**
  - patent "trolls" or "non-practicing entities" who use patents to try to extract money by litigation
  - Open Patent, PubPat look for prior art to invalidate bad patents
- **recent Supreme Court cases try to decide what is patentable**
  - KSR v Teleflex: stronger standard of what is obvious (4/07)
  - Bilski: patentability of business methods
    decided June 2010: this patent rejected but door left open to others

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?**

Standards and standardization

- **standard: technical specification sufficiently precise that it ensures independent implementation, uniformity, interoperability, ...**
  - physical measurements: length, weight, time, chemical composition, ...
  - mechanical properties: plugs & sockets, CD/DVD dimensions, …
  - electrical properties: voltage, frequency, …
  - software: character sets, programming languages, operating system interfaces, data formats, information exchange protocols, ...
- **standardization: process of establishing a specification**
  - usually involves competing entities, so tradeoffs are needed between mutual benefit and competitive advantage
  - often international (e.g., ISO: International Organization for Standardization)
- **de facto vs de jure standards**
  - de facto: Windows, Office, Flash, PDF, ...
  - de jure: ASCII, Unicode, some programming languages, ...

FRAND: fair, reasonable and non-discriminatory licensing

- **licensing of patents for technologies required by standards**
- **fair: not anti-competitive,**
  - e.g., requiring other licenses or limiting sales to competitors
- **reasonable: aggregate price not too high**
  - can't price so high that patent costs would make something unmarketable
- **non-discriminatory:**
  - same terms to all (though ok to have volume discounts, etc.)

Data formats

- **many data formats are proprietary**
  - can only be produced and interpreted by proprietary software
  - e.g., Microsoft Office 2003 (Word, Excel, ...), Flash, ...
  - if available at all, may require royalty payment to owner
  - can be used to control markets, maintain competitive advantage
    e.g., by incompatible upgrades or incomplete disclosure
- **some formats involve patent protection, usually because the algorithm is patented**
  - patent owner may require payment
  - e.g., (in theory) JPEG, GIF, MP3, FAT
- **"open" formats are non-proprietary**
  - can be produced and interpreted by anyone for free
  - e.g., HTML, PNG (portable network graphics), ODF (open document format), PDF (as of 2008)

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 copied and used**
  - 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 Free Software Foundation**
- **wanted source code to be released so that it could not be converted to 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"

Open source examples

- **Linux, other Unix variants**
  - FreeBSD (Mac OS X uses this), NetBSD, OpenBSD, OpenSolaris
- **Apache web server**
- **Mozilla web browser (Firefox), Chrome web browser**
- **OpenOffice**
  - work-alike for Microsoft Office
- **GIMP**
  - Photoshop alternative
- **GCC (GNU compiler collection)**
  - compilers for C, C++, Fortran, etc.
- **Perl, Python, PHP, …**
  - major programming languages
- **MySQL, SQLite and other database systems**
- **lots of smaller systems**
  - standard Unix tools, languages, etc.

Open source questions

- **why do programmers contribute?**
- **why do companies use it?**
- **is it better than commercial software?**
- **is it a serious threat to Microsoft and other companies?**
- **what economic model explains it?**
- **will its legal protections hold up?**
- **should it be required for crucial systems like electronic voting?**

Real programs (warning: some of these numbers are flaky)

- **6th Edition Unix 9,000 lines**
- **Linux: 14.8 M lines, 33,000 source files in C (v 3.6.5, 11/12)**
  - includes many device drivers, etc., not all needed, not all simultaneous

- **first C compiler: about 2,700 lines**
- **GCC C/C++/… compiler: 4.9 M lines, 43,500 files (v 4.7.2, 11/12)**

- **Firefox: 2.4 M lines, 11,000 files C++ (v 3.6.23, 11/11)**

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

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

Special purpose systems

- **not all computers run a general-purpose operating system**

- **game machines, cell phones, digital cameras, camcorders, e-book readers, …**
  - may run a single program specialized to a single task
  - but increasingly these use versions of standard operating systems, most often Linux

- **it's easier to build a new product if you can use off-the-shelf software as a controller**
  - much less work to get started
  - easier to add new features

Why software instead of hardware?

- **general-purpose software instead of special-purpose hardware:**

- **software is**
  - more flexible
  - easier to change in the field
  - cheaper to manufacture (though often costly to create originally)
- **hardware is**
  - faster, more efficient
  - more reliable, more robust
  - more secure against intrusion, theft, reverse engineering

- **dividing line is not always clear**
  - flash memory, etc.
  - plug-in cards, game cartridges

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: there are many, many languages
- **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