

CPSC156: The Internet Co-Evolution of Technology and Society

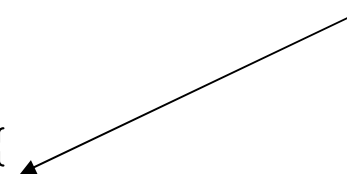
Lecture 12: February 22, 2007

Open Source

Writing Code

```
int WINAPI WinMain (HINSTANCE hInst, HINSTANCE
    hPrevInst, LPSTR lpCmdLine, int nCmdShow)
{
    WINDOWCLASS wc;
    HWND hwnd;
    populatewc (&wc);
    if (RegisterWndClass (&wc) ) {
        if (hwnd = CreateWindow ( ... )) {
            ShowWindow (hwnd, nCmdShow);
            ...
        }
    }
}
```

Source code can refer to objects like "windows" and "files." The machine doesn't understand this! It's too complex.



These are lines of text written in *C* for the beginning of a Windows program. A computer cannot be fed this code to run it! This code must be combined with many other pieces and turned into something more basic that the processor can understand.

Compiling to Machine Language

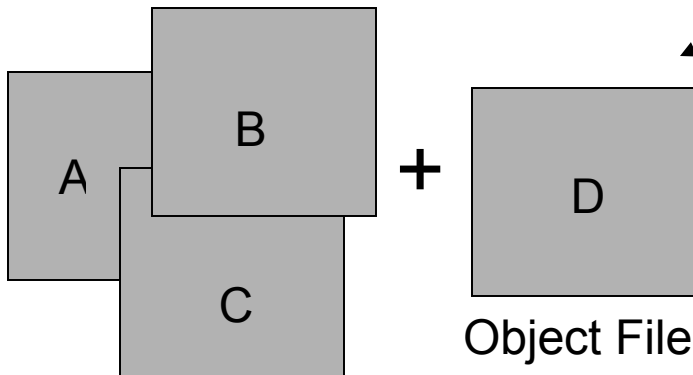
```
int WINAPI WinMain (HINSTANCE hInst, HINSTANCE  
hPrevInst, LPSTR lpCmdLine, int nCmdShow)  
{  
    WINDOWCLASS wc;  
    HWND hwnd;  
    populatewc(&wc);  
    if (RegisterWndClass(&wc)) {  
        if (hwnd = CreateWindow( ... )) {  
            ShowWindow(hwnd, nCmdShow);  
            ...  
        }  
    }  
}
```

Source Code

compiler →

```
010010100001111101  
10101110111101010  
01010101001111101  
0100011010010101  
0100010010101100  
...
```

Object File



Other pieces of code
needed by the program

Distributable
Executable Image

Each set of bits represents a simple instruction to the processor (e.g., "read block 5 from the hard disk" or "make this screen pixel green").³

A Software Business Model

- Create an idea for useful software.
- Develop software (write **source code**).
- Compile software (on specific platforms) to **binary version**.
- Package and sell **binary version**.
Legal uses of the software are controlled by a **software license**.
- Issue upgrades: Add features that people want, and sell new versions.

Business-Model Pros

- The developers' intellectual property rights are protected.
 - People use the software exactly as the developers want them to.
 - Users never see how the software works and so others can't steal the nuts and bolts.
- Developers receive compensation for their hard work! There's an incentive to code and create new ideas.
- “The market” determines what is good, including software and hardware platforms.
 - Operating systems and machine architectures good for users *and* developers are the ones that flourish.

Business-Model Cons

- Users can't customize software to their needs, and the developer may not be able to satisfy all users' requests for changes.
- Binary versions run on certain machines only:
 - Forces people to buy specific platforms
 - Can't necessarily use one machine to do everything
- “Standards” can cause monopolies (?)
- Source code hidden
 - Can't be improved and studied by others
 - Hard to design interoperable products (why does MS Office on MS Windows work especially well?)
- Using computers is expensive!

How Much Does Software Cost?

Source: Amazon.com, 2/26/2007

- Operating Systems:
 - Microsoft:
 - Windows Vista Home Premium, full version: \$228
 - Windows Server Standard 2003, 5-client: \$1495
 - Apple: MacOS X 10.4.6, Tiger full version: \$104
- Office Suites:
 - Microsoft Office Professional Edition 2007: \$438
 - Apple iWork '06: \$70
- Other Software:
 - Adobe Acrobat Standard 8.0 (documents): \$290
 - Adobe Photoshop CS2 (graphics): \$585
 - Microsoft Visual Studio 2005 Professional (programming): \$680

An Alternative Approach

- “Free” software: Gives users the power to use software as they wish.
 - “‘Free software’ is a matter of liberty, not price ... ‘free’ as in ‘free speech,’ not as in ‘free beer’ ...”
--- Free Software Foundation
- Development is not controlled by a small group; people can learn from the code.
- Revenue models can still develop around software distributed with a *free license* !

Open Source

- Technical definition:
Anyone can look at the source code.
- Benefits:
 - Interoperability
 - Education
 - Cross-platform compatibility (if code can be compiled by users)
- Still protects intellectual property:
 - Uses of code are still limited by a license.
 - Developers maintain rights to code and official releases of the product.

Consequences of Open Source

- Software can be closely scrutinized; performance can be analyzed and attributed to parts of code.
- Ideas behind code can become standards.
- **Distributors** can specialize in building more features on top of open-source software, offering customized packages with support options.
- “Open source” is a more business-friendly term than “free.”

Free Software

- Technical definition (from the *Free Software Foundation*): Users have the freedom to:
 - (1) run the software, for any purpose;
 - (2) study how the program works and adapt it to their needs;
 - (3) redistribute copies;
 - (4) improve the program and release improvements to the public.
- Access to source code is necessary for (2) and (4); so, “Free” can include “Open Source.”

Consequences of Free Software

- “Free” software can be modified, used, and even sold as users see fit.
- Selling free software requires having additional services that the user wants:
 - Packaging, delivery, installation mechanisms
 - Help, support, training
 - Customizing software for specific needs
- Better versions of software can be released by any user.

Software Licenses

- The software license indicates what users can do with software and code.
- **Traditional licenses** strictly govern use of software based on purchase.
- **Open-source and free licenses** indicate how code can be used, reused, and distributed, usually asserting user rights like the “four freedoms.”

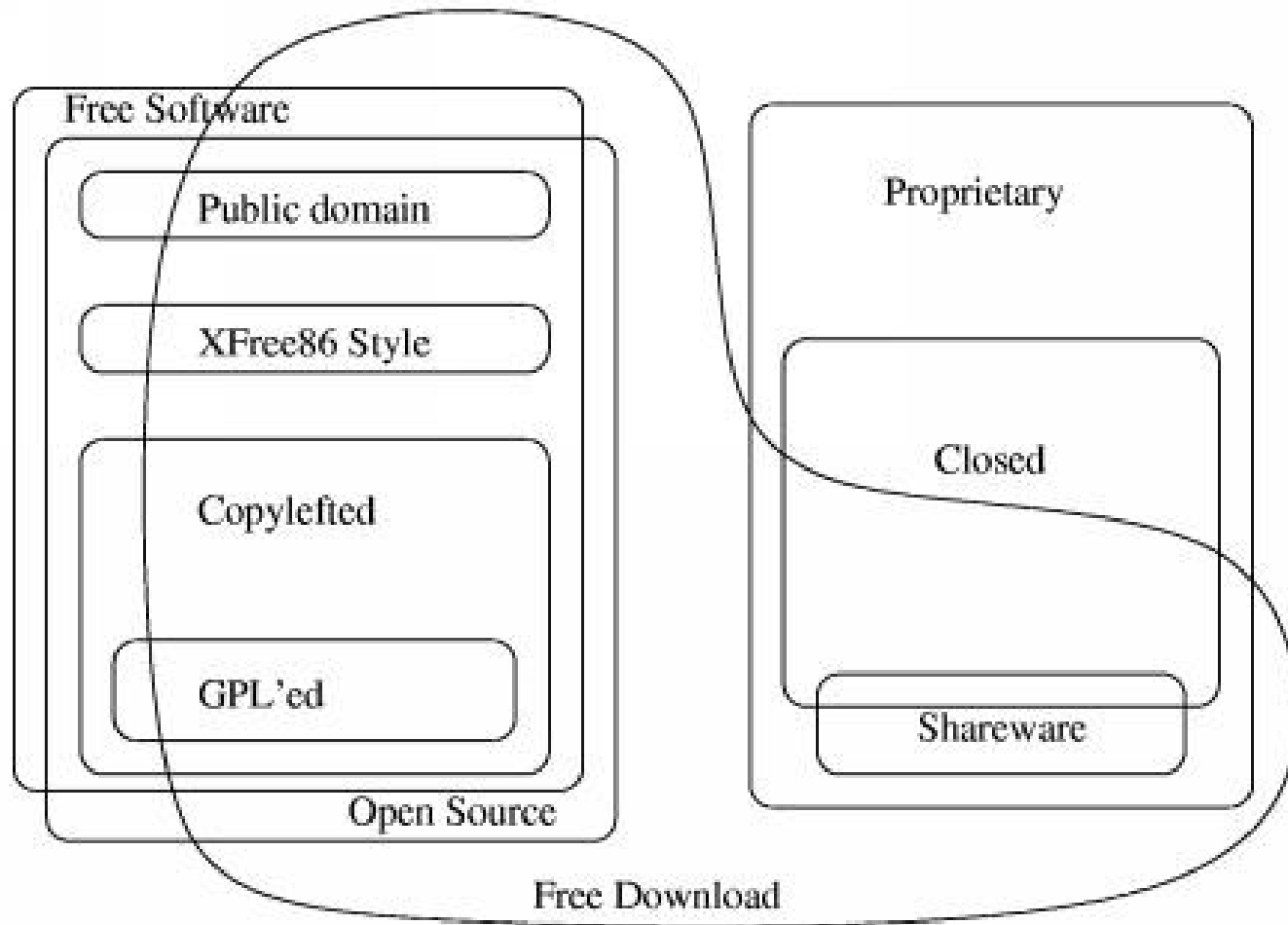
GNU General Public License (GPL)

- Formalization of ideal software-distribution model by the Free Software Foundation and the GNU Project.
- Developers can choose to release their “free” software under the GPL license.
- Requires that users maintain the original copyright on the code and clearly mark any changes when distributing it.
- Source code is included, and users can modify and compile it where and how they see fit.
- **Copyleft:** Redistributed versions must give users the same rights (must include source code that can be modified, *etc.*).

Other Popular Licenses

- Many licenses exist that come from organizations that develop “free” software.
- **GPL-compatible** licenses are those that allow software covered to be combined with GPL-covered software to produce larger “free” products.
 - Examples (non-copyleft): MIT, BSD (Berkeley), X11 (windowing system)
- Other public licenses: Netscape-JavaScript, Artistic, W3C Software

Relationships Among Software Distribution Models



Source: Free Software Foundation

How The Movement Began

- 1983: GNU Project developed
 - Goal: to produce a UNIX-compatible free-software system (GNU = “GNU’s not Unix!”)
 - Idea conceived by Richard Stallman, who worked in an MIT group that exclusively used free software (~1975).
- 1985: Free Software Foundation
 - Group that manages GNU project and distributes GNU software

GNU History (*continued*)

- 1985: GNU Emacs editor available
 - First major piece of usable, free software
- 1990s: Most pieces of free system complete, except operating system kernel. Combined with Linux kernel by Linus Torvalds to produce a **GNU/Linux** OS distribution. This is a complete UNIX-compatible system that contains free-software tools.

Free UNIX-Compatible Systems

- MINIX (1987-) and Linux (1992-) are free OS kernels (originally) developed for academic use.
- GNU/Linux and BSD (Berkeley) are the two most popular UNIX-like OSs.
- Stable, robust systems incorporating:
 - standard Internet and networking protocols
 - standard development tools
 - many other free, widely-used tools
- Allow users to set up servers and workstations at little cost that can do almost anything PC- and Mac-compatible systems can do!

Tools in Free OS Distributions

- Web server, *e.g.*, Apache
- Mail server, *e.g.*, Sendmail
- Other Internet and network daemons, *e.g.*, SAMBA and other file servers, OpenSSH, FTP servers.
- Development tools, *e.g.*, compilers for many programming languages
- User tools, *e.g.*, web browsers, graphics tools, editors, spreadsheets, Ghostscript and other document tools, sound players and media tools
- Graphical user interfaces, *e.g.*, Gnome, KDE, and other windowing systems

Making Money From Open-Source Software

“Open source is the foundation of Red Hat's business model. It represents a fundamental shift in how software is created. The code that makes up the software is available to anyone. Developers who use the software are free to improve the software. The result: rapid innovation. ... Red Hat delivers the innovation of open source to our customers.”

“Red Hat solutions combine Red Hat Linux, developer and embedded technologies, training, management services, technical support—all enabled through a single delivery platform: Red Hat Network.”

--- Red Hat Investor Fact Sheet, 1/16/2001

Clever Distribution of Linux

- Red Hat offers its GNU/Linux-based OS for free download from its website, but offers no technical support. Download is huge and is only for experts!
- Red Hat sells packaged versions: CDs with manuals, technical-support options, and an easy installer that works with many computers.
- Red Hat will customize and install its OS for companies that make special-purpose devices, have specific security concerns, *etc.*
- Red Hat partners with computer manufacturers (*e.g.*, Dell) to create a cheaper alternative to Microsoft OSs.

The Marketing Spin

- “Red Hat Enterprise Linux,” “Red Hat Enterprise Network,” and “Red Hat Stronghold Enterprise” provide a Linux-based server core and management tools for businesses.
- Convince businesses that they will lower costs, have a smooth migration to Red Hat Linux, get support and training, and achieve more reliability and security.
 - The software is open source, so it adheres to standards, can be examined for flaws, and has many developers contributing to it.
 - Claim: It is worth paying Red Hat for the initial training and support. The switching costs are lower than the cost of maintaining more expensive (*e.g.*, Windows) systems.

Discussion Point

- As you heard in Katz's lecture on February 20, open-source software can be a component of A2K (access to knowledge) in developing economies.
- Google has made powerful search technology available to everyone on the Internet without anyone's having to buy it; the entire technology platform is advertiser-supported. Could advertising provide a comprehensive, alternative approach to A2K in developing economies?