INFORMATION TECHNOLOGY WANTS TO BE FREE

JONATHAN A. PORITZ

Abstract. *The free-software and open-source communities, dedicated to the open exchange of research and to the idea that knowledge is a public good, are the natural allies of academic faculty.*

Sometimes I hear through the grapevine, in the cohesive community where my regional comprehensive university is located, of a recent graduate who is using calculus in an unauthorized way. Perhaps this person is an engineer optimizing a process in one of our remaining local industries, an executive maximizing profit in a new venture, or even a soccer mom or dad doodling on a fast-food wrapper, trying to figure out the best location for defensive players in terms of how much of the field they can control. These and, let's face it, most applications of calculus to real life were not covered in my class, and therefore are infringing uses by my customers.

The recent explosion of nonfaculty university employees, which has come at the cost of actual teachers, means that even a small campus such as mine has a crack team of in-house intellectual property lawyers who are ready to swoop in on the food-wrapper doodlers and demand damages from these calculus pirates. The fines in such cases are rarely so high as to bankrupt the criminal, even in these tough times, because single incidents of this type do not threaten all future calculus revenue to the university.

More significant fines are often levied, however, when the offense is *viral*, meaning that its possibility of propagation from person to person calls into question the entire future business model of calculus instruction. When one of my students explains to her high-school-age brother how to use the chain rule to differentiate nearly any function that he could write down, this technology could propagate to potential future students and *their* younger siblings, depressing my prospects for any further income based on teaching rules of differentiation. It makes sense in such cases to seek a harsh judgement that will send a message to all college students about the importance of respecting the intellectual property of others.

Some will argue that the analogy implicit in this fictional (for the moment) scenario relating current university intellectual property law to limitations on the noninfringing uses of the fruits of education is flawed. I certainly won't defend it in detail. But it underlines a fundamental tension between the very mission of education and the current free-market-based intellectual property regime. Nowhere is this tension more visible, and more rife with potential for both great gain and great loss, than in the use of information technology on campus.

1. Scholarly Software

Let's consider another scenario, moving now from life in the hallowed halls of academe to a very different setting: the fast-paced, entrepreneurial world of IT companies.

Imagine I have an alter ego, James, working in a computer company. I spend most of my day preparing for and teaching several courses (the same ones I have taught for many years), grading dozens of student papers every day, and sitting on committees and doing other minor service tasks. James likewise spends most of his time keeping a corporate network running smoothly, writing database manipulation code or other applications that any undergraduate computer science major could as easily do, and occasionally participating in group meetings.

My peers judge me primarily on the basis of my scholarship, not those time-consuming daily activities. Pursuing this scholarship, I work many evenings and weekends, I read new books and journal articles, I attend conferences during my vacations, and I struggle to write papers and books. When I achieve breakthroughs, I write down every insight I have and send my work, with no expectation of financial remuneration, to anyone who expresses interest. I want my peers to be able to reproduce my work completely and to incorporate my best ideas into their own work.

James likewise works in his spare time, putting in the effort and finding the creativity to write amazing new programs that do things no one ever thought to do before or ever thought would be possible. Then he posts the new programs on his web page, along with detailed documentation (allowing anyone to use them effectively) and the source code of the programs (so anyone can take them apart and reuse whichever pieces seem convenient in their future work).

Since James is giving away the source code of his creative output, these programs are called *open source*. Note that he cannot charge for his open-source software, because with the source on his website, anyone who wishes can download the program and compile it for herself.

Although my hiring, tenure, and promotion depend profoundly on my scholarly activity, my salary is largely given to me for the teaching I do – certainly in the eyes of the legislators in my state – and not for the scholarship I simply give away as I try to build my reputation.

James is also paid a salary for his daily activities, but he earns prestige and advances his career by his open-source production.

Open-source programmers have coauthored the kernel of an operating system, along with all the various tools and utilities to make this OS functional and reliable – a task that has been likened in complexity to building a modern city skyscraper – by spontaneously self-organizing in their off hours. This open-source operating system is called GNU/Linux, often shortened to just Linux. (GNU stands for "GNU's Not Unix.")

Some software terminology:

Source Code: Human-readable and human-writable text which tells a computer how to do something. This is the *program*, as the programmer wrote it. The program may be written in any number of languages (such as `Java`, `Perl`, or `C`) but which the program's users need never see that programming language.

Compiler: A program that can read source code in one particular language and translate it into machine instructions (all in 1s and 0s) for a particular computer hardware.

Executable: The compiler's output when given a particular program to process. This is a data file which looks like complete gibberish but is the only version of the program that the hardware can actually run.

Operating System (OS): The large program that stands between software and the bare hardware of the machine and knows how to get input from particular devices, produce output on others, talk to the network, share hardware resources (for example, the processor and memory), between several programs that want to use them, and so on. `Windows`, `OS X`, `GNU/Linux`, `Android`, and `iOS`, are examples of operating systems.

The free-software community is the natural ally of academic faculty: both groups operate with the same economic model, only in different arenas. Furthermore, the ethos of free software is largely identical to that of higher education in its emphasis on enhancing the free exchange of ideas and the creativity and unfettered autonomy of thought exercised by the members of our communities.

The Linux Penguin

2. What Is Copyleft?

Richard Stallman, an early theorist of open-source computing, quit his job as a programmer at the Massachusetts Institute of Technology's Artificial Intelligence Lab in the 1970s to write a manifesto, along with a lot of GNU software. He did it without a corporate master overseeing his work. Planning ahead for the running battle that programmers were going to fight in order to keep a portion of their creative output freely available in the IT intellectual commons, he created the idea of "copyleft," a kind of anti-copyright.

"Copyleft is a general method for making a program (or other work) free, and requiring all modified and extended versions of the program to be free as well," Stallman writes in *What Is Copyleft?* He continues, "In the GNU project, our aim is to give all users the freedom to redistribute and change GNU

software. If middlemen could strip off the freedom, we might have many users, but those users would not have freedom."

The freedom here is, in Stallman's memorable phrase, "Free as in speech, not as in beer." While most people use the term *open-source software*, Stallman prefers *free software* and emphasizes the self-perpetuating nature of the freedoms in copylefted software. In an attempt to give it negative associations, Stallman's critics called this feature the "viral" nature of copyleft.

The GNU Logo

The epithet "viral" is not worrisome to academics: we always hope that the ideas in our scholarship go viral. And though it is unnecessary for me to say, for example, that anyone who uses a theorem of mine must not publish corollaries in his or her own work without publishing the proofs as well, in the IT world, it is necessary to formalize some kind of viral open-source requirement. To this end, Stallman created a series of licenses that open-source software authors can use. The most important of these is the GNU general public license, currently in version three: *GPLv3*.

Software authors now have the option of creating truly free software, protected by the general public license. The open intellectual commons they have thereby created is filled with tools that are perfectly suited for use in the academy. In fact, academics who use nonfree, closed-source software run the risk that their scholarship and pedagogy will come under the control of some corporate owner whose interests are far removed from the goals of academia.

3. **Open Culture**

Stallman's work with the GPL led directly to Lawrence Lessig and others founding the Creative Commons. This organization seeks to bring the ideas of copyleft to other, more traditionally copyrighted media such as text, music, video, and so on. For example, videos I made for a class this spring were posted to YouTube with a *Creative Commons Attribution-ShareAlike* license. The goal here is that creators who actively desire sharing, collaboration, and openness – such as academics! – can use an appropriate Creative Commons license, to opt out explicitly of the usual copyright regime.

Collectively, these various ideas of copyleft for software and cultural products like music and text met under the rallying cry "Information Wants To Be Free", which phrase seems to have a rather complicated history. Cory Doctorow, who is one of the most insightful thinkers on these issues writing today (all of the works to be found on his site `craphound.com` are worth reading), points out that "information doesn't want to be free, people do." While this is certainly true, as a revolutionary slogan IW2BF states a position in opposition to the commoditization of information. Academia has a long tradition of success with this open approach, and many current battles around higher education seek to defend the non-commodity approach at least behind our ivyed walls.

4. **And A Thousand Flowers Bloomed**

Meanwhile, the free software engineers went forth and multiplied.

Let us first describe the ecosystem of free tools and technologies.

Most (but not all) free software exists for the GNU/Linux OS. That OS itself exists in different versions, called *distributions* (or *distros*). The kernel and core GNU utilities are largely the same, but the other software included in a distro varies quite a bit.

Further, as the worldwide community of free-software developers adds new software and finds bugs or adds new features to existing software, GNU/Linux users like to get these updates. Different distros therefore manage *repositories* of executables (with sources), for those who want them, and users can chose which repositories to trust and when or how frequently to visit them.

The Ubuntu and LinuxMint Logos

At this time, the two most popular distros are Linux Mint and Ubuntu, both of which are very easy to use in a quite sophisticated manner by typing commands and editing configuration files but also allow simple point-and-click usage.

Beyond the distros themselves, there are by now thousands of free-software projects, many of them crucial to academics in their everyday teaching and research lives, including some of the most widely used software on the Internet. Here are a few examples:

Some Open Source Projects

- The majority of web servers on the Internet run the **Apache HTTP Server**.
- On top of Apache sits **MediaWiki**, the software which runs Wikipedia.
- Many web surfers access the web with the browser **Mozilla Firefox**.
- Many graphics designers use the sophisticated image manipulation program **GIMP**.
- Educators in both K-12 and higher education use the learning management system **Moodle**.
- For academic scientists and mathematicians, typesetting beautiful copy that looks like it was ripped directly from the pages of a scholarly journal is fairly easy with the TEX program; many (if not most) academic publishers in math and science now expect articles to be submitted in TEX.
- There are several office productivity suites with word processor, spreadsheet, and presentation programs, such as **LibreOffice**.
- **WeBWorK** is a online homework system for math and science supported in part by the National Science Foundation.
- **Kuali** is a suite of administrative tools for higher education that includes library management, student enrollment, scheduling and financial aid, and research administration.

The list goes on and on – applications for most things users want to do can be found in multiple free versions in the repositories.

5. Campus As Free Software Utopia

If universities are going to live up to their ideals, we must use free software and open standards whenever possible. Of course, we should not be coercive, so if an individual faculty member wants to use a closed OS on her office machine or wants to use some closed commercial software in a computer lab, she must be permitted to do so. But free software should be the default for individual machines, computer labs, and

How would this play out, on a practical level? Here are some problems, and possible solutions:

5.1. **Ease of Use.** Many people think free software is not user-friendly, that it is for IT geeks, and that users must learn to use arcane commands rather than point and click in graphical user interfaces. Such interfaces are supposed to be the result of the evolutionary pressures in a competitive market for commercial software.

The problem with this idea is that, as we all know, perfect markets do not exist, and monopoly positions, for example, can remove such competitive pressures. Since Windows has an enormously dominant position in the desktop OS market (of approximately 80 percent of machines, by some estimates), it would not be surprising if some Windows interfaces were in fact rather clunky. Which they are. Apple, struggling from behind, would also be predicted to have much more beautiful interfaces, which many users think is true.

The ease-of-use argument is also quite ironic in light of the susceptibility of closed-source software to data loss and other problems. Free software tends to be better thought out and tested than closed-source software because more people are looking into the source codeâĂŤin essence, free software undergoes the equivalent of academic peer review, unlike closed-source software.

5.2. **Diseases of a monoculture.** "Malware" is the catchall term security professionals use for viruses and other malicious software that appropriate the control of your machine or data. But GNU/Linux has fewer than ten documented malware strains, while the more popular closed operating systems have

tens if not hundreds of thousands of such viruses. Typical public computers running closed operating systems can be assumed to have a menagerie of malware.

The source of this malware epidemic on Windows computers is quite simple. The dominance of Windows in the desktop PC environment creates an ecosystem with little variation – a software "monoculture." Any diseases to which this one organism is susceptible will spread almost uncontrollably.

Hence, nonfree IT environments are so frequently compromised that the average user's productivity is significantly diminished.

5.3. **Servers.** This argument about the quality and stability of free software also applies to the central servers that host the website, e-mail, and shared storage on most campuses. A majority of web servers on the Internet use the open-source Apache HTTP Server because of its stability and robustness, and there is no reason universities should not do so as well.

Campus IT departments, however, will tend to resist a move to servers running only free software, largely because they want to be able to pass the buck rather than assume responsibility when an IT problem arises. It is standard procedure in nonfree software environments for IT departments to explain during crises that there is nothing to be done because they "called it in to the [provider's] help line." As we have just seen, however, free server software is widely used and extremely robust, so there are likely to be fewer problems than with closed-source software, and the backup and recovery methods are more widely discussed and implemented.

Furthermore, most IT departments run their campus networks with total control. When faculty from different institutions get together to talk about their respective campus IT environments, they often complain about what is permitted and what is forbidden on university computers. Campus chief information officers should not be setting policy related to software that supports pedagogy and scholarship, although they do on most campuses. IT departments can let go of this power, and this responsibility, instead providing basic infrastructure and advice on particular needs. Networks that are decentralized and not monocultural are far more robust, leaving IT departments with a smaller domain to worry about and more successes to brag about.

I am suggesting that campus IT departments see themselves as "guides by the side," not "sages on the stage," to borrow some current terms from discussions of pedagogy. Faculty members know the best tools for their scholarship and pedagogy, while IT departments understand infrastructure best. Chief information officers can make infrastructure their fundamental job and leave all decisions about software to faculty and staff.

5.4. **Students and administrators.** Fortunately, it will be a cinch to get the majority of students involved in a transition to free software: the challenge of learning a different interface for free software is far less daunting for members of "Generation Y" than it is for members of older generations, since a host of different digital media – all with their own interfaces – has always been a part of their lives.

Administrators will appreciate that free software costs very little. They will also be nervous about not having the help line of a corporate vendor to call in case of emergency. However, many companies sell service contracts for free software, and such contracts can be purchased for a few years to ease the transition to this new world.

6. Allies and Partners

It makes sense for college and university faculty to ally with the free and open-source software community. They share common values. A marvelous additional benefit is that free software on our campuses would significantly advance our pedagogy and scholarship, increase our efficiency, and save us money. Only unquestioning obedience to market fundamentalism – or simple ignorance of the existence and quality of free software – could prevent universities from reaping the benefits of this partnership.

The use of open-source software on campus is a matter of freedom, with a bearing on academic freedom because what is at stake is the freedom to determine how information technology aids our scholarship. Campus IT should not be left in the hands of commercial interests that view our teaching and research only through the prism of the profit motive, which is at odds with the aspirations of higher education and scholarship, when instead a vibrant and effective alternative exists: free software.

Author Bio: Jonathan A. Poritz is associate professor of mathematics at Colorado State University-Pueblo. He previously learned, taught, and used mathematics and computer science at nearly a dozen other universities and IT companies. His website is `http://poritz.net/jonathan` and his e-mail address is `jonathan.poritz@gmail.com`.