# Software Freedom, BSD Unix, and the Liberal Arts
## An Unconventional Celts Computing Conversation with Dr. Corey Stephan

October 24, 2023
12:50–1:50 PM, CSHP 243

Hello, everyone. My name is Dr. Stephan, and I work here at the University of St. Thomas as Assistant Professor of Theology and Fellow of the Core. I exclusively teach Core Theology classes, including Faith, Reason, and Revelation (THEOC 1301) and The Return to God (THEOC 3301), the latter of which I developed and piloted and in which I hope to have all of you as my students. Thank you, members of the Celts Computing Club, for hosting me today, and thank you, Dr. Carlos Monroy, for inviting me. We planned this meeting on late notice, so what I have prepared is rather like stereotypically sloppy Python – a set of imported frameworks hastily glued together rather than neatly coherent code. In other words, this short speech is a set of remarks that I have written or spoken in various places before, hastily glued together and then only gently smoothed. Yet, I trust that you will find them engaging, all the same. My objective is to open multiple doors for conversation. I plan to leave a solid block of time for us to talk freely. Please jot down your questions, comments, and/or snide remarks.

Additionally, I wish to note that this is more a philosophical discussion than the technical and/or professional kind of discussion that you are likely accustomed to having in this Club. I encourage you to listen with the same critical open-mindedness, that is,

intellectual curiosity, that I would ask you to have in one of my classes.

Alright. Let us begin.

[PAUSE]

In May of 2020, when COVID-19 lockdowns abounded and remote learning via the World Wide Web was skyrocketing in popularity, Zoë Kooyman, Program Manager for the Free Software Foundation, posted an article titled "Remote education does not require giving up rights to freedom and privacy." Yet, at many colleges and universities, including our own, proprietary (nonfree) software is the norm not only for remote learning but, actually, for all networking and communications: Microsoft Teams, Google Docs, Blackboard, Desire2Learn (D2L), Zoom, the Respondus Lockdown Browser (which is straightforwardly malware), and a long list of names that are familiar for anyone involved in higher education. Those of us who work at Catholic institutions are as guilty of breaching our students' software freedom as our peers elsewhere, since we rely on the same products.

Software of that kind is definitionally not in keeping with the Free Software Foundation's "Free Software Definition," including the core "four essential freedoms":

Freedom 0: The freedom to run the program as you wish, for any purpose.

Freedom 1: The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.

Freedom 2: The freedom to redistribute copies so you can help others.

Freedom 3: The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

*https://www.gnu.org/philosophy/free-sw.html*

Perhaps most disconcertingly, the freedom that is "a precondition" for freedoms 1 and 3, preceding even freedom 0—namely, "access to the **source code**"—is nowhere to be found. Rephrased, since the software applications that all of us here are required to use are **open source**, they lack the most basic or fundamental software freedom.

Using software that is not open source for higher learning causes at least three major problems:

First, **reduced security**. Cloud-based proprietary services for learning store sensitive information in off-campus server clusters, sometimes running multiple layers of closed source software (e.g. a learning management system [LMS] installed on Windows Server in a virtual machine that is managed by VMWare).

For an example of a popular closed source application that is notoriously insecure, -- in this specific case, routing personal data via proprietary code through servers behind China's Great Firewall -- see Zoom, the darling of these COVID years. in open source projects, security holes tend to be found and closed more rapidly than holes in their closed source counterparts; it is always better to have more eyes looking at the code. That is part of why OpenBSD's developers are able to boast at the top of their home page (and rightfully so): "Only two remote holes in the default install, in a heck of a long time!"

Second, **lack of privacy**. Proprietary software applications for communication can (and sometimes do) include spyware and other malware. The preponderance of sensitive materials that we handle at universities – including all of your names and class grades – makes institutional spyware particularly concerning. Without access to the source code, spyware cannot be identified and refuted. One must run the spyware without any way of knowing that he/she is running it, except when it manifests itself plainly. One such manifestation comes to me when Microsoft issues its automatic monthly report summarizing every virtual interaction that I have had in my institution's digital sphere. ("MyAnalytics" can be disabled, but the personal data collection and storage that enables the creation of the reports cannot. Thus, it might be best to leave the reports themselves enabled in order to have some sense of what Big Brother is tracking.)

Third, **hindered learning**. If students and faculty cannot "study how [a] program works" (freedom 1), then they cannot learn from it. Paradoxically, you, computer science students, are required to use long lists of software—written in thousands upon thousands of lines of code—from which you cannot learn anything about coding. It would be as if my students in theology were unable to read or study any primary literature. I do not know how you learn what you need to learn when the powers that be refuse to allow you to access the materials that you need to learn what you need to learn.

Now, it might not seem that any of this has anything to do with Catholicity. After all, the Catholic Church does not (to the best of my knowledge) have an established doctrine about software ethics. A subject as rapidly changing as computer programming is a poor candidate for permanent direction from a stable Mother who evolves century to century rather than week to week.

, we must not forget Canons 793 through 795, that is, a few key entries in the Code of Canon Law, which famously summarize our responsibilities as Catholics with regard to education. The spirit that lies behind (or the link between) those Canons is that a Catholic education should be accessible for as many students as reasonably possible.

Free and open source software provides one clear way to assist us in achieving that lofty goal—even if modest (compared to the great hurdle of tuition, for an example with which I am sure to win favor from you students). Not only *free* software it as in (the exact meaning of in the phrase "free software," often said in the French ), but it also generally is as in . With a legal requirement from Mother Church, in Canon 794 §2, for "pastors of souls" to make "all possible arrangements so that all the faithful may avail themselves of a catholic education," it becomes difficult to justify requiring any software program that is not able to be run on a diversity of devices and operating systems but, rather, requires an expensive, closed environment. Any program that is only capable of running in Windows 11 64-bit on new hardware rather than being able to run natively also in GNU/Linux and the BSDs on second-hand computers (still sometimes 32-bit) and/or low cost ARM-based systems is **not** *truly* accessible. When I look around my classrooms here at St. Thomas, I see students using hand-me-down laptops from older relatives. Nothing that a student *needs* to do in the computer space in order to learn computing as such requires anything more than such older hardware, just as nothing that a student *needs* to do in the theology space in order to learn theologizing as such requires anything more than low cost, readily accessible scholarly materials, such as the $30 Study Bibles that I require all of my students to purchase.

There is also something to be said for the permanence of open standards. Part of why scholars at the Vatican Library chose the Flexible Image Transport System (FITS), an open standard that was developed by NASA for space imaging, for the gargantuan project of digitally preserving over 80,000 manuscripts is precisely that, as an open standard, it never will go obsolete. To paint the matter as starkly as possible (hyperbolically, no doubt), whereas a .docx file (a word processing file saved in Microsoft's current proprietary format) might not be accessible in 50 or 100 years, a .odt file (a word processing file saved according to the OpenDocument Format standard) surely will be readable at that time. If you, as a 20-something student, should write an essay in one of my classes on Mark's Gospel now, in 2023, you ought to have no doubt that you will be able to reopen that document when you are in your 80s—say, in 2083—without difficulty.

We, faculty and administrators at Catholic institutions, have a responsibility—perhaps, I dare suggest, a moral imperative—to employ free and open source software. That responsibility becomes particularly clear during a time when we are all involved in remote instruction in at least some capacity. Now, in the wake of lockdowns, with the possibility of more coming, we have a unique opportunity to reevaluate our software choices. Let us not allow that opportunity to be wasted. Moving forward, I contend that we ought to require our students to use free and open source software.

We can start by encouraging colleagues and students to switch to free and open source software for communication, document sharing, word processing, and other quotidian tasks. Let us move from Zoom to Jitsi Meet, Teamspeak to Mumble, Microsoft Teams or Slack to Rocket.Chat or Riot.im, and Microsoft Office 365 or Google Docs to LibreOffice (which is now available for online collaboration as LibreOffice Online). For a fuller list of free software for remote communication, see LibrePlanet's "Remote Communication" wiki entry. TechRepublic has a short list of general productivity tools that are free and open source, and web searches reveal far more. I myself have a "Statement on Software Freedom" at the end of all of my course syllabi in which I explain that I never will require any non-free software of my students that is not strictly required by the University for the sake of class organization and record keeping, and I actively encourage students to learn, at a minimum, LibreOffice, LaTeX, Zotero, and so on.

In addition, we can avoid invasive and otherwise problematic means of assessment. If a particular type of assessment only works with the use of Proctorio, the Respondus Lockdown Browser, or another piece of malware whose sole purpose is to spy on students while they work in their own homes, then the powers that be ought not require or encourage such an assessment. Drew Harwell of the Washington Post paints a sobering picture of how invasive remote proctoring software can be, and students throughout the United States have been signing large-scale petitions against it. Essays, open book exams,

and oral exams by video chat are three alternative methods of remote assessment that respect software freedom. With this in mind, I conduct face-to-face oral exams rather than creepy, bogus spyware exams.

[PAUSE]

*Alright, Dr. Stephan*, I am sure that you are thinking, *this is all fine, I suppose, but how on Earth did you get interested in these things? What do you do with them?*

When I was a freshman or sophomore in high school, sometime between late 2008 and early 2009, I had my first laptop computer, and it was *slow*. It had 512 megabytes of RAM, which to an old-timer like Dr. Monroy might sound like a lot, but even for that time, when most of you students were about 4 or 5 years old, was quite low. The standard was two or four GB. That bugger came with Microsoft Windows Vista, and eventually, it occurred to me that Vista itself might be what was causing the overall sluggishness of my system. I was looking for something that would be more efficient with more workflow customizability. I was only doing high school work things, but even then, I did not appreciate being limited in my computing by the arbitrary decisions of some suits in Redmond. Accordingly, I did what any millennial does when one of us wants to find an answer to something with regard to computers. This was before I discovered the principles of software freedom, and I was still using Google rather than an usable search engine, so I did a Google search, and I discovered Ubuntu GNU/Linux. I instantly fell in love with almost everything about it.

After that, I went back and forth with different software. I am embarrassed to admit that I was an Apple guy for several years. Of course, Macs became worse and worse over time, too. Around 2009-2010, believe it or not, Mac OS X was still more-or-less recognizably Unix. A few years later, it was no longer technically possible to apply any changes to OS X's layout. The folks at Apple thought that they knew better how I should work than I do. Dissatisfied, I paused and thought: *I already did some of this legwork with GNU/Linux in high school, so let me pick this up again*. In a way, then, I suppose that it was Apple's descent into the madness of maintaining a default policy of refusing to let its end users own the hardware and software that they have purchased that is responsible for my final, wholesale conversion to free software zealotry.

Bit by bit, I went from casually using Manjaro GNU/Linux's GNOME edition to reading every word on every page of Michael Lucas's glorious 600 page *Absolute FreeBSD*, 3rd edition and writing a formal review of it for the FreeBSD Journal. That is a great book, by the way, for anyone interested in UNIX-like operating systems. It made me better at FreeBSD, opened the door to me to understand OpenBSD, and improved my efficiency quite dramatically in GNU/Linux.

After working through that book, and doing lots of other self-study, I wrote my own dot files from scratch. I call them "theological-dots." They are designed and tailored for efficient multi-source research and writing in *BSD or GNU/Linux. In their FAQ section on GitHub, one of the imagined questions that I ask is this: "Why are these called

theological dots?" Well, they not blessed by a priest or any such thing. They are simply a theologian's dotfiles.

*What other notable computer things have I done over the years?*

In each of 10 summers, I co-taught Build Your Own Computer Camp for middle school students at my alma mater, Marshall School in Duluth, Minnesota. There, I taught students how to install and use GNU/Linux. They loved being able to play Minetest, an open source clone of Minecraft that is, of course, even more scriptable, against each other, as well as doing things like theming their desktops. The absolute customizability of free software makes it tremendously useful for teaching middle schoolers how to compute through what they think is simply playtime. Learning is, after all, supposed to be both fun and enriching.

I love computer building, and I used to use a home-built desktop computer called the Mace Windu for most of my work. After making my recent cross-country move from Wisconsin to Texas, however, I (most sadly) no longer have the Mace Windu. Instead, I use a docked ThinkPad T590 with Debian while at my desk and a freestanding ThinkPad X270 with OpenBSD while away from my desk. On my website, coreystephan.com, I have a blog post in which I document how I have OpenBSD configured on the X270; this post has been shared somewhat widely across the Web, especially in IRC (which, before you ask, is the only social media that I use; you may

find me with the handle "coreystephanphd," all lowercase without spaces or any such thing, in Libera Chat).

I use a wide range of open source tools in parallel. I have a database of ancient Greek texts and a Bible study tool with six texts open in parallel, Zotero for reference management, and LibreOffice with a bunch of extensions. The best of those is a little-known but powerful ancient Greek extension for LibreOffice. It does spell checking even on Byzantine Greek that is astonishingly accurate.

Additionally, I love exploring things -- everything. That is probably a trait that all scholars share in common. We are nerds. As I often say to my students: *H*