RICHARD STALLMAN

TALKING TO THE MAILMAN

Interview by Rob Lucas

You're widely recognized as the world's leading campaigner for software freedom, having led the development of the GNU operating system. Today, the GNU/Linux operating system, and free (libre) software more broadly, underpin much of the internet, yet new structures have emerged that can wield a great deal of power over users. We'd like to talk to you about the present computing landscape and the political relevance of free software. But may we start by asking about your formation, as a programmer and as a thinker—how did it all begin?

I grew up in Manhattan, born in 1953. I was a behavioural problem—I couldn't go to a public school without getting in trouble—and started working with computers at an early age. In 1969, during my last year of high school, an IBM lab let me come and use their computers. In 1970 I had a summer job there. They gave me a project to do, implementing a certain algorithm to see how well it would work. I finished that in a few weeks, so they let me spend the rest of the summer being paid to write whatever I felt like. I went to Harvard to study physics, and carried on programming there. Towards the end of my first year I started visiting computer labs to look at their manuals, to see how the computers differed. When I visited the Artificial Intelligence Lab at MIT, they didn't have much by way of a manual, because they had developed their own time-sharing system. The administrator there decided to hire me more or less straight away. So although I graduated from Harvard in 1974, I had actually been an employee at

MIT for three years. Harvard's computer was a lot better to play with than IBM's, but it didn't have a lot of memory, whereas MIT's computer at the AI Lab had plenty. Not only that, they let me change the time-sharing system; in fact, that was my job—they hired me to work on that system. I added lots of features to lots of different programs—whatever I thought of, or people suggested to me, that seemed like a good idea, I would implement and then people would use it. And this was absolutely delightful—and gratifying to make things that people used and appreciated—so I kept working there. From that point on, I did programming using the machine at MIT.

Were there any people at MIT who were influential in how you learned programming?

There was Richard Greenblatt, who later started the Lisp Machine project; to some extent there was Don Eastlake. And Bill Gosper, although he was more of an inspiration in terms of hacking and math than in how to program. There were a lot of smart people there. But also I was inspired by the attitude at the MIT AI Lab, where the hackers said: 'We're not going to let the administrators tell us how to do things; we're going to work on what they need, but *we* will decide how; and we won't let them implement computer security to restrict us with.' This was a conscious decision of the hackers who had written the time-sharing system, which they'd started a couple of years before I got there. Their attitude was, yes, the administrators could fire us, but we were not going to suck up to them. They weren't going to stand being treated like ordinary employees. I wouldn't have had the strength to do this on my own, but as part of a team, I learned it. We were the best, and most of us weren't getting paid an awful lot—any of us could have got a much better-paying job someplace else if we'd wanted. We were there because we were free to improve the system and do useful things, the way we wanted to, and not be treated like people who had to obey all the time.

And this had the consent of the AI Lab's directors?

To some extent. The Lab's leaders, Marvin Minsky and Patrick Winston, were perfectly content with it. Minsky, I was told, didn't like having doors locked, because he had a tendency to lose his keys. So the doors to the Lab and all the offices inside it were always open. There were no passwords for the time-sharing system. There was no file protection—literally: anybody could sit down at any console and do anything.

To what extent do you think that the collective ethos at the MIT AI Lab back then—which is a striking feature of this history, and is typically cited in the origin stories of free and 'open source' software—was premised on the fact that you were working with a different kind of technology?

It had to do with the fact that we used a shared computer. The PDP-10 was the size of a room and cost a million dollars, so to get another one was not easy. Time-sharing started in the 1960s, but even in 1980 no one thought we could afford a computer for each person—not a real computer. Yeah, there were these toy PCs, but what could you do with those? The point is that when people share a computer, either they do so as a community, where they trust each other and resolve disputes, or it's run like a police state, where there are a few who are the masters, who exercise total power over everyone else.

So you'd agree that the origins of the free-software movement had something to do with the shared use of a computer?

Yes and no. At the MIT AI Lab, the hackers were the authors of the software and were also in charge of the machine. Perhaps guided by the spirit of Marvin Minsky, we developed a culture of welcoming everyone to come and work on everything, and share. So, we resisted security measures. Anyone could look at anyone else's terminal through the system. If you had real work to do, you didn't do that very much, because you were busy. But the kids, teenagers coming in over the internet—Arpanet, as it was then—they would watch, and they would learn things. They would also watch each other and notice if someone was causing harm.

So they could watch people programming?

Yes, they could. Our way of dealing with kids coming in over Arpanet was to socialize them. We all participated in that. For example, there was a command you could type to tell the system to shut down in five minutes. The kids sometimes did that, and when they did we just cancelled the shutdown. They were amazed. They would read about this command and think, surely it's not going to work, and would type it—and get an immediate notification: 'The system is shutting down in five minutes because of . . . '

It sounds like chaos.

Except it wasn't, you see. There was always a real user, who would just cancel the shutdown and say to that person, 'Why did you try to shut the machine down? You know we're here using it. You only do that if there's a good reason.' And the thing is, a lot of those people felt outcast by society—they were geeks; their families and their fellow students didn't understand them; they had nobody. And we welcomed them into the community and invited them to learn and start to do some useful work. It was amazing for them not to be treated as trash.

And these kids were primarily coming in over Arpanet?

A few would come in physically, but most came in over Arpanet.

One thing that's striking about that culture—which is legendary in the history of computing—is that it flourished in an institution largely funded by the American Defense Department. Do you find it paradoxical that this sort of freedom could develop under the carapace of the Pentagon?

It's paradoxical, but it actually makes perfect sense. They wanted to fund some research. They didn't need to make it be done by jerks and downtrodden people—they just wanted it to get done. During the seventies, a number of the hackers at the AI Lab were bothered by the fact that it was funded by the US military. I thought that what mattered was what we were doing, not where the money came from, and at some point I reached a conclusion that funding from business was much worse.

Worse than funding from the military?

Much worse, because the businesses would try to restrict the use of what you did.

Nevertheless, this was the state that was bombing Vietnam.

Yes, it was. I was against the Vietnam War, just like everyone else in the Lab, but we weren't *helping* them bomb Vietnam. Our work wasn't particularly military, or even likely to be used in the short term. For instance, Greenblatt did a lot of work on chess programs; I mostly worked on improving various system programs—I developed the first Emacs text editor during that time.

At what point did that culture change—and how?

Even in the AI Lab, by the late 70s there was pressure from the administration to try to get things under control, which I explicitly resisted. Once I was in the elevator with an administrator who had instituted some forms that every user was supposed to fill out, and I hadn't. He said, 'It seems you haven't filled out the user forms.' I replied something like, 'Yes, I don't see a reason to.' He said, 'Well, you really should fill them out, otherwise somebody might delete your directory, if it isn't clear what it's for.' I said, 'That would be rather a shame, since some of the system source code currently resides in my directory—it would be rather a problem for the Lab if it got deleted.' The thing is, I could do things and he couldn't. But by around 1980, somebody insisted on putting on passwords on the time-sharing machines at the MIT Lab for Computer Science. I'd been getting paid half and half by the AI Lab and the Laboratory for Computer Science, but at that point I switched to just the AI Lab, because I wouldn't do anything for LCS anymore.

Did other people have the same reaction?

Many did, but the others basically got worn down, and eventually gave in. Over the period of the seventies, the spirit of resistance got worn down in others, whereas it got strengthened in me.

How do you explain that?

I don't know. I guess I had found something worth being loyal to, and I basically had nothing else.

It was at this point too that commercial ventures began dividing the programmers at the AI Lab?

It was a process that happened over a few years. Greenblatt started the Lisp Machine project—Lisp was a particularly powerful programming language, in which a program has a simple, natural representation as data that other programs can operate on. Everyone at the Lab agreed it would be great to start a company to make Lisp Machines, so other people could have them, but there was disagreement about how to do it. Greenblatt wanted to start a company without outside investors, but the other hackers questioned his business acumen. So he brought in Russell Noftsker—the man who had hired me in 1971, when he was the administrator of the AI Lab. Noftsker then decided to jettison Greenblatt and start a company with investors, in the usual way. So around 1980 two rival companies got started—the second, Symbolics, by stabbing Greenblatt in the back. I didn't want to join either company—I just wanted to keep working at the AI Lab, which for me was ideal.

Were similar dynamics of commercialization affecting the hacker culture elsewhere, at Stanford for example?

I don't know. I visited Stanford, but I wasn't there enough to know what sort of things were going on.

How did the rivalry of these two companies affect the situation at MIT?

To begin with, both Symbolics and LMI, Greenblatt's company, cooperated in maintaining the MIT Lisp Machine system. But in 1982—in fact on my birthday, March 16—Symbolics announced that MIT could no longer include its changes into the MIT version of the system, which meant everybody had to choose a side: the Symbolics system or the MIT system, used also by LMI. And, as a neutral who had just been attacked and given an ultimatum, I really had no honourable choice except to fight that side. I always thought of this as war, and I understood my part as a rearguard action. My goal was to keep the MIT version of the system viable long enough for LMI to escape being destroyed by Symbolics, which was its goal. They were making all their improvements available to MIT, of course. But I didn't want to read their code and then write something similar, so I looked at their documentation and then implemented a comparable feature—not necessarily the same, because I had a chance to do it better, and often I saw a better way to do it.

Except that there were, what, a dozen of them and one of you?

There were more like six main guys. But, yes, I had to match what they did, so I worked very, very hard. But this showed me that I could do something big like GNU.

Right. So that was an education, in a sense?

It was. It was an education in concentrating very hard on getting software features working—making them as clean as I could, and getting them working reasonably soon. Anyway, they couldn't destroy LMI this way, but they could work on another generation of computer, which LMI couldn't. And in 1983 their computers, their newer versions, started to arrive, and the MIT system couldn't run on them. I saw that I would be unable to continue what I was doing. But I also saw that I had done enough—it had been a successful delaying action; it had enabled LMI to get going and start developing another model of computer, and hire some people to do the software for it. So then I decided, I don't want to spend the rest of my life punishing Symbolics for its aggression—I want to try to build a replacement for what they destroyed.

You mean, the culture at the AI Lab?

The culture—but above all the hacker community, and a system that we could work on freely and share. But having a free-software community depends on having a body of free software you can use—and that's why I had to develop another operating system. The only way there would be a free operating system was if somebody wrote one for modern computers—ordinary computers, rather than special ones, like the Lisp Machine. By the 1980s, lots of people were buying IBM PCs. Although they were still weak at that stage, I realized that future PCs would be able to run the system I was going to develop. That meant it was best to make a system compatible with Unix, which gave me the basic parameters of GNU.

It's interesting that you describe it primarily in terms of creating a community, rather than creating the technology.

The community needs the technology—you can't have a community in which software-sharing is your way of life unless you've got free software to do everything. The point is, the software had the purpose of making the community possible. But part of the idea was that I wanted everyone to be part of this community—the aim was to liberate all computer users. At the same time, I was getting one lesson after another in the injustice of non-free software. MIT had bought a new machine which ran Digital's time-sharing system, TWENEX, instead of the one we'd been developing; it had security features that allowed a group of users to seize power over the machine and deny it to others. I saw the repressive rules for student computers introduced at Harvard. I suggested they apportion a computer to each group of students living together, and let them all run it; those who were interested would develop the skill of system administration, and they would all learn to live together as a community by resolving their own disputes. Instead I was told: 'We've signed contracts for proprietary software, which say we're not allowed to let any of the students get at them.' This was a proprietary operating system that made it possible to have programs that people could run but couldn't actually read. It taught me that non-free software was a factor in setting up a police state inside the computer—which at the AI Lab was generally understood wisdom; I wasn't the first to call that 'fascism'. I was also the victim of a non-disclosure agreement, which taught me about the nature of those agreements—that they are a betrayal of the whole world.[1]

The GNU/Linux operating system now constitutes a very significant part of global computing infrastructures, and has countless contributors globally. How did you go about writing GNU at the start—to what extent was it a collective endeavour?

In the early days there were not very many people participating. Gradually, more joined. I tried approaching companies to see if they might fund the effort, but none did. I'd announced the GNU project on Usenet in September 1983—Usenet was a pre-Internet system for net news, set up by AT&T. Initially only one person actually wanted to work, so there were just two of us when we started in January 1984. At that point, I formally quit my job at MIT, but the head of the Lab, Patrick Winston, offered me the use of the Lab's facilities to do the development work, which was helpful. In 1985 we set up the Free Software Foundation and published the GNU Manifesto; that attracted more volunteers. Coming up with replacements for all the components of Unix was a big job, and I had to find people to do each part. This stage, I'd say, finished around 1990, when I got to be a little famous, got an award and got more attention.

This was more or less coterminous with the development of the modern internet, in terms of the publication of Berners-Lee's proposal for the worldwide

[1] See Sam Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software*, Sebastopol, CA 2002, ch. 1 (also available online in an edition revised by Stallman).

web. What explains the attention you were getting at that moment—the attractions for programmers of the free-software movement?

We'd drawn up the GNU General Public Licence, which guarantees end users the freedom to run, study, share and modify the licenced software program. It was the first 'copyleft' licence for general use—meaning that software deriving from that program has to be distributed under the same GPL terms. Most of all, we were producing software that lots of programmers were finding useful because it was more reliable than the commercial proprietary alternatives. Someone did an experiment, testing the various programs—versions of Unix, or our free replacements—to see whether the program would crash. Ours were the most reliable.

Could you explain why they were so reliable?

I guess because we really cared—and we would really fix bugs that the users reported. Not only that: users could send us fixes, as well as just bug reports. We had a policy of thanking users for their bug reports. The main milestone was passed in 1992, when we had a complete system, using the kernel designed by Linus Torvalds. We'd started developing a kernel in 1990, but the design I chose turned into a research project—it took six years to get a test version. So Linux is the kernel that we actually use with GNU, for the most part.

And Torvalds fortuitously used a GNU licence to do it?

Not initially. In 1991, Linux was proprietary. In '92, he re-released it under the GNU GPL.

You persuaded him?

No, I didn't—I had never heard of him at that point. But he had, I believe, seen me give a talk at the Helsinki University of Technology. When he freed the kernel we could use it, calling that combination GNU/Linux.

What's your definition of free software?

Free software is software that respects users' freedom and community. It's not about price. It's *libre*, not gratis. With any program, there are two possibilities: either the users control the program, or the program controls the users. When the users control the program, that's free software—they control the things they do with it, and thus it respects their freedom and their community. If they don't have full control over it, then it's user-subjugating, non-free proprietary software—the program controls the users, and the program's owners control the program, so it becomes an instrument of unjust power for the owner over the users. For the users to have control, they need four specific freedoms—the concrete criteria for free software. Freedom Zero is the freedom to run a program however you want, for whatever purpose you have. Freedom One is being able to study the program's source code and change it so that you can make the program run the way you wish.

What about people who can't do that—ordinary users who aren't programmers?

I don't think that everyone has to learn how to program—not everybody has a talent for it. But they still deserve control over their computing activities. They can only get that through collective control, which implies the right of users to work together to exercise control over what the program does for them. So further freedoms are necessary for collective control. Freedom Two is the freedom to make exact copies of a program and give or sell them to others. Freedom Three is being able to make copies of your modified versions, and give or sell them to others. This makes it possible for users to work together, because one of them can make a modified version of a program and distribute copies to others in the group, and they can make exact copies and pass them on. That's free software—and all software should be free, because the user's freedom should always be respected. Every non-free program is an injustice. The fact that it exists is a social and ethical problem for society; and the goal of the free-software movement is to put an end to that.

You've argued that this foregrounding of freedom radically differentiates your movement from so-called 'open source', which started later.

Open source is an amoral, depoliticized substitute for the free-software movement. It was explicitly started with that intent. It was a reaction campaign, set up in 1998 by Eric Raymond—he'd written 'The Cathedral and the Bazaar'—and others, to counter the support we were getting for software freedom. When it started, Eric Raymond called me to tell me about this new term and asked if I wanted to use it. I said, I'll have to think about it. By the next day I had realized it would be a disaster for us.

It meant disconnecting free software from the idea that users deserve freedom. So I rejected it.

It's one of the ironies of the history of free software that its moment of greatest fame was associated with this term, open source, which you reject.

Because it's not the name of a philosophy—it refers to the software, but not to the users. You'll find lots of cautious, timid organizations that do things that are useful, but they don't dare say: users deserve freedom. Like Creative Commons, which does useful, practical work—namely, preparing licences that respect the freedom to share. But Creative Commons doesn't say that users are *entitled* to the freedom to share; it doesn't say that it's wrong to deny people the freedom to share. It doesn't actively uphold that principle. Of course, it's much easier to be a supporter of open source, because it doesn't commit you to anything. You could spend ten minutes a week doing things that help advance open source, or just say you're a supporter—and you're not a hypocrite, because you can't violate your principles if you haven't stated any. What's significant is that, in their attempt to separate our software from our ideas, they've reduced our ability to win people over by showing what those ideas have achieved. People who don't agree with us—people who think what matters is what's convenient—have a right to present their views. But they did it in a way that misrepresents us, too. And that, I think, is unfair.

In that sense, it really does appear a kind of political recuperation.

I wouldn't use that term—to recuperate means to recover from some sort of illness, and I don't see who's recovered here. I would call it co-optation—they co-opted our work. They did it intentionally, and they succeeded to a large extent. They would have succeeded 100 per cent, except that we fought back.

Could you explain in concrete terms how non-free software is unjust for users?

The mere fact that the users can't add features, can't change features, and can't fix bugs is an injustice. The users of old versions of programs that are no longer supported are effectively compelled to change to a newer version, whether they want to or not. Non-free, proprietary software is also much more likely to be malware—to contain malicious functionalities, of which there are many kinds. Non-free programs can spy on the users, report on them. Many are designed specifically to restrict what users can do—that's their purpose. The proponents of these malicious functionalities have a term for this: they call it 'digital rights management', DRM. That's a propaganda term—I never use it. I call it Digital Restrictions Management.

Then there are backdoors, which means that somebody can send a command remotely to the program and tell it to do something to the user, as Microsoft did when it forced users to upgrade to Windows 10—whether it's upgrading or downgrading is a matter of opinion—and then made it impossible for them to cancel. This apparently was done through commands sent to Windows remotely by Microsoft, which essentially owns those users' computers, because it put a universal backdoor into Windows. Just as a computer is a universal computing engine, because, with the appropriate program, it would do any computation, likewise, these backdoors are universal, because, by forcibly installing the appropriate code, they could forcibly do anything to the user.

Another form of malicious functionality is tying the program to a specific remote server, as has happened with the so-called 'Internet of Things'—which I call the Internet of Stings. If they shut off the server, the product—the refrigerator or the heating system—doesn't work anymore. Sometimes these products have a universal backdoor, and the company can forcibly change the software such that the users can't do certain things anymore, unless they go through an account on the company's server—which means it's tracking them. Anything that you have to make an account on is tracking you.

How would you periodize the development of these malicious functionalities?

In the 1980s, I would say proprietary-software developers had some ethical standards—in general you could count on them not to put anything malicious into a program, and if something of that sort was found it was a real scandal. In the 1990s, Microsoft's Windows operating system did spy on people in some ways: it reported to Microsoft what programs were installed. But there were so many objections that Microsoft had to take it out—and there were also commercial competitors to Windows at that time. Once Microsoft had established an effective monopoly, it felt it had a licence to mistreat users and reinstalled the software. Windows XP had a universal backdoor when it was released in 2001.

Apple and Microsoft—how would you compare them?

Microsoft and Apple have been changing places. For a long time, Microsoft was the main enemy of users' freedom, and then, for the past ten years or so, it's been Apple. When the first iThings came out, around 2007, it was a tremendous advance in contempt for users' freedom because it imposed censorship of applications—you could only install programs approved by Apple. Ironically, Apple has retreated from that a little bit. If a program is written in Swift, you can now install it yourself from source code. So, Apple computers are no longer 100 per cent jails. The tablets too. A jail is a computer in which installation of applications is censored. So Apple introduced the first jail computer with the iPhone. Then Microsoft started making computers that are jails, and now Apple has, you might say, opened a window into the jail—but not the main door. A study of mobile apps found that, on average, each app informed a hundred different sites about the user; the worst one informed 2,000 sites.[2]

What about Google?

Google distributes proprietary software—parts of Android are proprietary—which includes a backdoor. With very few exceptions, all Google services require running non-free software; it's Javascript in a webpage, but it's still non-free software that they insist you run on your computer. And of course Google does other bad things, including collecting lots of data from users. Gmail goes through Google Services, and Google looks at it to try to learn things about people.

Facebook, Instagram, YouTube, Skype?

I have never been a used of Facebook—I call them 'useds', not 'users', because Facebook is using *them*. If people take photos of me, I ask them not to post them on Facebook. Instagram is the same, as far as I'm

[2] Luigi Vigneri et al., 'Taming the Android AppStore: Lightweight Characterization of Android Applications', Eurecom Research Report RR-15-305, 27 April 2015.

concerned, since it's the same company. The bad thing about YouTube is that you have to run non-free software to watch something from the site in the usual way, and it doesn't offer an option to download unless you use specialized software. Skype is designed to be snooped on by Microsoft. But there are free-software alternatives—Linphone, Ekiga, Jitsi. The Free Software Foundation's high-priority projects include developing real-time voice and video chat, as well as a free phone operating system.

How does the periodization of these technological developments relate to that of state surveillance?

I started to be concerned about surveillance in the 1990s, when I found out how portable phones were being tracked and learned about the spying going on through Windows. In the late 1990s, the US law that enabled wire-tapping on telephone exchanges was extended to digital servers. But the big change came after the second September 11 attacks—the 2001 US terror attacks, that is, not the Pinochet coup in Chile.

The NSA was given a huge amount of money for digital surveillance at that point.

Right. The PAT-RIOT Act was proposed then—I always split the acronym like that; I'm not going to attach the word 'patriot' to such an un-American law that explicitly authorizes massive surveillance. When a program spies on its users and sends the data to a US company, under the Act the FBI can collect all that personal data without even going to court. Worse, the data can be used to profile people, and then manipulate them.

Was it from this point that so-called 'big data' really began to develop into something that could be harvested and scanned?

Maybe it was just beginning then, but I think the real collection of lots of information about people, and the warping of technology into a scheme to collect people's data, didn't start until five years or so after that. Of course, there are other forms of surveillance—surveillance is not solely done through non-free software. By rejecting non-free software, which we have plenty of other reasons to do, we block certain paths for surveillance over us, but not all. For instance, the cameras on the street that recognize licence plates, and maybe now faces—we can't block that by insisting on free software on our computers. The only protection against that kind of surveillance is political. We need to demand, and campaign for parties that will protect our privacy from government oppression.

But you think that the question of surveillance intersects with the issue of free software?

They're related issues—I don't know what it means for issues to intersect.

Okay. How is the free-software movement related politically to other issues—does it have any natural allies?

The free-software movement doesn't require you to have any particular political stand on other issues. And the Free Software Foundation doesn't take a position on other political issues, except to defend human rights in computing—because the freedom to control your computing must be regarded as a human right. That also includes not surrendering your computing to anybody else's server, because you can't control how it's done by someone else's computer; those services that offer to do your computing for you are inviting you to give up your freedom. We oppose general surveillance, because that's a violation of basic human rights. But, for instance, there are right-wingers that support the free-software movement. We welcome them. I don't agree with them on other things, but I'm happy to have their support in campaigning for free software.

Basically, free software combines capitalist, socialist and anarchist ideas. The capitalist part is: free software is something businesses can use and develop and sell. The socialist part is: we develop this knowledge, which becomes available to everyone and improves life for everyone. And the anarchist part: you can do what you like with it. I'm not an anarchist—we need a state so we can have a welfare state. I'm not a 'libertarian' in the usual American sense, and I call them rather 'antisocialists' because their main goal is a *laissez-faire, laissez-mourir* economy. People like me are the true libertarians. I supported Bernie Sanders for President—Clinton was too right-wing for me—and the Green Party.

Would it be quite right to say there's no anti-capitalist dynamic to free software? After all, capitalism proper involves excluding most of the population from means of production, and free software makes such means readily available to anyone. Market exchange is another matter, and could also characterize libertarian socialism, for example.

As I understand the term capitalism, it doesn't necessarily mean that there are quasi-monopolies or oligopolies that politically dominate the world. I do condemn the current system of plutocracy very strongly. When I talk about capitalism, I mean private business. There is a difference between the economic system that the US has now and what it had in 1970. There are two different forms of capitalism, you might say—this one I call extreme capitalism, or plutocracy, in which businesses dominate the state. I'm definitely against plutocracy, but I don't wish to identify capitalism with plutocracy, because there are other forms of capitalism that I have seen in my life. Basically, businesses shouldn't decide our laws.

So your political orientation would be to some kind of social democracy with a mixed economy?

Yes. But that's economic; and my political orientation is democracy and human rights. But to have democracy means the people control the state, which means the businesses don't. So, in terms of economy, yes, I favour having private businesses. I don't think state restaurants could have made the meal we just had.

This raises the issue of how free software is related to broader questions of control and ownership.

Control and ownership are not the same. Some people propose a capitalist-market solution to data harvesting and surveillance, which is that people should 'own' their data and be paid a tiny amount for surrendering it to a company.[3] This is a complete red herring and won't solve any practical problems. Another approach, adopted by the new European Data Protection Directive, is to require people to give 'consent' for their data to be collected. That'll help against certain things—namely, data collection by systems that you never knew you were using, or never agreed to use. But it's so easy for them to get formal consent that it's a worthless barrier. Any site that has a Facebook 'like' button, or a Google Analytics tracker, just needs to include in its Terms and Conditions—which nobody reads—a statement saying: 'I consent to my visit being recorded by various other sites that have an agreement with this one.'

[3] See Rob Lucas, 'Xanadu as Phalanstery', NLR 86, March–April 2014.

I call this 'manufacturing consent'. Consent regulations may be useful against Facebook, because it's been tracking visitors to thousands, if not millions, of websites for many years, and hasn't even bothered to get this sort of consent—so Facebook is in trouble if it uses any of this data. But in future this will probably just be a formal restriction, and not a matter of substance. We should not allow any collection of data that goes beyond the minimum that is inescapably necessary for the site to do its main job—and then we should develop technology to reduce that minimum-necessary amount.

Does that imply that, ideally, people would be doing more computation on their own devices?

Yes, people should. People who are not programmers are likely to think, 'Well, I'll never have control—I don't know how to change these programs.' And that may be true, that they'll never directly change programs themselves. But when you're using a free program to do the job, there's a user community, with many users, some of whom are programmers. *They* can fix bugs, *they* can add features—it means the program will never be discontinued, because the user community will be able to fix the problems, and you'll get the benefit—and not only that: the fact that the program is free is a powerful deterrent to malicious functionalities, because the programmers are aware that they have no power over the users. This means they don't get corrupted, the way proprietary developers do.

This brings us back to ownership and control. With free software, you always own your copy, which goes with having control over it. You don't own the program, in an abstract sense—well, if you wrote it, you do. But usually the program was written by others, so you don't; but you do own your copy. Whereas, with proprietary software, the developers normally say that users can *never* own a copy. Proprietary software typically carries what I'd call an anti-socializing contract, usually known as end-user licence agreements, or EULAs. We speak of the socialization of children—teaching them to be nice to other people, help and cooperate with others. Well, this contract does the exact opposite—people have to commit to *not* making copies for other people, *not* lending or giving their copies to others. I've never agreed to such a thing. Sad to say, most people already have, without thinking about what they were ceding.

That suggests there ought to be a kind of imperative for all users to commit acts of civil disobedience?

Absolutely. But, beyond that, governments should pass laws saying that agreements of that nature are invalid—that they have no legal force in this jurisdiction, no matter where they were signed. This applies to non-negotiated contracts, where the terms are just imposed: 'If you want this, agree to the contract'—there's no chance to negotiate, and that's the great abuse. It's different when the parties actually talk about what terms they want; there may be no need to restrict those. But when the powerful bully the weak into agreeing to contracts—we have to reject that strategy on a moral plane. I believe that the freedoms of free software should be inalienable rights of all users of software. But also, perhaps, the freedom to give, lend or share copies of any published work should be an inalienable right. Governments should actively block or prohibit all methods that *deny* people that right. These anti-socializing contracts are one method—they're the legalistic method; they should be rendered legally without force. As for digital restrictions management, that should put you in prison for many years—it should be a felony to make, sell, lease, import products with DRM.

In material terms, the biggest impact that free software has had in the world is its use by the technical community, on servers and by big business . . .

And that is not my goal. I wanted to give freedom to the users, not the companies. Yes, I believe that a company ought to have control of its own computing. It doesn't make the world a better place if Company A's computing is under the control of Company B. But liberating companies from this mistreatment is not my priority. It's humans, it's people, that I want to liberate.

But the question remains: GNU/Linux distributions are widely used for web servers—probably a majority of servers run them. So when people are interacting with a website, they will typically be interacting with free software in some sense, but that doesn't prevent the sort of abuses you've been detailing here—backdoors, data collection.

When you visit a website, you are not using the software that's in the web server—the organization which owns the website is using that software, to talk to you. Computer scientists are accustomed to analysing systems by saying that the thread of execution can move from one computer or process to another. In some scenarios, it makes sense—technically. But I don't think it makes sense for a discussion about ethics, not when the scenario involves entities that can't trust each other and whose interests conflict. The scenario in which I visit your website involves two entities, you and me. The programs in your web server are working for you. The browser and other programs in my laptop are working for me. They are not a single system, they are two systems which communicate. Of course you don't control the software in some other entity's website—why should you? And how could you? You don't, and you shouldn't—but that's fine. If I'm talking with you, I'm not using your brain. But we're talking, and *you're* using your brain.

Certainly, users visiting a website are likely to be interacting with someone else's computer that's running free software, and that doesn't necessarily help those users. Just because the server uses free software, and the company whose server it is can change it, doesn't mean visitors are treated ethically. And whether or not a website is on a computer that is running free software, it may be collecting data about users. Also, it's very likely to be sending non-free software to their browsers, to run on their machines. Many websites do this. That software does the visitor's computing, and the visitor doesn't control it—because it's non-free software, and because it's running directly as it's sent from the server. It's not easy for the user to interpose control over that.

Suppose some of the software in the web server is *not* free. Who does that hurt? The entity whose website it is—so I would urge it to free its websites by switching to free software; that's in its interest. But I have no reason to boycott a website just because there's non-free software running it, just as I have no reason to boycott a restaurant because their cash register has non-free software in it—*I* don't use their cash register.

So how, more broadly, should we address that issue of entities on the internet that may well use free software on their own servers, but interact with their user base in very opaque ways, wielding a lot of power over data collection and so forth?

Collecting data is a separate issue, and it's the responsibility of separate laws. Free software avoids a particular mechanism of injustice—but it's not coterminous with all of ethics for computing. And this shouldn't surprise people; there are various ethical issues in life. If you look at a store, for example, there are different ethical questions that can arise, and you wouldn't expect that one requirement would fix them all. For instance, the store can mistreat the staff. The solution to that might be to encourage unionization, have strict laws to prevent wage theft. Then there's discrimination, which is mistreatment of those who aren't staff but would like to be. There are other issues that affect the customers only—for instance, are the products what they are said to be? A union might ensure the staff get paid, but might not care about cheating the customers. There are multiple ethical issues; and we shouldn't expect that correcting one kind of injustice automatically corrects the other kinds.

Given that that's the case, again, are there related political campaigns that you would support?

Putting an end to massive surveillance that endangers human rights. For this, we need to require that all systems be designed to limit the amount of data they can collect. You see, data, once collected, will be misused. The organization that collects them can misuse them; rogue employees of that organization can misuse them; third parties, crackers, can break into the computer system and steal the data and misuse them; and the state can take them and misuse them. Laws restricting the use of the data, if properly enforced, would limit misuse by the organization collecting the data, and maybe to some extent misuse by the government—though not enough, because the state will usually give itself exceptions to do what it wants. It will write the laws so it's allowed to get those data, and use them to find whistleblowers and dissidents.

On the other hand, if the system is designed not to collect data, then it can't be used for that, unless the state intervenes to alter it. So unless you can be confident that the state will respect democracy—which is hardly the case here—and respect the right to dissent, you should make sure that massive surveillance is not going on. I've proposed various technical approaches for designing systems to collect much less data—you can see them on the GNU website.[4] The basic point is: when there's another Snowden, how do we make sure the state can't find that person?

You would argue for legislative campaigns to restrict this type of surveillance?

[4] See Stallman, 'How Much Surveillance Can Democracy Withstand?', available on gnu.org; originally published in *Wired*, 14 October 2013.

Legislative and juridical—there are people who go to court to try to stop the collection of data, or limit the use of it. My point, though, is that if we really want to secure our privacy, we've got to stop the *collection* of the data. Rules to limit how the collected data may be used may do some good, but they're not very strong protection. The privacy issue is broader—the cameras that recognize licence plates and track cars: this has been used to crack down on dissent. For instance, there was a picket at a UK coal-fired power plant, and suspected protesters were tracked and then stopped on a road by cops, who held them there until the protest was over, not bothering to charge them with anything.[5] It's clear that they were not suspected of any crime—there were no grounds to charge them with anything. It was a digital attack on dissent, surveillance used to sabotage democracy. It would be good to establish an archive of all these incidents.

What's your view of platform cooperativism—the idea of creating cooperative alternatives to things like Uber, for example?

In general I'm in favour, but it doesn't automatically address all the wrongs. Mistreatment of workers may be avoided by having a worker-owned cooperative—the workers will treat themselves as well as is feasible in the circumstances. But that doesn't mean they will treat the customers ethically. What's wrong with Uber? Well, one thing is, it pays drivers peanuts, which is why I call it Guber, and that's a good reason to refuse to use it. But even worse is the way it mistreats customers, making them run non-free software and keeping a database of where each passenger has gone. Uber has actively tried to eliminate all other alternatives by running at a giant loss, undercutting competitors, aiming to drive them all out of business. Now, if that were replaced with a worker-owned cooperative, they might basically keep running it the same way, but with higher pay for the drivers, who are now the owners—which would not make it any more acceptable, in my view. It's not just a matter of labour versus management, or whether a company is mistreating its workers; the rights of the customers are equally important, and just having a worker-owned cooperative will not guarantee that these are respected.

How about a cooperative of users and workers? There's been the idea of Twitter's user base buying it out.

[5] Paul Lewis and Rob Evans, 'Activists repeatedly stopped and searched as police officers "mark" cars', *Guardian*, 25 October 2009.

Maybe. I'm not sure whether it makes sense to do this for a ride company, though. What does it mean for the customers of a ride company to be co-owners? You can't expect every passenger to do that. And if they were co-owners, that wouldn't protect them at all unless they insisted on anonymity, and, at present, most people don't think about it enough to insist.

Perhaps the example of Twitter makes more sense; you could imagine a Twitter that was owned by its users and workers. And it's imaginable that the user community would be able to ensure that, for example, Twitter ran only free software.

Oh, but that's a different question. Of course, they ought to do that for their own control over their operations. Twitter probably *does* run mostly free software already, because that's standard practice in servers. They're probably running the GNU/Linux system, plus a lot of software that they wrote, which is free software in a trivial sense. But that's not what affects justice for the users, as I've explained.

What do you think about the idea of creating socialized alternatives to the data-hoarding tech giants—as mooted in these pages by Evgeny Morozov?[6]

I think it makes little difference whether data are collected by governments or by companies, because either way the data menace whistleblowers and dissidents. Whatever data businesses collect, the state can easily get.

How about the various attempts to create federated alternatives to Facebook and Twitter—social-communication software like GNU social and Mastodon?

Not just attempts—they work. And from what I hear Mastodon, which is a sort of upward-compatible offshoot of GNU social, is getting to be rather popular. However, the difficulty with a social network is its 'network effect'. Suppose Facebook released all its software as free software: you could set up a server doing the exact same things that Facebook's servers do, but you wouldn't initially have any users. The main thing that pressures people to be Facebook useds is that the people they know, or want to communicate with, are useds as well. My ethical analysis of a communication system with a network effect is that the useds of

[6] Evgeny Morozov, 'Socialize the Data Centres!', NLR 91, January–February 2015.

Facebook are victim-coperpetrators. They may recognize that it mistreats its useds, yet they go along and become one because other people are pressuring them to. But, as a result of surrendering, which means they're victims, they then become part of the pressure on others to do the same—a victim-coperpetrator, in other words. This is the same with Skype, and any other such system that mistreats people but which lets them communicate with each other. Normally, if you're using a non-free program, you're the victim of that, and so I wouldn't rebuke you, I would only encourage you to stop. But, where there's a network effect, I would say it's your moral duty to cease pressuring others to use, and be used by, that system.

I've found GNU social close to a replacement for Twitter, whereas Diaspora was more like Facebook.

That's what I gathered. I'm in favour of projects like these, because I know they're useful for other people, but it wouldn't fit my lifestyle. I just use email. You can call me the Mailman—as in *True Names*, Vernor Vinge's science-fiction story.

It's often argued now that attempts to deter American corporations from massive data collection, in the name of privacy or civil liberties, will allow China to win the new 'space race' in machine-learning and artificial intelligence. What's your response to that?

Freedom and democracy are more important than advancing technology. If China and the US are in a race for Orwellian tyranny, I hope the US loses. Indeed, the US should drop out of the race as soon as possible. Our society has been taught to overestimate the importance of 'innovation'. Innovations may be good, and they may be bad. If we let companies decide which innovations we will use, they will choose the ones that give them more of an advantage over us.

A complementary argument is that Silicon Valley may have the advantage over China in terms of innovation, but that the next stage in AI will be about implementation—the sheer amount of data an organization can run through the algorithms, and the scale of its processing power—where China has the advantage of a population four times greater than the US. (This seems to be the claim of Kai-fu Lee in AI Superpowers.*) How do you view these developments? What are the implications in terms of, first, software freedoms, and second, civil liberties?*

Such AIs will work for companies; their use will be to help companies manipulate people better and dominate society more. I think we should restrict the collection of data about people, other than court-designated suspects. If that holds the companies back in developing AI techniques to help them dominate society, so much the better.

Quantum computing is being touted as the next digital breakthrough—variable 'qubits' rather than the zeros and ones of conventional computing—supposedly opening the way to an exponential increase in computational power and machine-learning. Do you anticipate this having any implications for software freedom?

This means that computers could do certain tasks faster (not all tasks). For the most part, that would be nice but would not fundamentally change anything. However, in one specific case, it could do great harm: our current public-key encryption algorithms would be broken. People are trying to develop adequate replacements before quantum computers are big enough to be used for this.

Would the principles of the free-software movement apply to quantum computing, just as to conventional computing?

Certainly. This is a general principle and doesn't depend on details of the computer's operation.

What do you think have been the major victories and setbacks of the free-software movement?

Of victories, having a free operating system—in other words, the existence of GNU/Linux. Before that, it was impossible to do anything on a computer without proprietary software. Internationally, there have been some partial legislative victories; several countries in Latin America have passed laws to move the government towards free software, but they vary in how strong they are. The one in Peru is rather weak. Argentina has not passed one, but some provinces have. Ecuador was the best example of a system designed to cause migration, but the person that Correa put in charge of that agency didn't do his whole job. I think they migrated the public schools, which is very important. A similar thing happened in Venezuela, except that the policy was not as coherently designed, and some medium-level officials went against it, and the ministers were changed over and over, and the activists had trouble getting political support from the ministers to back them up. In India, the state of Kerala migrated some levels of schooling to free software a decade ago—at the time, they couldn't migrate the last years of high school, because those were controlled by a curriculum board. India adopted a central law saying that there had to be a preference for what they call 'open source', but in fact they're using the definition of free software.

What about the US public school system?

It's just all bad. As for setbacks: the three main ones are mobile computing—phones and tablets, designed from the ground up to be non-free. The apps, which tend now to be non-free malware. And the Intel management engine, and more generally the low-level software, which we can't replace, because things just won't allow us to do so.

Can you imagine a situation in which there was no longer any unfree software, or is that essentially utopian? Is there any way to get there from here?

People said that having a free operating system was utopian and impossible. They argued that there was no use even trying, because it was so difficult. But I think that there's a fundamental error in that question, which is that it assumes that giving up would be okay. I don't use non-free software. I don't use the facilities that require users to run non-free software. So, the free software we have is *already* useful—and I'm sure we can achieve a lot more if we try than if we give up. I don't say that free software is more important than defeating plutocracy, or more important than curbing global heating; and I wouldn't try to argue that people should work on one rather than another. But we've got to have people working on this one—and people in the software field can't avoid the issue of free versus proprietary software, freedom-respecting versus freedom-trampling software. We have a responsibility, if we're doing things in the software field, to do it in a way that is ethical. I don't know whether we will ever succeed in liberating everyone, but it's clearly the right direction in which to push.

Previous texts in the 'New Media' strand of this series have been Bhaskar Sunkara, 'Project Jacobin' (NLR 90) and Francis Mulhern, 'A Party of Latecomers' (NLR 93).