LibrePlanet 2022

”Living in freedom with GNU Emacs“

by Protesilaos Stavrou
(https://protesilaos.com)

Hello LibrePlanet! Hello folks! My name is Protesilaos, also known as "Prot". I am joining you today from the mountains of Cyprus. Cyprus is a country in the Eastern Mediterranean Sea.

In this presentation I will explain how GNU Emacs has allowed me to live in greater computing freedom. Most of this talk will cover what Emacs is, why it can help the user live in freedom, and how those relate to my particular case.

There will also be some more theoretical insights. They are not specific to Emacs or computing in general, but will help us reason about liberty both in its personalised experience and community-based actuality.

The text I am reading from is provided to the organisers of this conference. Those interested will be able to retrieve it from the relevant sources.

A brief introduction to Emacs

Let's start with an overview of what GNU Emacs is. At a surface level, it is a capable text editor. One can use it to do programming or write prose. Emacs has everything a user needs to perform these tasks, such as efficient keyboard-driven motions, mouse support, spell checking and code linting, line numbers to show where the focus is on the current file, integration with the underlying operating system, and much more.

One can use Emacs as a generic text editor, if they want. Though Emacs is much more than that. Here are some other highlights that are provided by the official package:

- A thoughtful windowing or multiplexing facility so that the user can have different files or "buffers" displayed on the screen at once.
- A potent file manager or what is technically referred to as "directory editor" (=dired=).
- A comprehensive suite of tools for handling to-do lists, capturing and filing ideas or information, organising a daily agenda, establishing hyperlinks to all sorts of contexts to create a knowledge base or simply retrieve what is needed, and writing potentially technical documents with a lightweight-yet-powerful markup language. This is the much-lauded "Org mode" and its numerous accoutrements.
- A framework to search through lists of candidates and narrow to a given item using simple or advanced pattern matching styles.

The list of features that are built in to Emacs is too long. But that is still not exhaustive, because Emacs is an extensible program. There are literally hundreds of third-party packages which provide additional functionality to fill in practically every niche. For example, there is a superb package which provides a fully fledged interface to the Git version control system (=magit=). Or another nimble tool which centres the contents of the buffer in the window (=olivetti=). I am using Olivetti right now for this presentation together with my own extras.

In short, Emacs is powerful out-of-the-box and it gets even better with additional configuration. All are available under libre terms.

Emacs as a computing environment

Now I want to tell you why I was drawn to Emacs by explaining what underpins all of the aforementioned features.

Packages with advanced functionality are nice to have, but they still do not quite capture the essence of Emacs and what its true value proposition is. Besides, most modern editors have a plugin system to give the user whatever piece of ad-hoc functionality they require. So what's so special about Emacs?

The answer consists in the fact that Emacs is not really a text editor. It is a programmable platform where text editing is one of the main points of interaction.

Emacs is written in a programming language that is a dialect of Lisp. It is called "Emacs Lisp" or "Elisp". The vast majority of the code base is written in Elisp. While this language is also used to apply all user configurations. For the end-user then, there is only one language, one paradigm. In other words, there is no distinction between the built-in code and whatever code is provided by the user: everything is done uniformly.

At the core of Emacs is the capacity to read and run Elisp. This is known as "evaluation". When some Elisp is evaluated, its return value becomes readily available to the environment. There is no need to restart the program: extensibility happens in real-time where the user can discern the effects right away in an interactive fashion.

In this regard, Emacs is a Lisp machine that can be used to execute any sort of program. By "program" I mean anything from a powerhouse like Org or Magit, to small-scale functions that streamline text editing patterns and motions.

Furthermore, Emacs is self-documenting. What this means is that it understands when the value of a variable has changed and will inform the user about it in the relevant Help buffer. Same principle for referencing other functions whose new value is relevant to the matter at hand.

Lastly, Emacs is free software and provides the means to access the entire source code both for the built-in tools as well as any installed packages. This bridges the gap between what the documentation says and what the actual program is doing.

I use Emacs as an integrated computing environment

I switched to Emacs in the summer of 2019, without any knowledge of Lisp and with only a basic understanding of programming. My background is in the humanities and I was not a tech-savvy user until only a few years ago.

I was drawn to Emacs because I recognised the potential of a Lisp machine. What I wanted was to develop a layer of integration between the otherwise disparate tools I was using before for my day-to-day computing. I wanted a consistent theme, precise typography, the same set of motions or patterns of interaction for everything. Moreover, I wanted to be able to draw linkages between the various contexts or interfaces: my email client should talk directly to my task planner and file manager, the configurations that apply to prose should also be usable for programming, and the like.

Emacs makes all of this possible either out-of-the-box or with third-party packages and custom code. If the user is willing to learn some Elisp, the possibilities are virtually endless. Here are some common workflows that can be achieved without too much knowledge:

- Use the completion framework to run an asynchronous search for file contents, put the results in a dedicated buffer and edit them in place. The changes will be propagated to all the affected files.
- Capture the contents of the email you are reading and produce a to-do item out of them. This task can also have a link back to the original message and it may be assigned a scheduled date as well as a deadline, making it show up on the agenda.
- Mark some files in Dired and attach them to an email that is being composed. Dired can mark items one by one or with regular expressions and more advanced commands.
- Record a set of motions (a "keyboard macro") which uses Dired as a starting point, jumps to a given file, makes some changes, returns to the file manager, and repeats the process for the next file.

In all those cases, the user does not need to learn something new. For instance, keyboard macros are the same everywhere. Same principle for all the other pieces of functionality. They can be used on their own, yet also work in concert.

Emacs thus makes it possible to connect the various interfaces without any of the friction that context-switching causes.

This ultimately is about ease-of-use and simplicity at scale. For example, I use my own custom code to do this presentation. It takes a section of the document and narrows the view to give the impression that this is a pre-built slide. It is not. This is ordinary text that I can edit right now. So I am actually using a general purpose and minimalist "focus mode" which is equally useful for presentations, reading, writing, and programming.

Because I do everything inside of Emacs, I only have to implement this once. I don't need to have a bespoke focus mode for my email client, then another for my text editor or text processor, a third for my agenda planner, and so on. Also, I don't need separate tools for writing and presenting my text. It is one and the same. Nice and simple!

Add to this the fact that the underlying configuration is all done in Elisp, which further contributes to the streamlining of the overall result. Whereas before switching to Emacs I had to use different paradigms and/or languages for each application. For instance, Thunderbird and Libreoffice have their own settings menus that do not talk to each other. Mutt has another way of being configured; Vim another one; Tmux yet another; the terminal emulator has its own; and so on for practically every other application.

This is not to say that each of those applications is bad *per se*. What I am suggesting is that their combined use, their gestalt form, is not optimal. The user has to go to great lengths to piece together a system out of all those silos of functionality. And even with great effort and all sorts of possible hacks, the result will still leave something to be desired, as those programs do not share a common platform and do not speak the same language (figuratively and literally).

Whereas with Emacs every new package automatically gets what all other packages already have, such as the same font configurations and theme, a common set of motions and patterns of interaction, and so on. We then get the network effect that engenders emergent workflows. Everything that happens inside of Emacs partakes in the same environment. All contexts are

potentially interconnected and one may draw linkages at will. This does not involve fragile workarounds. Integration and the singular experience of consistency it yields is at the centre of the value proposition of Emacs.

Consistency and autonomy

Emacs is free software in terms of the license attached to it. Though the freedom it bestows upon the end-user is not limited to a legal arrangement or an abstract moral value. There is a practical aspect to it, which is the integrated computing environment I already outlined.

The extensibility of Emacs empowers the user to make their computer do what they actually want for a broad range of tasks. While the uniformity of Elisp lowers the barrier to entry, especially if we contrast it with the multifaceted corpus of knowledge one needs to possess in order to piece together applications with no shared basis.

Which brings me to the theme of this year's LibrePlanet about living in freedom. We want to appreciate liberty in its quotidian manifestation, as an expression of agency in the here-and-now. In this regard, Emacs is a tool that we use to minimise the distance between what we think and what the computer renders possible.

Freedom of software is about ownership of the means of computing. It enables the development of autonomy within this space, in the original sense of the word as "self-rule". We have autonomy when the tools we use do what we want, instead of having their own hardcoded telos or, worse, being employed in the service of an entity that operates outside our control.

In practical terms, I always found the lack of integration between the various applications I was using to be somewhat problematic. I felt like I was not getting the most out of my rights, as I could not make my tools do exactly what I wanted. Consequently, I was experiencing the seemingly paradoxical case where an assortment of free software applications was not leading to the maximisation of my autonomy. The emergent reality was not as good as the sum of the parts. The inconsistencies between those programs, or else their heterogeneity, was not working in my favour. Even though I was getting the maximum freedom on paper, the actual outcome was only one of partial autonomy; an /autonomy manqué/ or else the unfulfilled promise of liberty.

To put it differently, I was being alienated from my own tools and was thus left in a position where I was not truly in charge: a state of heteronomy (rule by another).

With Emacs, I managed to remove this heterogeneity and its resulting heteronomy from my everyday computing. Now I use Emacs for almost everything: to read and write, manage my files, organise my task planner, deal with my email correspondence, handle my music collection and play back tracks, browse the Internet, and so on. All contribute to the singular experience I alluded to earlier. The rare exception is when I have to use a graphical Web browser.

The general insight here is that freedom has two aspects to it: the nominal and the substantive. The former is about the license attached to the code base. While the latter pertains to the code /in vivo/: how it can be used and how it relates to other programs in a given environment or as part of a wider workflow.

I believe this has to make us reason about software freedom in the expanded sense. Consider code correctness, composability, and extensibility as attributes of a program that together contribute to its

emancipatory function in practice. We want the substance, not mere conformity with legal requirements.

Consider the case of a program that comes with no documentation whatsoever and is written in a way that makes the code difficult to read. It has a libre license but is otherwise not helping the user experience autonomy. The inner workings of the program make it hard for the user to understand what is going. In my experience, this is not sufficient for living in freedom, because "living" in freedom involves experiencing it, and "experiencing" is not limited to what a document says about the applicable copyright terms.

What Emacs has taught me indirectly or perhaps inadvertently is that our focus as a community must be the end-user. We do not want to have freed code that is operated by powerless users. Instead, we want to teach the user how they may assume stewardship over their own means of computing. Free software must not be a mere alternative to proprietary code. It must incentivise a shift in attitude towards self-rule. We should strive to educate the user through the design of the program that they can use what is offered to them in a way that makes sense for them and not just the developer or provider. Little by little, step by step, the user will learn to seek freedom in everything and assume the requisite responsibility.

The intent is to liberate the code from the familiar fetters of copyright in order to give power to the user. The code itself is not the final goal. As such, we must not limit our free software contributions to a formulaic =COPYING= file that accompanies some source code. The "take it or leave it" approach does not empower the user or anyhow lacks ambition. It is of paramount importance to make the program as accessible, configurable, and well documented as possible. Then we have both an excellent program and one that is conducive to the cause of freedom.

The ideal must then be to provide points of entry so that all programs can be made to work in tandem. The best outcome is for the user to be in charge of the entirety of their computer and for everything to behave as part of a greater whole.

Freedom as a collective achievement

If we discern the common in the multitude of all freedoms and concomitant rights, we find that freedom is two-fold: (i) the capacity for initiative and (ii) the absence of control. The first requires that the agent can act, while the second entails that the impetus for it is not provided, framed, or otherwise influenced by another agent.

Action necessarily involves an agent and a patient or what we have in grammar as the subject and object of a verb. For our purposes, human action affects other humans or species. Thus what is possible as just action in this case is whatever does not create a form of control and does not deprive others of their initiative. Put differently, freedom has a social aspect to it and, by extension, ethics belong to the field of politics in the broader sense. There is no action in a vacuum and thus no qualitative aspects to be discerned thereof.

Freedom is thus realised through the collective. The individual is but a part of the bigger picture. The realisation of freedom is a function of what the collective enables through the rules it establishes and upholds. These can be laws or norms of conduct with varying scopes of application. Freedom is thus instituted as such: humans enact and substantiate it.

As with the case of the free software program that needs to work for the end-user's autonomy, so freedom in general is not simply what some charter of fundamental rights declares. There has to be

a culture in place. The members of the collective must recognise each other's capacity for initiative and must not try to impose any arbitrary controls over it.

In short, we cannot have freedom simply by stating the rules. We must also educate the people to cherish and champion liberty. Which is also why conferences such as this one are important, as they raise our community's self-awareness while bringing forth ideas and information that are worthy of our common objectives.

We already know about the essence of freedom and its actuality as a collective achievement through our participation in the free software community. We understand that every contribution we make or benefit from is made possible by the work of others. We are aware of the fact that none of us can enjoy computing freedom as a decontextualised individual. No single person holds all the knowledge that goes into a computer and all of its programs. No one person can ever acquire such knowledge that is not communicated in some way, which implies interpersonal affairs or else the presence of the others beside the self.

Ask yourself: "who am I without the community?". The answer is "not much; not enough".

The Emacs community and how I became a contributor to core Emacs

Since my talk involves Emacs, we too have our own community. By and large, it welcomes newcomers and is eager to disseminate whatever knowledge it holds. The most obvious way the community helps the user live in freedom is with the lofty expectations it maintains about the quality of a package. All decent Emacs packages come with detailed documentation and they provide multiple options for the user to configure things to their liking.

The Emacs community recognises that it must not impose controls over the end-user's capacity for initiative. Hence the extensibility and customisability of all the relevant code. The community also understands the notion that freedom unfolds as an intersubjective experience and that it is not sufficient to emancipate the code from legal constraints. As such, Emacs users promote a strong documentation culture, where every Elisp form has to explain in ordinary language what it does and every package must provide instructions on how to use and configure it. These are in addition to the four freedoms which we expect as a minimum.

This is the "Emacs way" of doing things, which extends to the core application. The very design of Emacs as a Lisp machine empowers the end-user, since everything is subject to evaluation anew. For example, I started learning Elisp by writing my little wrapper functions that tweaked how some default motions behaved. I wanted to have a complement to the "move down a line" command where it would automatically move down 15 lines. Emacs can evaluate code live and, as I said, it grants access to the full source code as well as all the relevant documentation. I was thus able to learn how a function is defined and then through trial and error write my first custom Elisp.

I continued tinkering with Emacs to implement various tweaks or micro optimisations that made sense to me. In the process I learnt to use Elisp. Equipped with that newfound knowledge I was able to do practically everything I wanted with Emacs. Over the past 2.5 years I have written thousands of lines of Elisp and have made several contributions to core Emacs, including a pair of comprehensive, highly accessible, and customisable themes called the =modus-themes= (I am using the light variant right now, which is called =modus-operandi=).

To put it in perspective, I learnt more about programming in my initial three months as an Emacs user than what I had gathered in the three years prior as a member of the free software community when I first switched to a GNU/Linux operating system in mid-2016. This, to me, is the best

example of catering to the needs of the end-user. I, who was once not a programmer or tech-savvy user, was gifted everything I needed to learn how to program and to ultimately be in a position of autonomy insofar as computing is concerned. The rest is the hard work I put in by doing things from scratch.

Consider the bigger picture

Living in freedom demands that we broaden the scope beyond the narrow confines of the individual and also transcend the technical requirements of any one legal-institutional architecture to consider the context holistically: the context of how the program is used as part of a workflow; the context in which a user joins our community and wishes to learn from the wealth of knowledge on offer; the context of how that knowledge is conveyed; the context of freed code as instrumental to the lifestyle change that brings about computing autonomy; the context of how our individuality with its subjective conception of freedom is contingent on the collective.

When we do this, when we look at the bigger picture, we understand that what we really want is for everything that affects us to include our involvement and to share our common values. The goal is to live in freedom, not be a cheerleader of freedom from the sidelines.

This means that we cannot enjoy genuine freedom in our own little capsule, because we do not control the factors that contribute to the state of affairs. Private freedom, in a strict sense, is an illusion. Additionally, we cannot really rollover the problem to some other person. That may remove whatever stigma from us but it merely obfuscates the real problem where the cultural-institutional order inhibits our liberty indiscriminately. To remove the stigma is to engage in role-playing, where you claim to be holier than others even though the prevailing conditions remain illiberal.

Living in freedom, be it in the computing space or generally, implies that we cannot afford to make tokenistic decisions. Each of us is free when the community delivers freedom; when the collective institutes freedom as such. Liberty is intersubjective. It will remain incomplete for as long as some are ostensibly more free than others.

In conclusion, there is an anti-imperialist Greek song that was recorded in the 1980s and remains as relevant as ever---alas! I feel it expresses a profound insight in its chorus:

#+begin_quote
I fear everything that will be done for me without me.

---Vasilis Papakonstantinou - Fovamé (I fear)
#+end_quote

[Check the Annex for a faithful translation of its lyrics.]

On this note, I wish to thank you for your attention. I also want to express my gratitude to the organisers and volunteers of this year's edition of LibrePlanet. Goodbye folks!

Annex: translation of "I fear" lyrics

This is not part of my presentation. You might find it interesting and relevant both for our era and ages past.

The original:

#+begin_verse
Μπροστά σου τα φώτα μιας πολιτείας
που περιμένει τις ανασκαφές
Και τα κλουβιά με τα καναρίνια που κοιμούνται βαλμένα στη σειρά
Κι εγώ που δεν έμαθα ακόμα ποιος είμαι
ένας κουρασμένος σκοπός, χωρίς προοπτική
Και συ που σε λίγο θα σβήσεις
ένα από τα φώτα, για να κοιμηθείς με κάποιον που μου μοιάζει
Έτσι που τα σίδερα του κλουβιού
να χαθούν για μια στιγμή, μέσα στο σκοτάδι

Φοβάμαι όλα αυτά που θα γίνουν για μένα χωρίς εμένα

Τα ρούχα μου παλιώσανε και δεν αντέχουν
τρύπες στα γόνατα από τις υποκλίσεις
τσέπες ξηλωμένες απ' τα κέρματα
χαλασμένα φερμουάρ, χάσκουν χρεοκοπία
Το κορμί μου μελανιασμένο
μες το κρύο σαν λάθος που δεν το παραδέχεται κανένας
γυρνάει και ζητά τη ζεστασιά σου

Φοβάμαι όλα αυτά που θα γίνουν για μένα χωρίς εμένα

Τα τσιμέντα σου καινούρια
με έπιπλα λουστραρισμένα
Και μάρμαρα λευκά
μια γυαλάδα που στραβώνει
και δε σ' αφήνει χώρο να σταθείς
και μόνο εγώ απ' όλα εκεί μέσα
σαπίζω σαν σε αρχαίο τάφο
Σκεύη παραστάσεις βρέθηκαν εκεί
εκτός από εμένα, που σε κρύπτη μυστική
ψάχνω ακόμη να σε βρω να με αναστήσεις

Φοβάμαι όλα αυτά που θα γίνουν για μένα χωρίς εμένα

Τα ρούχα μου παλιώσανε και πέφτουν
σαν χρεοκοπημένες κυβερνήσεις
Γέρασα μ' ένα παιδικό παντελονάκι
και το πλοίο δε φάνηκε ακόμη
Σε σφίγγω πιο πολύ γιατί κρυώνω
το κορμί μου δρόμος, που εκτελούνται δημόσια έργα
κομπρεσέρ μ' ανοίγουν και με κλείνουν
Τράβα λίγο τη κουρτίνα να με δεις
έγινα διάδρομος για στρατιωτικά αεροπλάνα

Και το μυαλό μου, αποθήκη, για ραδιενεργά κατάλοιπα
Μέτρα ασφαλείας πήρανε, για την αναπνοή μου
και σε πολυεθνικό μονόδρομο
το μέλλον μου δώσαν αντιπαροχή

Φοβάμαι όλα αυτά που θα γίνουν για μένα χωρίς εμένα

Έτσι ζω προκαταβολικά το παρελθόν μου
και με δυο γυμνά καλώδια για χέρια
αγκαλιάζω τα ψηλά σου volt για στερνή φορά

Φοβάμαι!
#+end_verse

In English:

#+begin_verse
Before you lie the lights of a polity
that awaits the excavations
And the cages with the sleeping canaries lined up
And I who has yet to know who I am
a lost cause, with no perspective
And you who will soon switch off
one of the lights, to sleep with my look-alike
Such that the bars of the cage
vanish for a second, amid the darkness

I fear everything that will be done for me without me

My clothes are worn out and cannot hold together
holes at the knees from the kneeling
pockets ripped by the coins
broken zippers, staring at bankruptcy
My body bruised in the cold
like a mistake that none admits
turns and yearns for your warmth

I fear everything that will be done for me without me

Your cements are new
with polished furniture
And white marble
a blinding shininess
that leaves no room to stand
and only I out of everything there
rot as if in an ancient tomb
Implements of performances found there
except me, who in secret crypt
continue searching for you to resurrect me

I fear everything that will be done for me without me

My aging clothes are falling apart
like bankrupt governments
I grew old wearing a child's pair of pants
yet the ship hasn't appeared yet
I embrace you more tightly as I'm cold
my body is a road where public works are performed
drills open me up and seal me
Pull the curtain slightly to look at me
I've been turned into a runway for military aeroplanes
And my mind, a storage place, for radioactive waste
Security measures have been adopted for my breath
and on a [corporate] multinational one-way street
my future was given away

I fear everything that will be done for me without me

I thus live my past in advance [what has transpired is to be experienced again]
and with two stripped cables for hands
embrace your high voltage for the last time

I fear!
#+end_verse