Emacs: The Editor for the Next Forty Years

or... Some M-x info on Future-Proofing Emacs

Perry E. Metzger
perry@piermont.com

University of Pennsylvania
Department of Computer and Information Science

EmacsConf, November 2, 2019

It's been a while...

I learned Emacs in September, 1983

It is now November, 2019

...that's over thirty six years!

An amazingly long time.

What's this talk about?

We're here spending all day watching talks about a text editor?

Clearly we're all hooked. We think Emacs is a critical tool for our work.

Will future generations of hackers also get hooked on Emacs?

Let's make sure that happens.

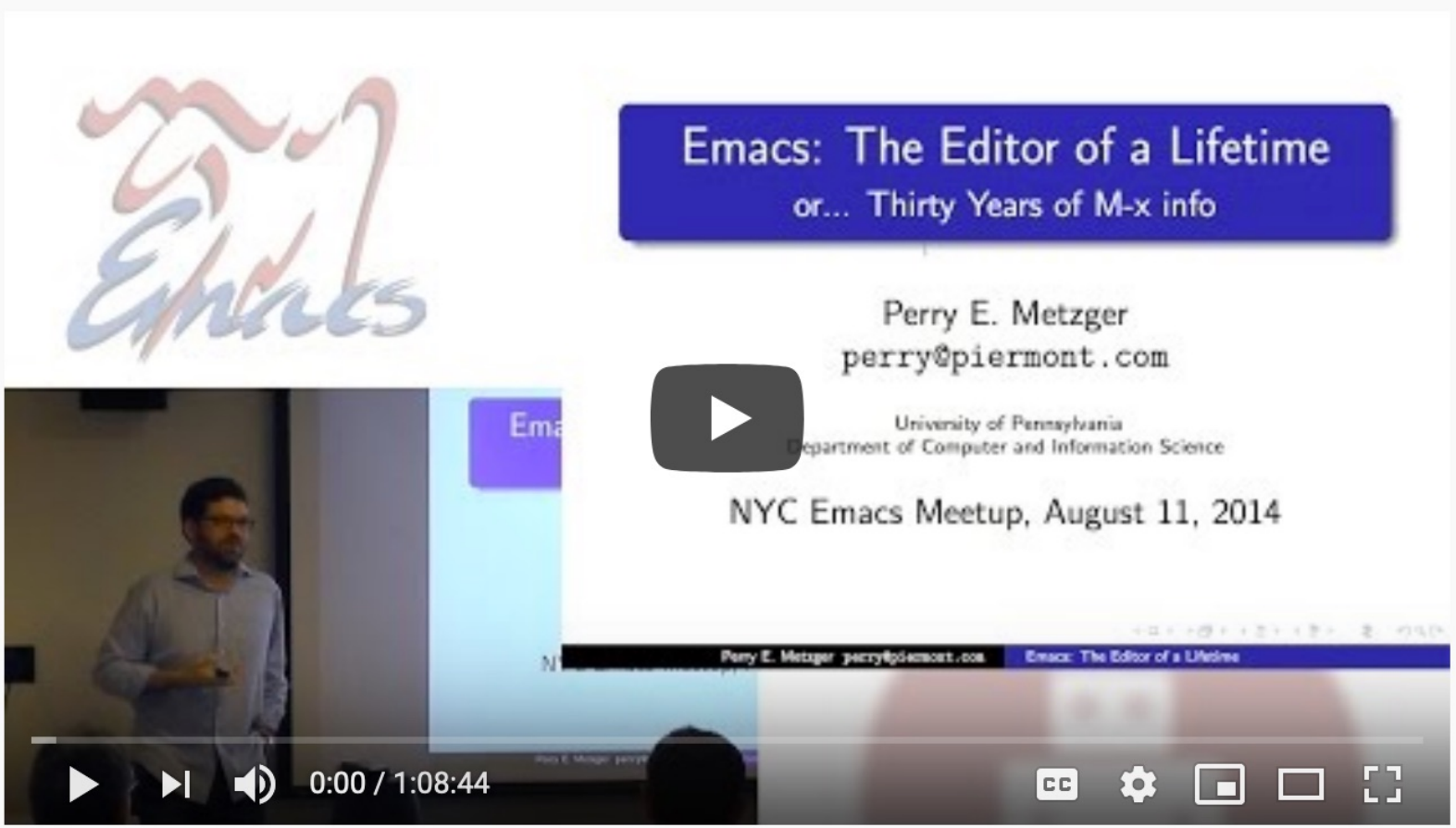
What's this talk about?

This talk is a call to action.

I gave a talk in April, 2013 on the 30th anniversary of my using Emacs.

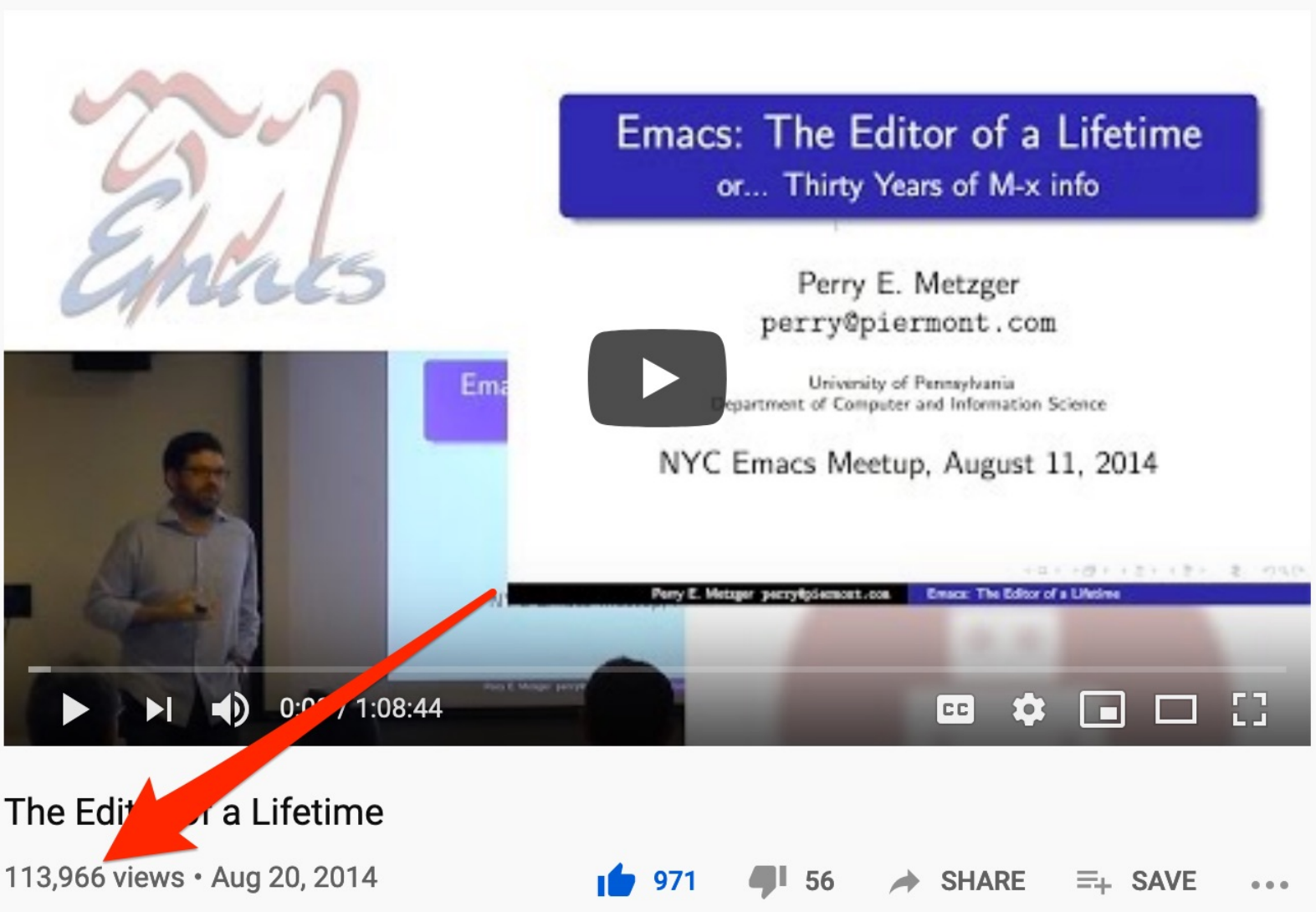
I repeated the talk for the NYC Emacs Club in August, 2014.

It got recorded.



The Editor of a Lifetime

113,966 views • Aug 20, 2014 971 56 SHARE SAVE


Emacs: The Editor of a Lifetime
or... Thirty Years of M-x info
Perry E. Metzger
perry@piermont.com
University of Pennsylvania
Department of Computer and Information Science
NYC Emacs Meetup, August 11, 2014
1:08:44
The Edit a Lifetime
113,966 views • Aug 20, 2014
971
56
SHARE
SAVE

What's this talk about?

The response was a bit surprising to me.

But I think I hit a nerve.

What's this talk about?

I'm not going to repeat all of that talk.

(What would the point be? It's online.)

(If you haven't seen the original talk, you should watch it to get some context. I'm sort of assuming you have.)

But I am going to expand on one section of it.

What's this talk about?

The topic today: can Emacs survive another forty years?

My answer: Yes!...

...but to do that, it needs to remain the best tool for future developers.

So, I'll be discussing what Emacs can do to adapt itself to the requirements of current and future developers.

Emacs evolves.

Emacs is really old...

It was written for a different world.

This was the best way to use Emacs when I started...

Now it is a museum piece.

The OS I used in 1983 was TOPS-20, on a DECSYSTEM-20

I don't miss it.

TECO was problematic...

```
GZOJ\UNQN"E 4OUN ’ BUH BUV HK
QN< J BUQ QN*10/3UI
QI< \+2*10+(QQ*QI)UA B L K QI*2-1UJ QA/QJUQ
QA-(QQ*QJ)-2\ 10@I// -1\%I >
QQ/1OUT QH+QT+48UW QW-58"E 48UW \%V ’ QV"N QV^T ’ QWUV QQ-(QT*10)UH >
QV^T @^A/
/HKEX$$
```

And so Lisp was a big improvement.

What changed?

Emacs *evolved*.

Some of the change (TECO Emacs to GNU Emacs) was abrupt.

Mostly it was incremental.

It *has* adapted...

Emacs has changed in 40+ years.

- Improved extension language.
- GUI window system support
- New modes
- New capabilities (networking)

...but it needs to go further.

Extension language still mediocre.
Implementation language is terrible.
Not enough of an OS! (Threading sucks!)
Not enough of an OS! (I still need to leave it!)
Needs more PIM integration (e.g. protocols).

Emacs is no longer unique.

For the first time, there are reasonable alternatives.

Visual Studio Code has a big following, and an extension language.

Relatively few young hackers are trying Emacs.

Emacs needs to adapt going forward.

Two kinds of changes...

There are two distinct kinds of changes Emacs needs.

Users care about *benefits*, not *infrastructure*.

But to get new important features, you may also need new infrastructure.

I'll be discussing infrastructure first.

The Emacs of Theseus

Famous thought experiment in philosophy: The Ship of Theseus

I don't care about the thought experiment for today.

But it's also a development philosophy.

Revolutions kill, Evolution preserves.

Emacs code base is gigantic.

Org mode alone is 120klocs.

Starting afresh means no users. No users means no developers. No developers means no interesting features. Which means no users...

Instead, fix things *incrementally*.

ELisp is Old!

Some improvements have been made
...lexical scope finally!

But
...API is awful.
...no modules.
...new threading system hackish (and unsafe.)
...not really a good lisp.
...really, really, needs to be replaced.

Solution: Build a better extension language in parallel!

Keep the old, add something new!

Have a *second* extension language that uses the same runtime.

The runtime is okay-ish, and if you share it, you get easy calling between the languages.

Encourage new code to be written in the new (better) language, but the installed base (and the user community) are protected.

What should a new language look like?

This part is more speculative.

(I think) it should be "lisp" so for comfort, to make interoperation natural.

Must provide strong concurrency safety guarantees.

Should be strongly typed. Type systems make development easier.

How to get there?

Experimentation is necessary. Really good languages don't happen all at once.

I also encourage getting PL theory people involved. *They know important things.*

And many of them *love* Emacs.

(Like, really really love it.)

A note on performance...

Emacs has gotten pretty slow of late.

JITting or otherwise compiling our new extension language will help.

(And if it's typed, compiled code will *scream*.)

Portable compilers are now easier than ever.

It's a parallel world.

Single threaded CPU performance has stalled.

Laptops with 8 and more cores common. Desktops with 32 cores are now available. Servers with hundreds of cores appearing.

Prediction: good programming models for parallel / concurrent code critical to future software, including Emacs.

It's a hard problem.

If your Emacs is showing you a tutorial on the web, helping you refactor a big program, and also checking for new mail and your next calendar reminders while running a chat window, it had better deal with concurrency well.

Concurrency is *notoriously* hard for programmers to get right.

Concurrency safety isn't an accident.

So far, *only Rust* has a good story on concurrency thanks to its type system.

Emacs' extension language and internal architecture needs to handle concurrency "trouble free" for the programmer.

That means *careful* design.

Good concurrency model is hard!

Some of the world's best minds have struggled to design good concurrency architectures!

Important to consult theory people, think carefully, work slowly, consider Emacs' special needs.

New extension language, concurrency model, implementation architecture changes need to be considered *together*.

C is dying.

C has over 200 forms of undefined behavior.

C programs *cannot* be written safely by mortals.

C has few features for modern programming

C really has to go.

Is this a crisis?

...not quite yet.

Having a creaky implementation language slows Emacs evolution.

Harder to debug, harder to write good code.

Which means it's harder to contribute to Emacs.

Better to fix slowly while it isn't a crisis yet.

What's a good new language?

My guess: Rust is the only practical option for now.

It's the only fully safe systems programming language in existence.

As much "control" as C has, easy interoperation with C.

Why Rust?

Fully safe type system. Even concurrency is guaranteed safe by the compiler!

Can still write low level unsafe code like garbage collectors.

Modern features (polymorphism, modules, etc.)

But this is a long discussion.

How: Incrementalism again!

My suggestion: *incrementally* replace C.

Provided your language can be called from C easily and can call C easily (Rust can), you can do this in pieces, not all at once.

It should be possible to replace sections of Emacs codebase slowly over a period of years.

C and other languages will remain indefinitely (window toolkits etc.)

Displaying Not-Text

Some features Emacs should have (I'll discuss these soon) demand the ability to render modern web pages *faithfully*.

Web rendering is a specialty. Emacs' developers cannot do that on their own. We need to use other people's libraries (like WebKit.)

(XWidgets seems a bit stillborn.)

Being able to render PDF would also be a good thing.

What does Web + Emacs look like?

User story: see a web page or email message in a buffer, isearch in it, select text with keystrokes, switch to another window (including email being composed in HTML!), paste.

Infrastructure to implement this is deep!

Doing HTML right

Need to be able to view page with or without JavaScript (mail has it turned off), filter which JavaScript (turn off non-libre code, dangerous code, etc.), filter URLs, selectively filter and render images...

Need to be able to walk and modify DOM inside extension language!

And much more! This a big project, but also a killer capability!

Why do we care about this?

Modern email is HTML. I need to see HTML email the way my colleagues see it. I have no choice. It's not 1992 any more.

I develop web pages and want to view my work in real time, as end users will see it.

I want to read programming documentation, which is all on the web, *inside* my editor, and see it formatted and rendered as the authors intended.

Why we care about HTML cont'd

I want to do all my chat inside Emacs inside Libre solutions as good as Slack and Discord, but modern chat systems expect to be able to embed web components (including video etc.)

I hate ever leaving Emacs. I'd like to surf the web in it. But that means good web support inside an Emacs-style interface, plus good concurrency and 20 other things.

That last bit is a very long term goal. If I can read email in Emacs again I'm going to squee very hard...

Also PDF...

Reading documents is something you want to do in your editor...

After all, you may want to make notes...

...and maybe you'd like to search, copy, switch buffers, paste...

Not as critical as web rendering, but it would be Very Nice To Have.

Proportional + Multi-Face Text

Some potential Emacs applications (like chat as good as Discord or Slack) could use "nicely" rendered text (proportional fonts, bold and italic, etc.)

Some writing tasks (documentation, etc.) probably are better done in such environments, too.

This is wildly inappropriate for programming modes, but Emacs isn't just programming.

Features...

Let's talk a bit about missing features in Emacs (and problems)...

Things users want or need built on the infrastructure I discussed.

Better Tools for Programmers

When I gave my last talk, Emacs was falling behind other editors on refactoring, symbol awareness, documentation popups, etc.

But now we have the Language Server Protocol!

LSP solves the $M \times N$ problem for supporting languages in editors.

And Emacs now has it!

Include LSP in Emacs!

LSP support doesn't come "in the box" with Emacs.

Emacs is used extensively by programmers, so Language Server Protocol support should ship with Emacs so that all language mode authors can assume it is present.

Integrate LSP in everything!

Make LSP excellent in Emacs!

Good part: fast JSON parsing is in Emacs 27 already; this is a big win!

Needed: better and better language support. But that seems to be happening! (Which is good news!)

Next big thing: Debug Adapter Protocol, so all modern debuggers can talk to Emacs.

Emacs as OS

When I gave my last talk, Emacs always blocked all windows when computing in any given window.

I still often run multiple Emacsen to get around this! (Boo!)

Thread support can help, but ultimately, a cross-Emacs concurrency model is needed. See above.

Needed if more people are going to truly live in Emacs.

I want to read my email in Emacs.

It's been over a decade now since I have read my email primarily in Emacs. I'm sad about this *almost every single day!*

Need better IMAP-oriented email modes (Mew & Wanderlust aren't well documented + maintained) + HTML email rendering.

If the HTML rendering problem gets fixed, the rest is a Simple Matter of Programming and Documenting. *Let's fix the HTML rendering problem.*

PIM integration

I want calendar modes to do CalDAV.

Contacts should really do CardDAV

Would be nice if underlying protocol tools were abstracted so that multiple modes could take advantage.

Semi-digression: Markdown, Org, etc.

Lots of people now use Markdown all day long for documenting their code.

Should it be possible to view Markdown (or Org) comments and documents both as source and *as rendered* everywhere inside Emacs?

Digression: ORG all the things?

Org mode is addictive to lots of people.

RMS once noted an architectural problem. Org mode is "separate" from Emacs.

Org provides all sorts of facilities you would want all over the place. Should it be pervasively integrated?

Better Chat Clients (and Protocols!)

IRC is getting crusty...

...which is why many are moving to Slack, Discord, etc.

...*which are not Libre and won't run in Emacs!*

Part of this requires good font rendering, and the ability to embed web widgets (like videos.)

But part of it also requires new, open protocols (and no, I don't mean a web site with an API.)

And now, a rant about

Hacker Friendly User Interfaces

UIs for experts.

I use a Mac.

Stop hissing.

Why do I use a Mac?

Part of it is because it supports an Emacser better.

Really!

What???

On MacOS, every input widget accepts Emacs keyboard commands.

No, not if you configure it to do that. *BY DEFAULT!*

By contrast, I've never figured out how to do that on Gnome or KDE.

And I've tried!

What??!

Emacs support *is on by default in MacOS!*

It might not even *exist* in Gnome and KDE

So the soi-disant "geek friendly" OSes are *less* geek friendly!

Stop sacrificing hacker usability!

To win, Libre software needs developers.

Which means it needs users who are hackers, because hackers fix their own tools.

Hackers using a Libre platform means hackers *improving* a Libre platform.

Besides, I'm a Hacker...

A Libre platform needs to be hacker-friendly for me to want it.

Mostly, Libre desktops try to be friendly *only* to normal users.

They're more like Notepad than Emacs. So why would hackers fall in love with them?

I don't use Notepad, do I?

Libre Desktops for Hackers

Maximal productivity for hackers $\neq$ user friendly for normal people.

Productivity means... Emacs like interfaces

Build desktops for geeks! Make Emacs's UI swallow everything.

Everything is "Emacs"

This could be literal (Emacs for email, Emacs for Web, Emacs for PIM, Emacs for Chat, Emacs for... editing.)

Or it can be figurative...

Everything is "Emacs", cont'd

Have all UI text widgets take Emacs commands...

Add "minibuffer" window to window manager...

Make Emacs-like keyboard commands be the best way to do *everything*.

Make entire UI extendable in the extension language...

Hopefully the same one Emacs uses, hint hint.

A deeper digression into keyboards...

Have I mentioned keyboards?

A Rant About Keyboards, Part 1

A Rant About Keyboards, Part 2

A Rant About Keyboards, Part 3

What's my Emacs priority list?

Top near-term priority: HTML rendering

Easy/soon: LSP support right in Emacs.

Second tier: modern email, PIM support, etc.

Long term slog but needs to start soon:
concurrency model and future extension language.

Questions?