How do I edit a file in a Terminal? 🤷

NCML lab meeting (open)

2024-08-21 Seung-Goo KIM

"Terminal" of what?

IBM 9020 equipment at the Jacksonville ARTCC at 1970s
Source: https://www.faa.gov/about/history/historical_perspective (Ch. 4, pp. 19)

http://www.catb.org/esr/writings/taoup/html/ch02s01.html

MAX PLANCK INSTITUTE FOR EMPIRICAL AESTHETICS

The first Unix in 1969, Bell Lab

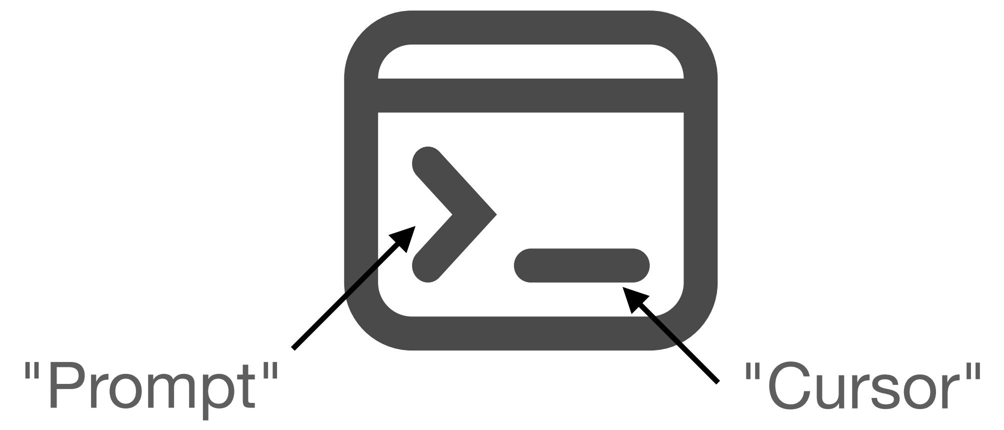
The mighty PDP-7

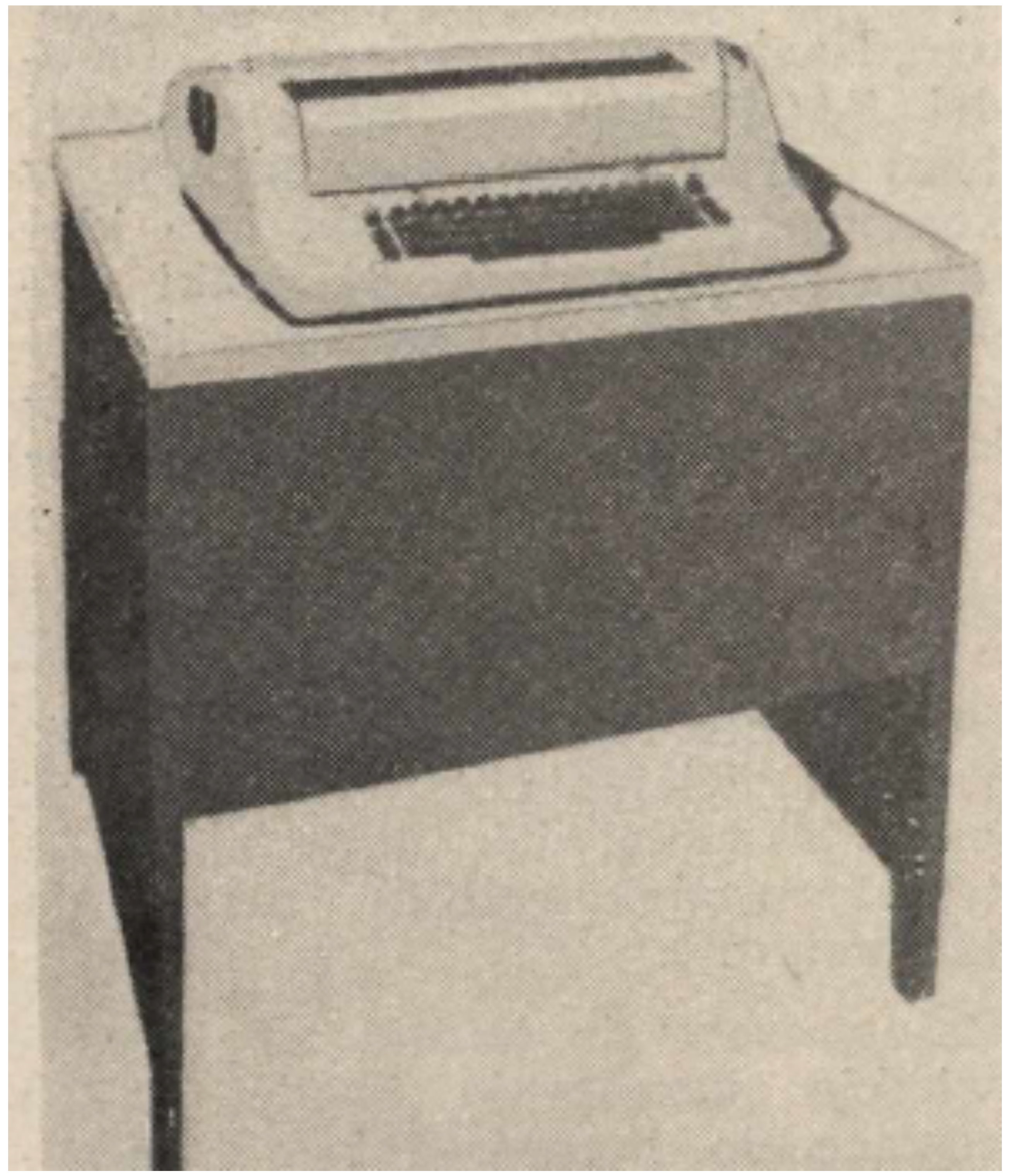
Ken (seated) and Dennis (standing) at a PDP-11 in 1972.

https://en.wikipedia.org/wiki/Computer_terminal

Early "Terminals" and modern emulations

"Command line interface" (CLI)




IBM 2741 (1960s-1970s), "Teleprinters"

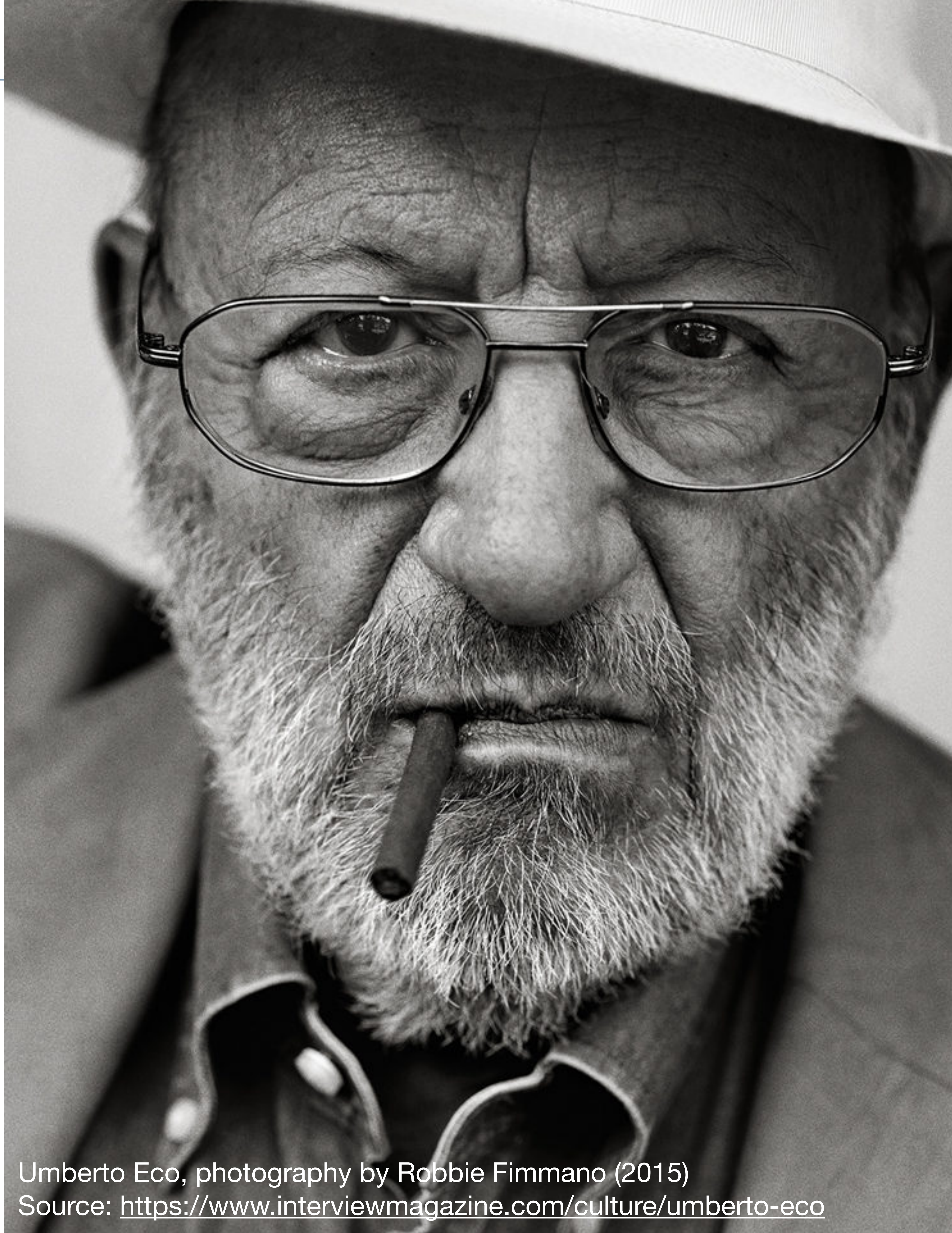
DEC VT100, 1978, the first to support cursor control on display

MacBook Air, 2022

The whole "computer" (CPU+GPU+RAM+SDD) is under the keyboards and it still looks like a "terminal" (keyboard+display)

Why do you want to emulate an ancient relic?

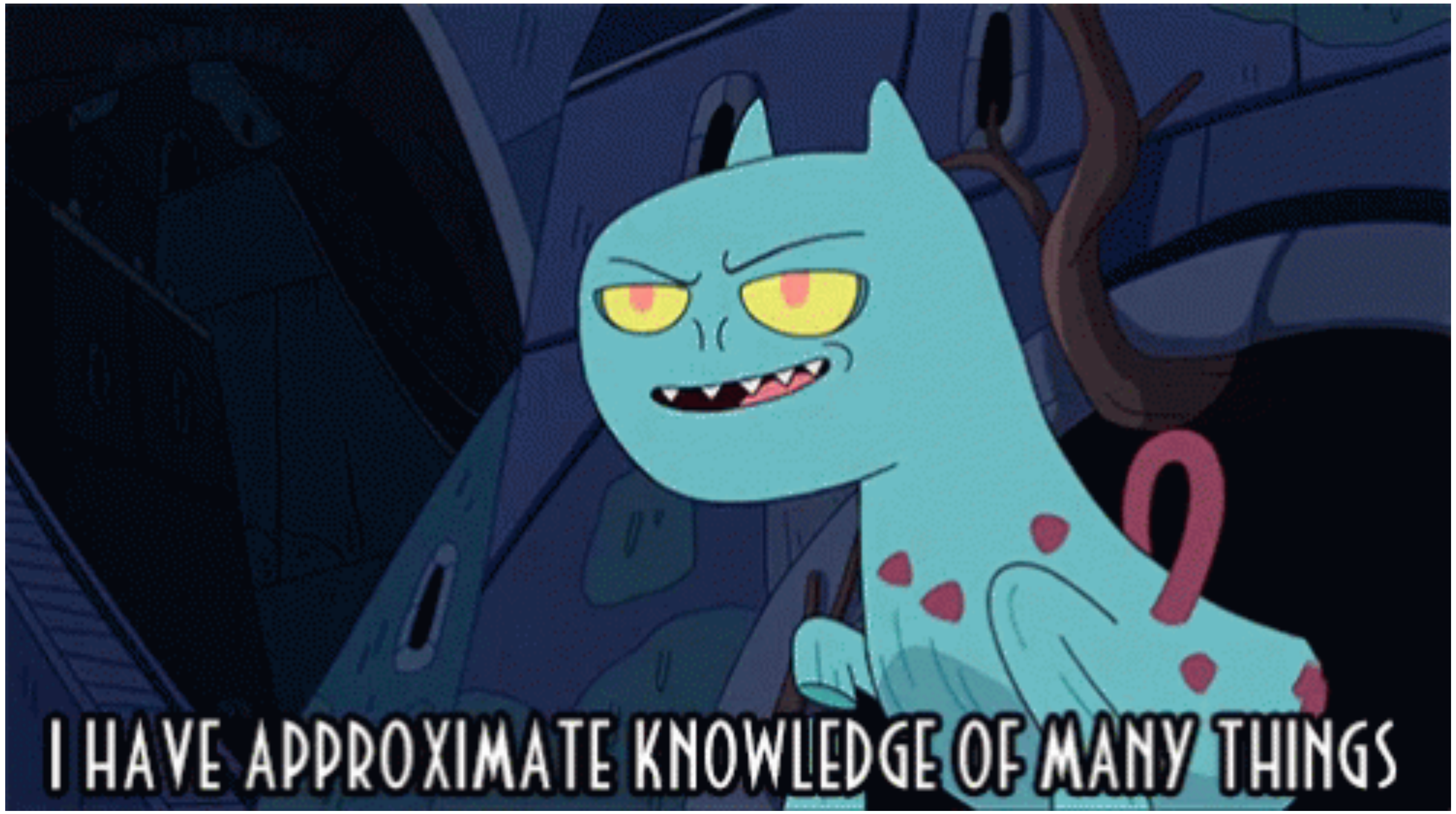
- "The book is like the spoon, scissors, the hammer, the wheel. Once invented, it cannot be improved. You cannot make a spoon that is better than a spoon." (Umberto Eco, Jean-Calude Carriere, 2009, "This is Not the End of the Book", Chapter 1, Northwestern University Press)
- So is the terminal! (Meself)


Umberto Eco, photography by Robbie Fimmano (2015)
Source: https://www.interviewmagazine.com/culture/umberto-eco

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - bash
  - Emacs
  - Vi, Vim, Neovim
  - nano & others

Disclaimer: what I say can be slightly inaccurate.



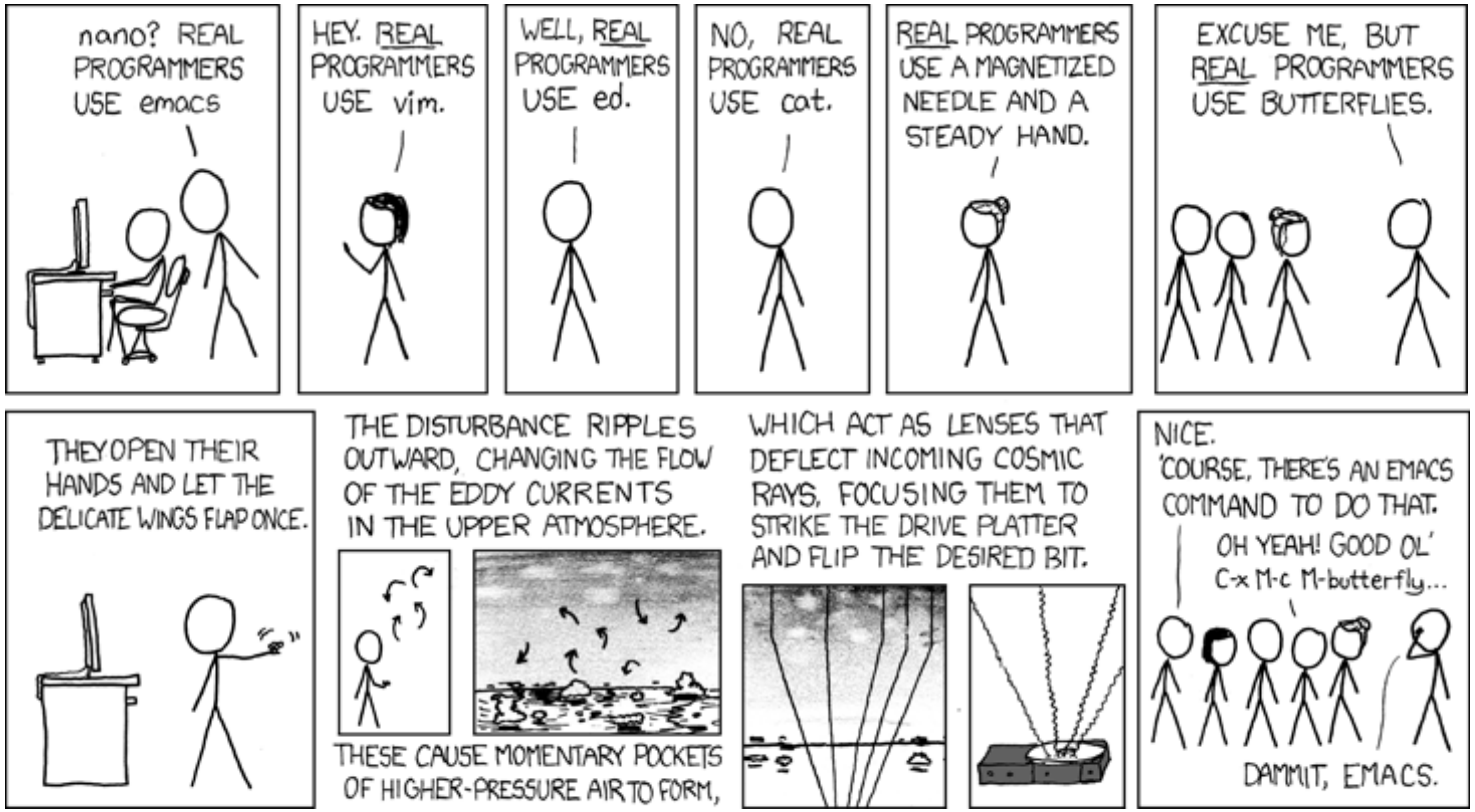
Demon Cat from the episode "Dungeon" of a series *Adventure Time*, (C) Cartoon Network, 2010-2018.

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - bash
  - Emacs
  - Vi, Vim, Neovim
  - nano & others

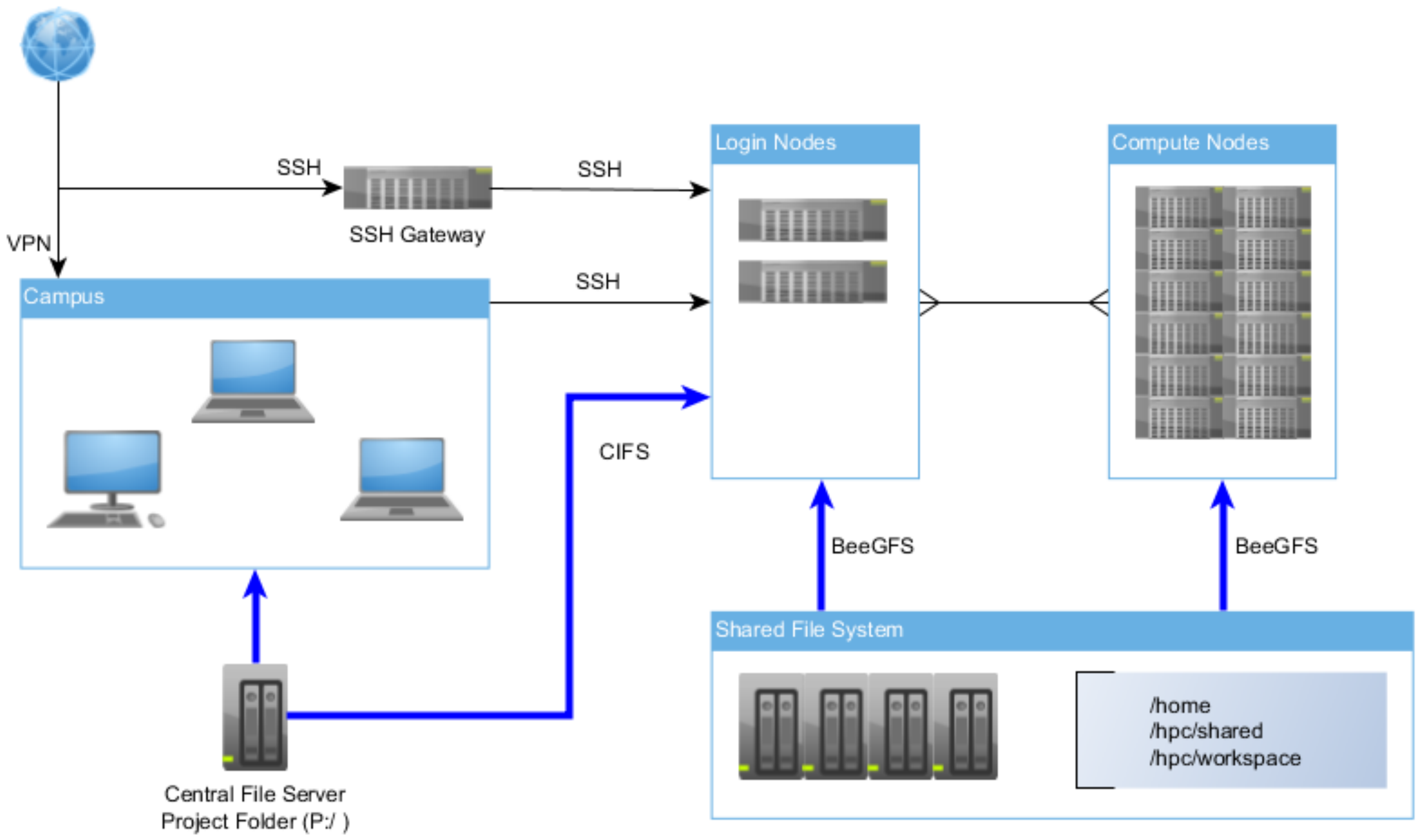
Because that's the way it's meant to be!

(or to show everyone that you're a REAL programmer)


nano? REAL PROGRAMMERS USE emacs
HEY. REAL PROGRAMMERS USE vim.
WELL, REAL PROGRAMMERS USE ed.
NO, REAL PROGRAMMERS USE cat.
REAL PROGRAMMERS USE A MAGNETIZED NEEDLE AND A STEADY HAND.
EXCUSE ME, BUT REAL PROGRAMMERS USE BUTTERFLIES.
THEY OPEN THEIR HANDS AND LET THE DELICATE WINGS FLAP ONCE.
THE DISTURBANCE RIPPLES OUTWARD, CHANGING THE FLOW OF THE EDDY CURRENTS IN THE UPPER ATMOSPHERE.
THESE CAUSE MOMENTARY POCKETS OF HIGHER-PRESSURE AIR TO FORM,
WHICH ACT AS LENSES THAT DEFLECT INCOMING COSMIC RAYS, FOCUSING THEM TO STRIKE THE DRIVE PLATTER AND FLIP THE DESIRED BIT.
NICE.
'COURSE, THERE'S AN EMACS COMMAND TO DO THAT.
OH YEAH! GOOD OL' C-x M-c M-butterfly...
DAMMIT, EMACS.

https://xkcd.com/378/

Or, because we are still using large computers!



How do we use those large computers?

We need to connect to the large computer via network

	Virtual Network Computing (VNC) — GUI	Secure Shell (SSH) — CLI
Pros	• Easy to learn/explore	• **Scriptable** (faster to repeat) • Quick to launch (near-native)
Cons	• Slower to repeat (need to click a lot; difficult to marco) • Slow to launch a virtual environment	• Difficult to learn/explore

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - Bash shell commands
  - Emacs
  - Vi, Vim, Neovim
  - nano & others

A shell?🐚 and kernel?🌽

Popular shells

- Shell🐚: An interpreter for humans🧑‍💻 to communicate with the core of an OS🌽.
- In fact, shells are **Scripting Languages** like R or Python. You can define variables, call functions, use conditionals, enumerate iterations, ...
- Most widely installed shells:
    - "`sh`" (Bourne SHell), 1979 [default in Version 7 Unix]
    - "`bash`" (Bourne-Again SHell), 1989 [default in many Linux]
    - "`zsh`" (Z shell), 1990 [default in macOS]
    - And "`pwsh`" (PowerShell), 2006, [default in Windows] but open-sourced and available across OSs

Bash

Bourne-Again SHell

- First release: Brian Fox (1989-06-08, Free Software Foundation)
  - As a free software alternative (GNU project w/ Richard Stallman) for the Bourne SHell (SH)
- Ported to Linux by Linus Torvalds, and widely used as a default shell in various Linux distributions

Brian Jhan Fox (b. 1959)

Bash demo

REDIRECTIONS, ECHO, CAT, SED

Bash demo

conCATenate and print files, Stream EDitor

```
$ echo "X" > file.txt
$ echo "Y" >> file.txt
$ cat << EOF > file2.txt
$ sed 's/OldString/NewString/g' test.txt
$ sed 's/t[^ ]*xt/TEXT/g' test.txt
```

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - bash
  - Emacs
  - Vi, Vim, Neovim
  - nano & others

What is Emacs?

"Emacs is more powerful than any OSs"

GNU Emacs

Editor Macros

- Original EMACS: David A. Moon & Guy L Steele Jr. (1984) MIT AI Lab
- Gosling Emacs: James Gosling (1981) UniPress (sold at 395 $/copy in 1983)
- GNU Emacs: Richard Stallman (1984) Free Software Foundation

Richard Stallman (b. 1953)
a.k.a. St. IGNUcius, the Church of Emacs

Emacs demo

Open & close, copy & paste, find & replace

Emacs demo

Open & close, copy & paste, find & replace

- `C-x b bufname RET`
- `C-x C-c`
- `C-a, C-e`
- `C-@, M-w, C-y`
- `C-x u`
- `M-%`

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - bash
  - Emacs
  - Vi, Vim, Neovim
  - nano & others

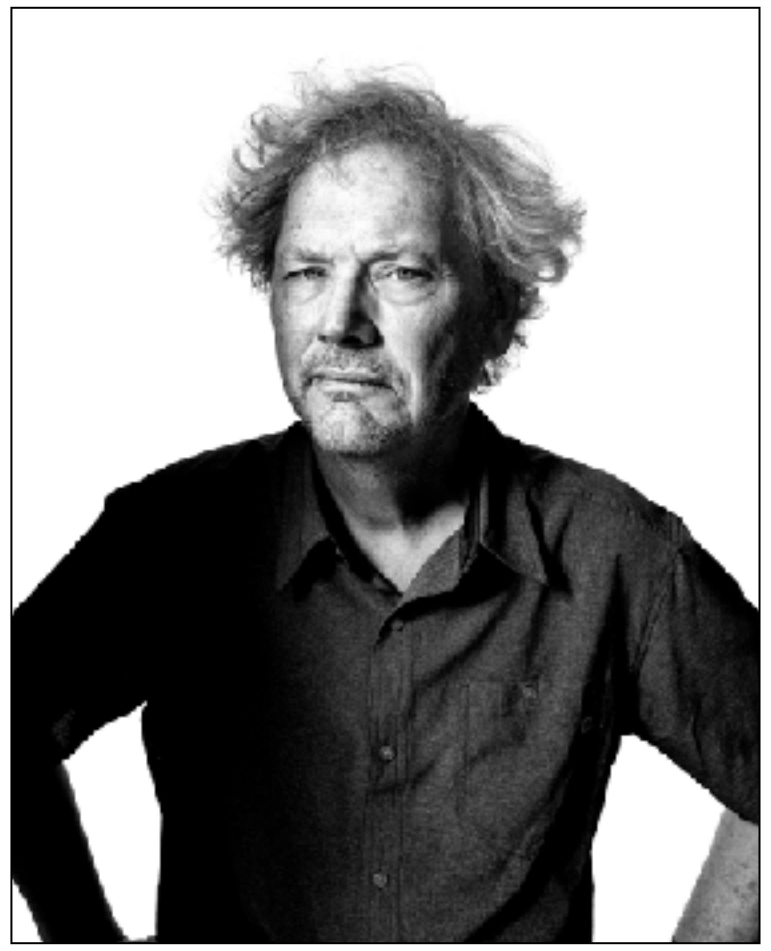
What is Vim?

"It will be painful at first and painful at last. Good."

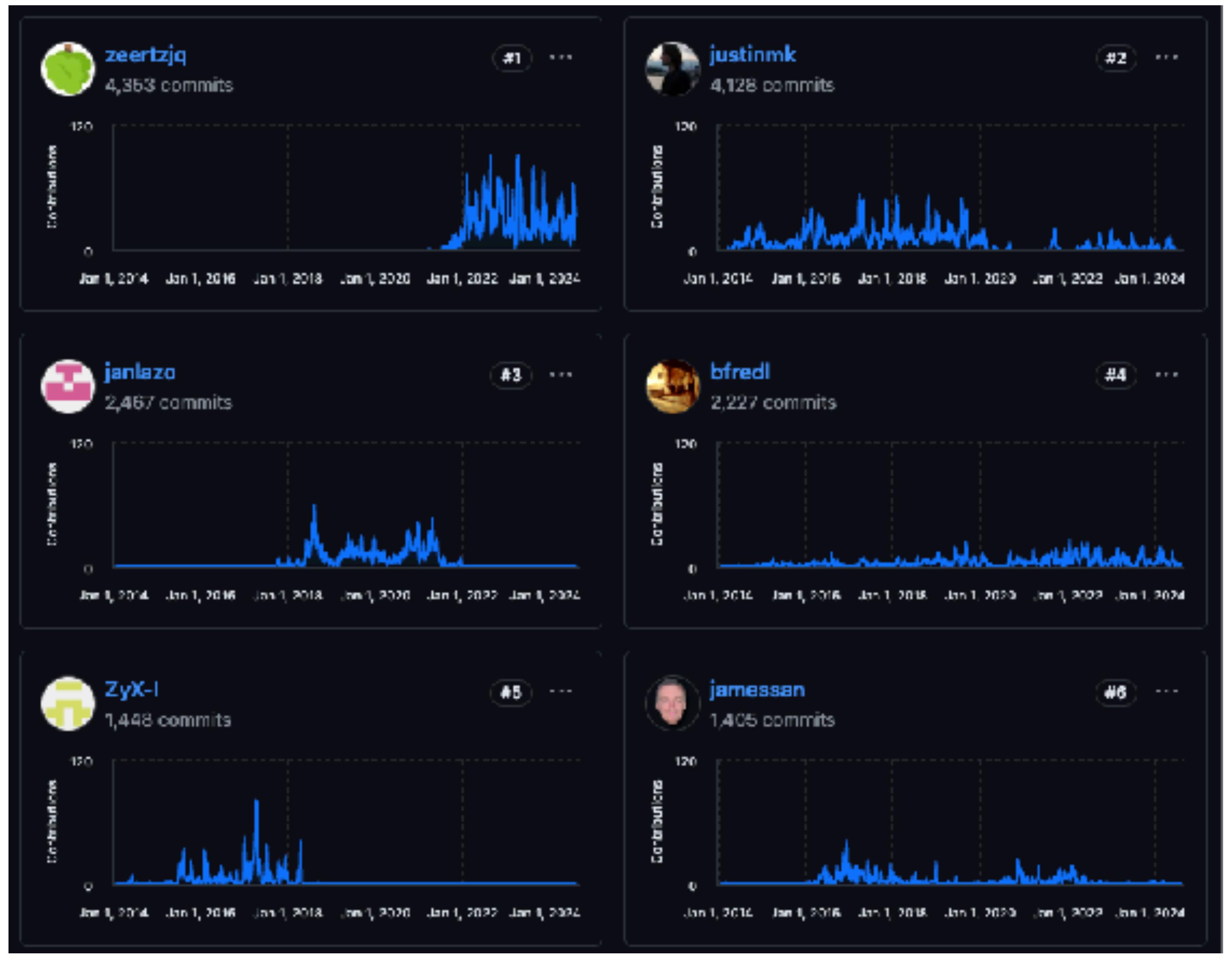
Vi/Vim/neovim

"vi" for visual mode of a line editor called "ex"

- **/ˌviːˈaɪ/**: William N. Joy (1976), BSD-licensed
- **/vɪm/** : Bram Moolenaar (1991)
- **/ˈnɛ.ovɪm/**: 30 core-devs & 1300+ contributors (2014)
- Most popular CLI-editor (Vim: 22.29%, Neovim: 11.88%, Nano: 8.98%, Emacs: 4.69%) in [Stack Overflow survey 2023](#) (for GUI-editor, VS Code: 73.71%, VS: 28.43%; multiple answers)


Bill Joy (b. 1954)

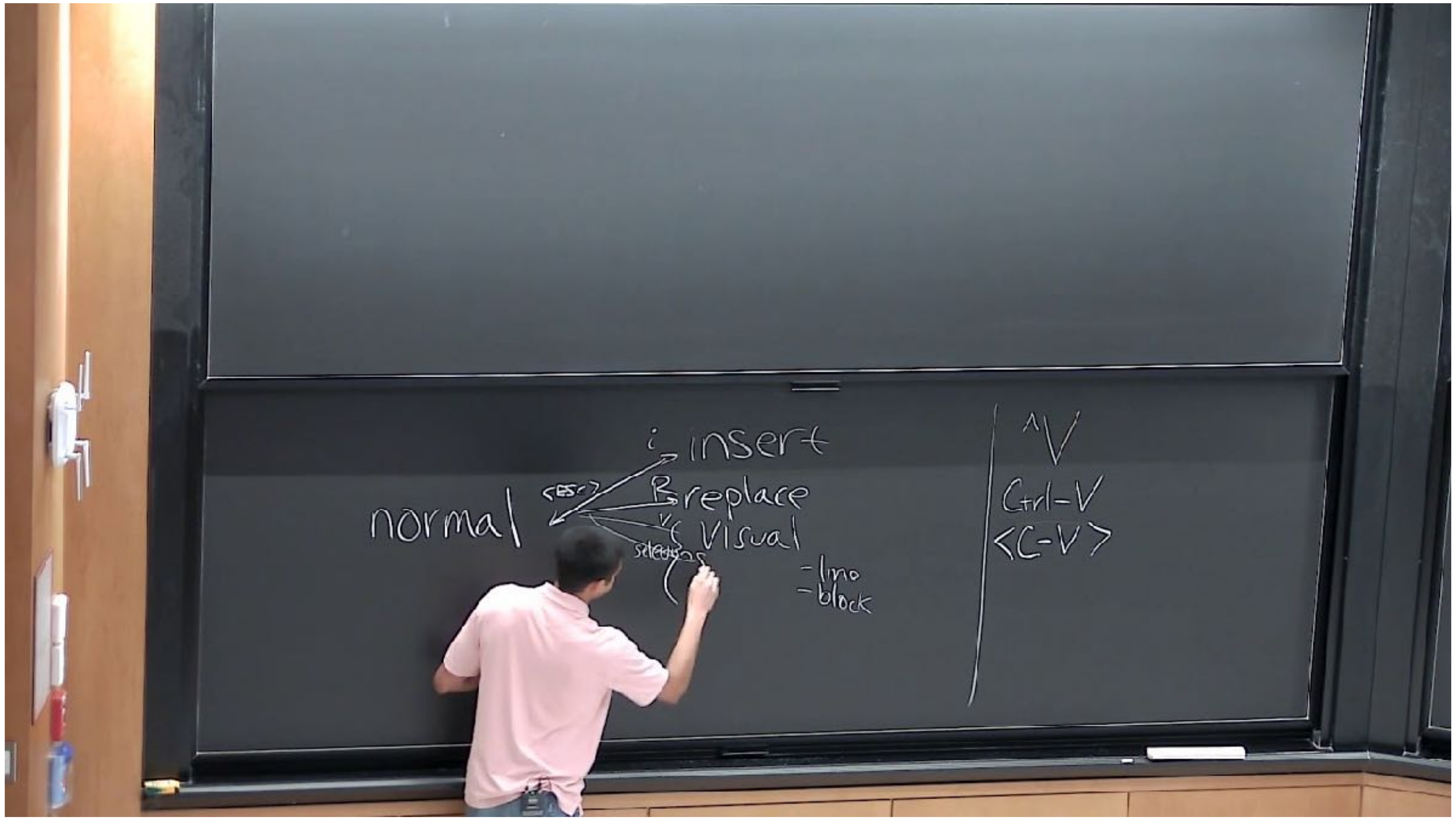
Bram Moolenaar (1961-2023)


zeertzjq 4,353 commits #1
justinmk 4,128 commits #2
janlazo 2,457 commits #3
bfredl 2,227 commits #4
ZyX-I 1,448 commits #5
jamessan 1,405 commits #6

https://github.com/neovim/neovim/graphs/contributors

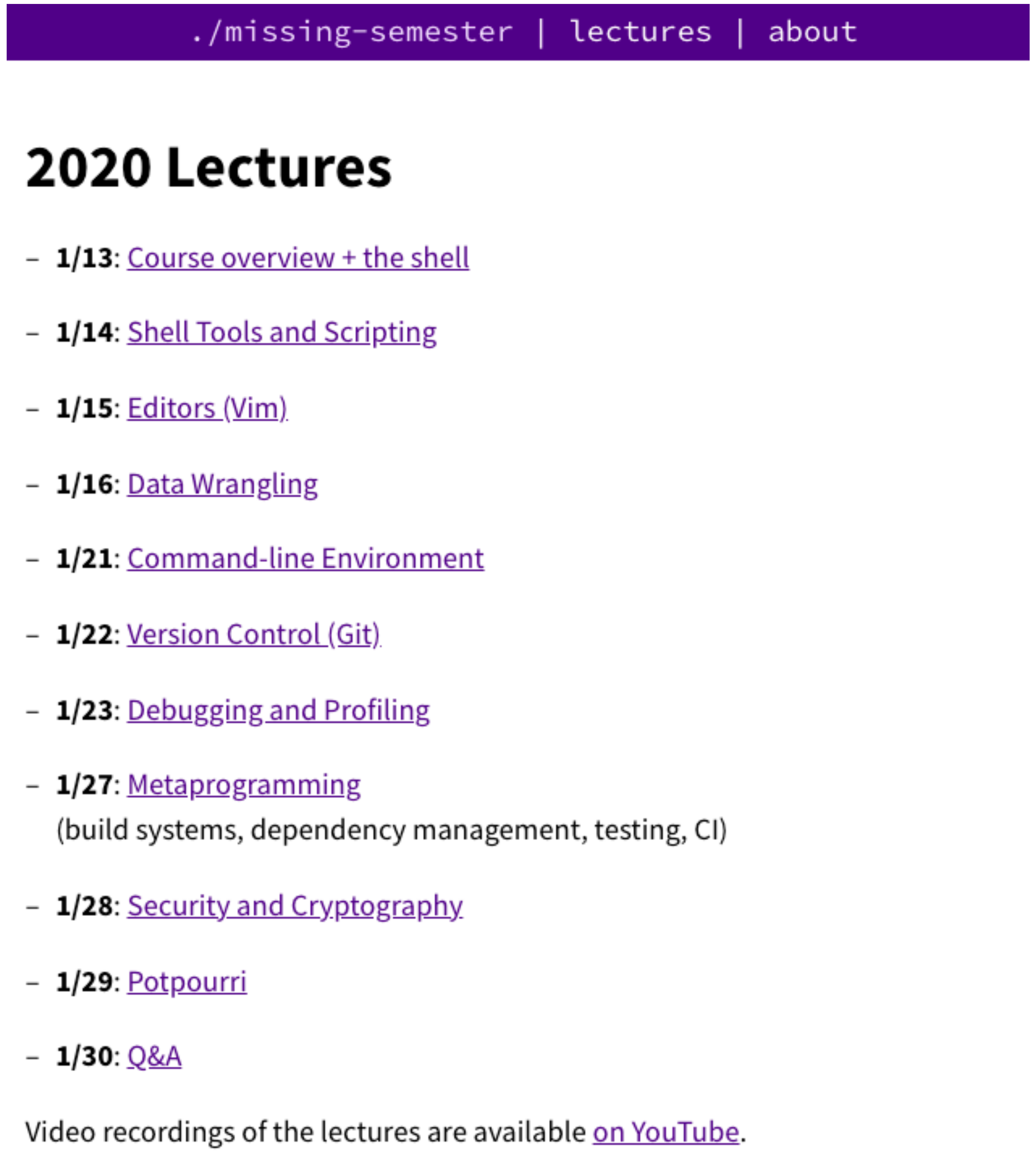
MIT lecture on how to use Vim! 🤯

"Vim's interface is a programming language!"



“The Missing Semester of Your CS Education”

https://missing.csail.mit.edu/

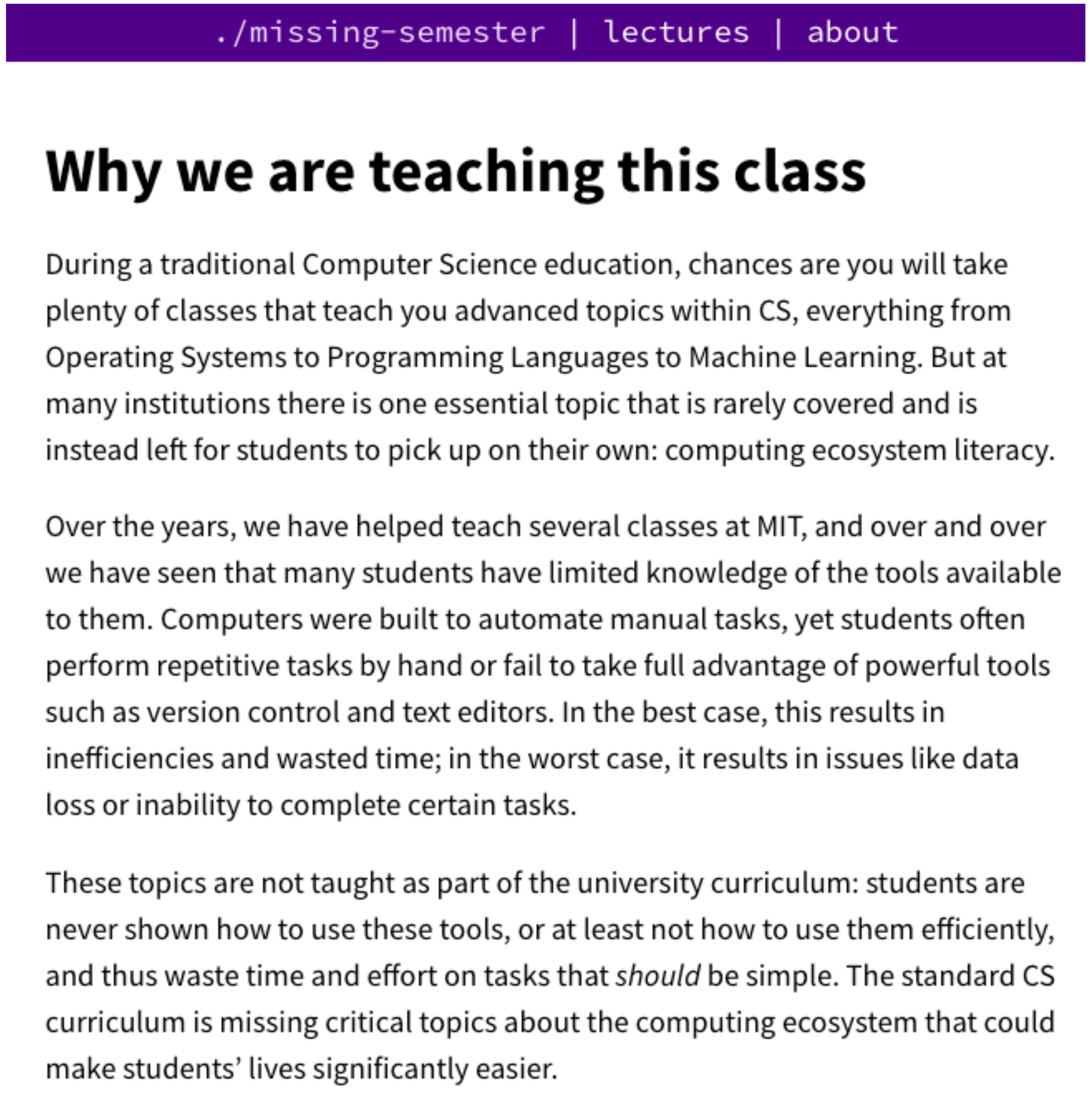


./missing-semester | lectures | about

2020 Lectures

- **1/13**: Course overview + the shell
- **1/14**: Shell Tools and Scripting
- **1/15**: Editors (Vim)
- **1/16**: Data Wrangling
- **1/21**: Command-line Environment
- **1/22**: Version Control (Git)
- **1/23**: Debugging and Profiling
- **1/27**: Metaprogramming
  (build systems, dependency management, testing, CI)
- **1/28**: Security and Cryptography
- **1/29**: Potpourri
- **1/30**: Q&A

Video recordings of the lectures are available on YouTube.



./missing-semester | lectures | about

Why we are teaching this class

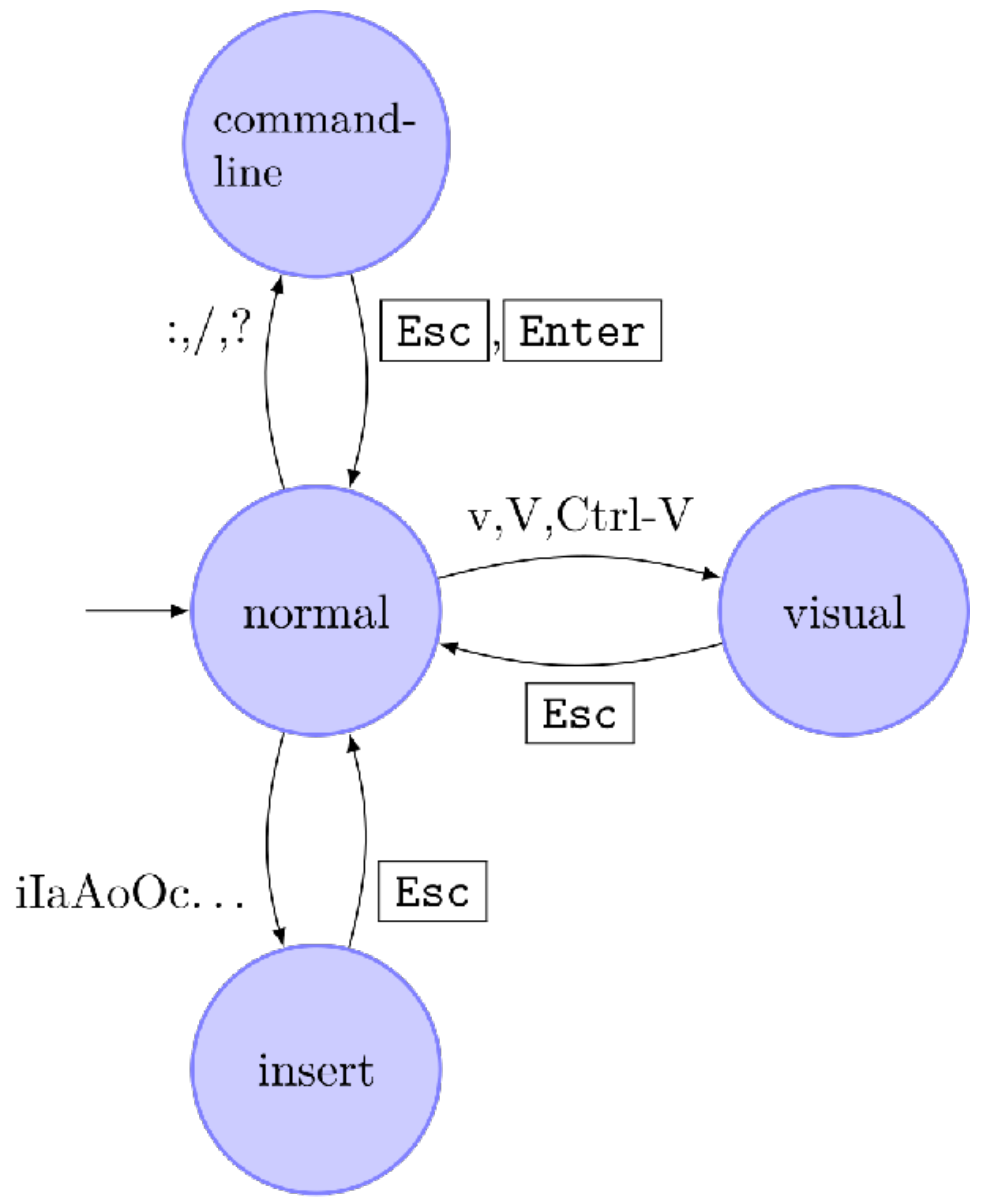
During a traditional Computer Science education, chances are you will take plenty of classes that teach you advanced topics within CS, everything from Operating Systems to Programming Languages to Machine Learning. But at many institutions there is one essential topic that is rarely covered and is instead left for students to pick up on their own: computing ecosystem literacy.

Over the years, we have helped teach several classes at MIT, and over and over we have seen that many students have limited knowledge of the tools available to them. Computers were built to automate manual tasks, yet students often perform repetitive tasks by hand or fail to take full advantage of powerful tools such as version control and text editors. In the best case, this results in inefficiencies and wasted time; in the worst case, it results in issues like data loss or inability to complete certain tasks.

These topics are not taught as part of the university curriculum: students are never shown how to use these tools, or at least not how to use them efficiently, and thus waste time and effort on tasks that *should* be simple. The standard CS curriculum is missing critical topics about the computing ecosystem that could make students’ lives significantly easier.

Vim demo

open & close, copy & paste, find & replace



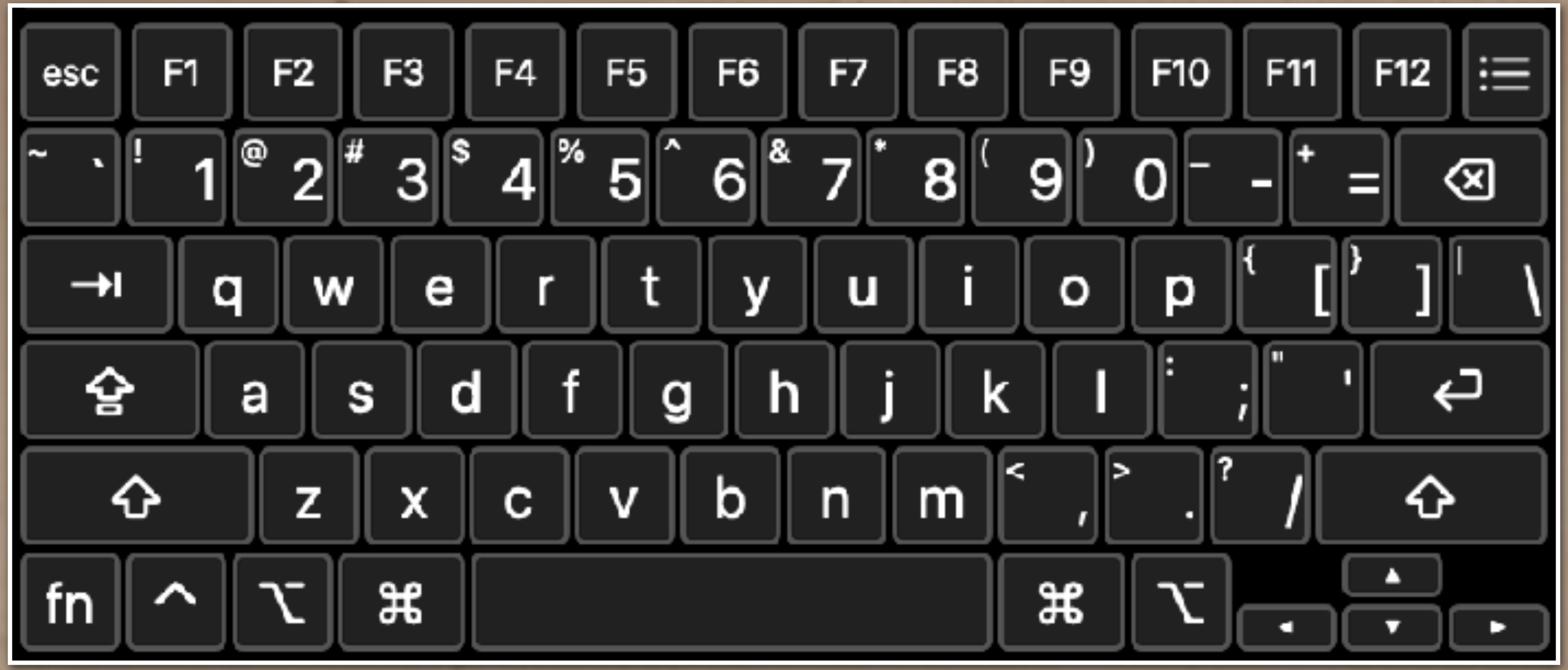
Vim demo

Normal mode commands

General: `u, C-r, .`

Motions: `hjkl, wbe, }{, 0^$, HML, C-u C-d, C-f C-b, G gg, [number]G, f[char], F[char], [count]l, [count]b, %, /[pattern], ?[pattern]`

Edits: `d[motion], c[motion], iI, oO, aA, yy, y[count]y, y[motion], p, [count]p`



https://terminals-wiki.org/wiki/index.php/File:Lear_Siegler_ADM-3A-2.jpg

Vim demo

How you talk to Vim: Composability of Vim's syntax

```
[edit]..[motion]: dw

[edit]..[[count]×[motion]]: d3w

[edit]..[object]: das

[count]×[motion|edit|general]: 3w, 2p, 100.

[count]×[[edit]..[motion]]: 3dw
```

Vim demo

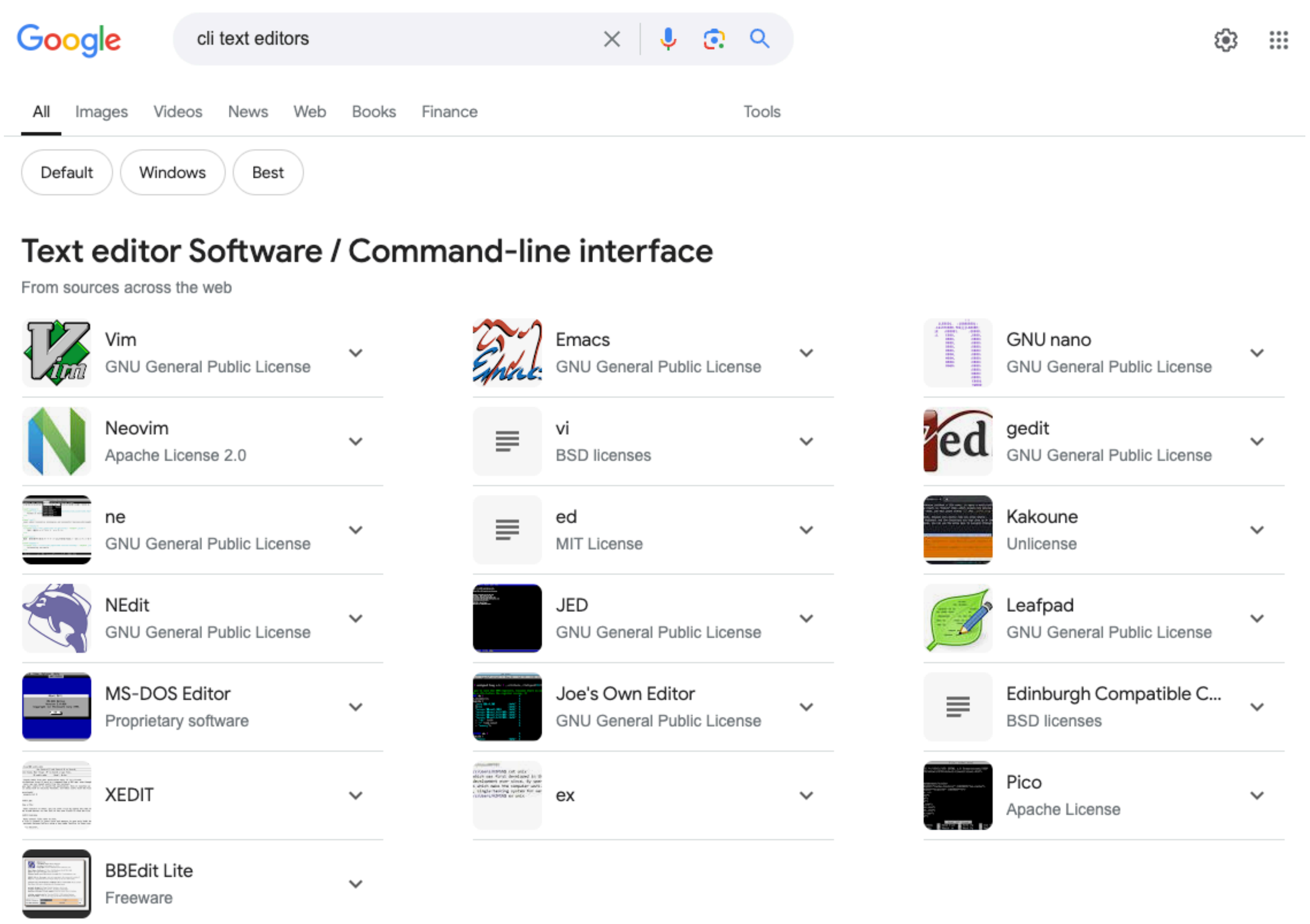
Command mode

```
/[pattern] AND n OR N

:noh

:1,10s/this/That/gc

:%s/this/That/g

:sp OR :vsp AND C+w w

:tabedit OR tabnew
AND gt OR gT
```

```
:edit [filename]

:ls

:[number]RET

:qa!

:wq
```

Agenda

- **DEFINITION**: What is a "Terminal" (Command Line Interface; CLI)?
- **MOTIVATION**: Why should we use a Terminal and edit a file there?
- **METHODS**:
  - bash
  - Emacs
  - Vi, Vim, Neovim
  - nano & others



Text editor Software / Command-line interface

From sources across the web

Vim GNU General Public License	Emacs GNU General Public License	GNU nano GNU General Public License
Neovim Apache License 2.0	vi BSD licenses	gedit GNU General Public License
ne GNU General Public License	ed MIT License	Kakoune Unlicense
NEdit GNU General Public License	JED GNU General Public License	Leafpad GNU General Public License
MS-DOS Editor Proprietary software	Joe's Own Editor GNU General Public License	Edinburgh Compatible C... BSD licenses
XEDIT	ex	Pico Apache License
BBEdit Lite Freeware		

Nano demo

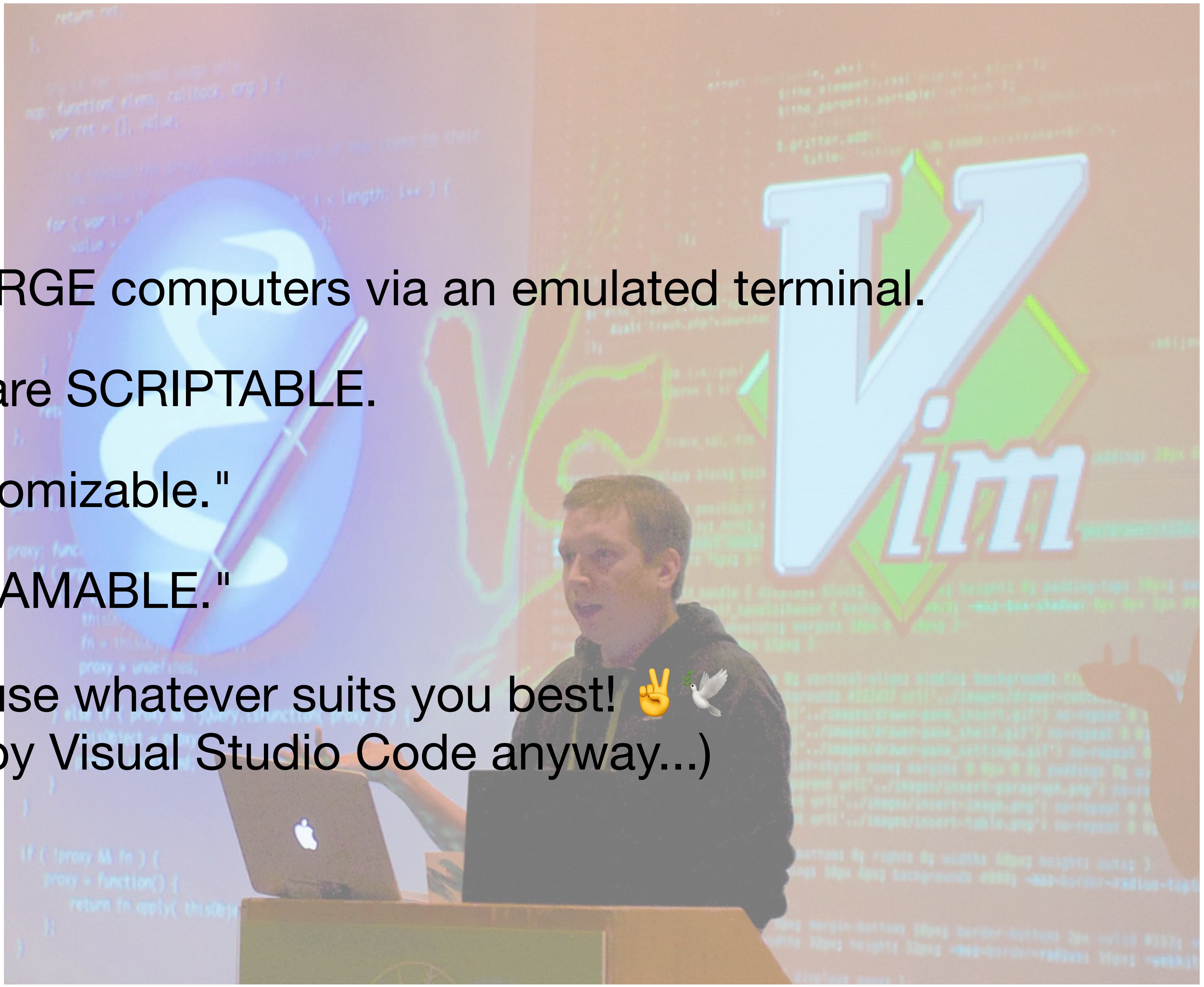
Cursor movement, copy & paste, find & replace

How to set up nano as a default editor for git?

If you don't like Vim... 😢

```
$ git config --global core.editor nano
```

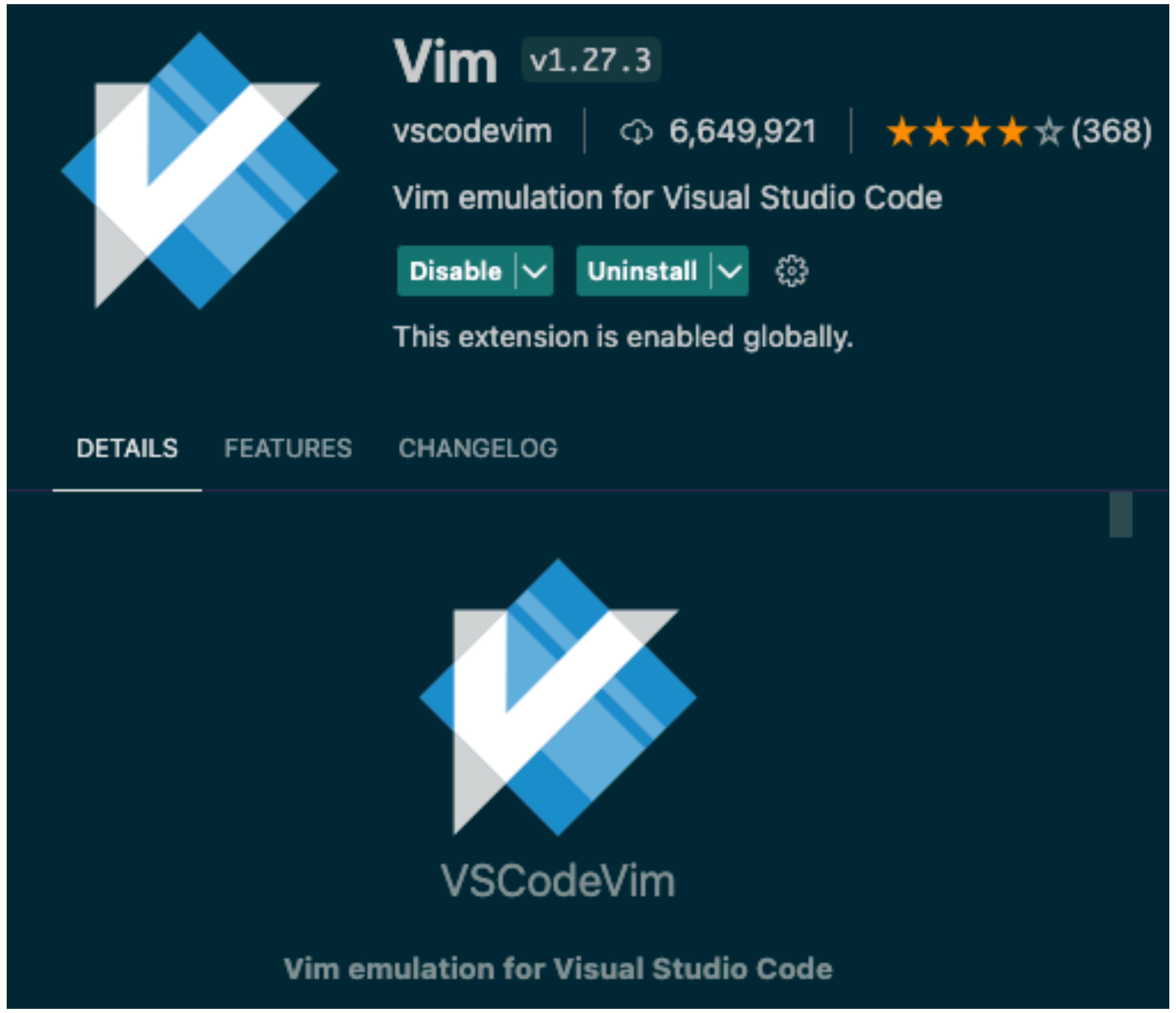
Summary



- We are still using those LARGE computers via an emulated terminal.
- Command-line interfaces are SCRIPTABLE.
- "Emacs is completely customizable."
- "Vim's interface is PROGRAMABLE."
- But of course you should use whatever suits you best! ✌️🕊️
  (The Holy War has ended by Visual Studio Code anyway...)

Summary

- (Or is the War back?)



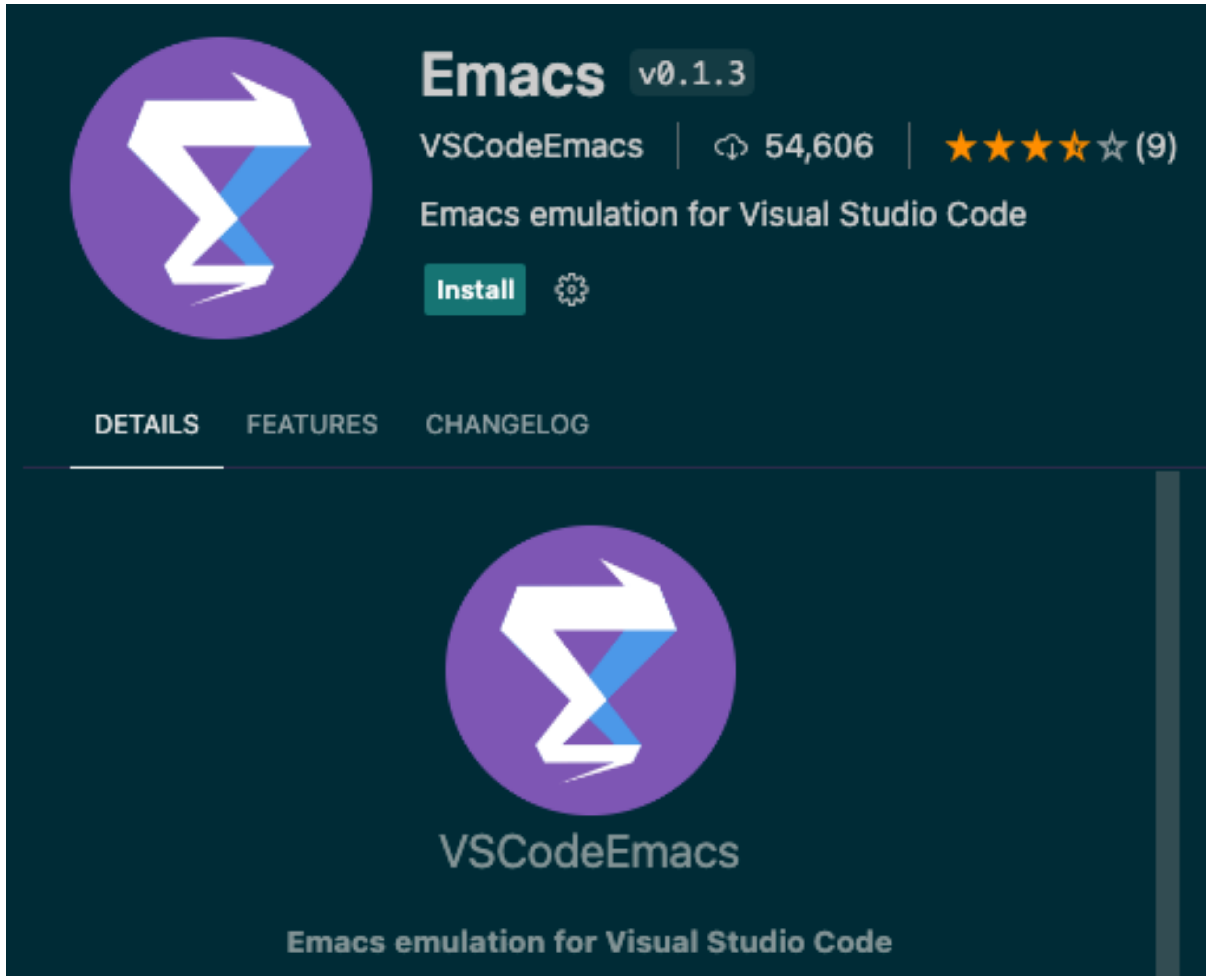
Vim v1.27.3

vscodevim | 6,649,921 | ★★★★☆ (368)

Vim emulation for Visual Studio Code

This extension is enabled globally.

Vim emulation for Visual Studio Code



Emacs v0.1.3

VSCodeEmacs | 54,606 | ★★★★☆ (9)

Emacs emulation for Visual Studio Code

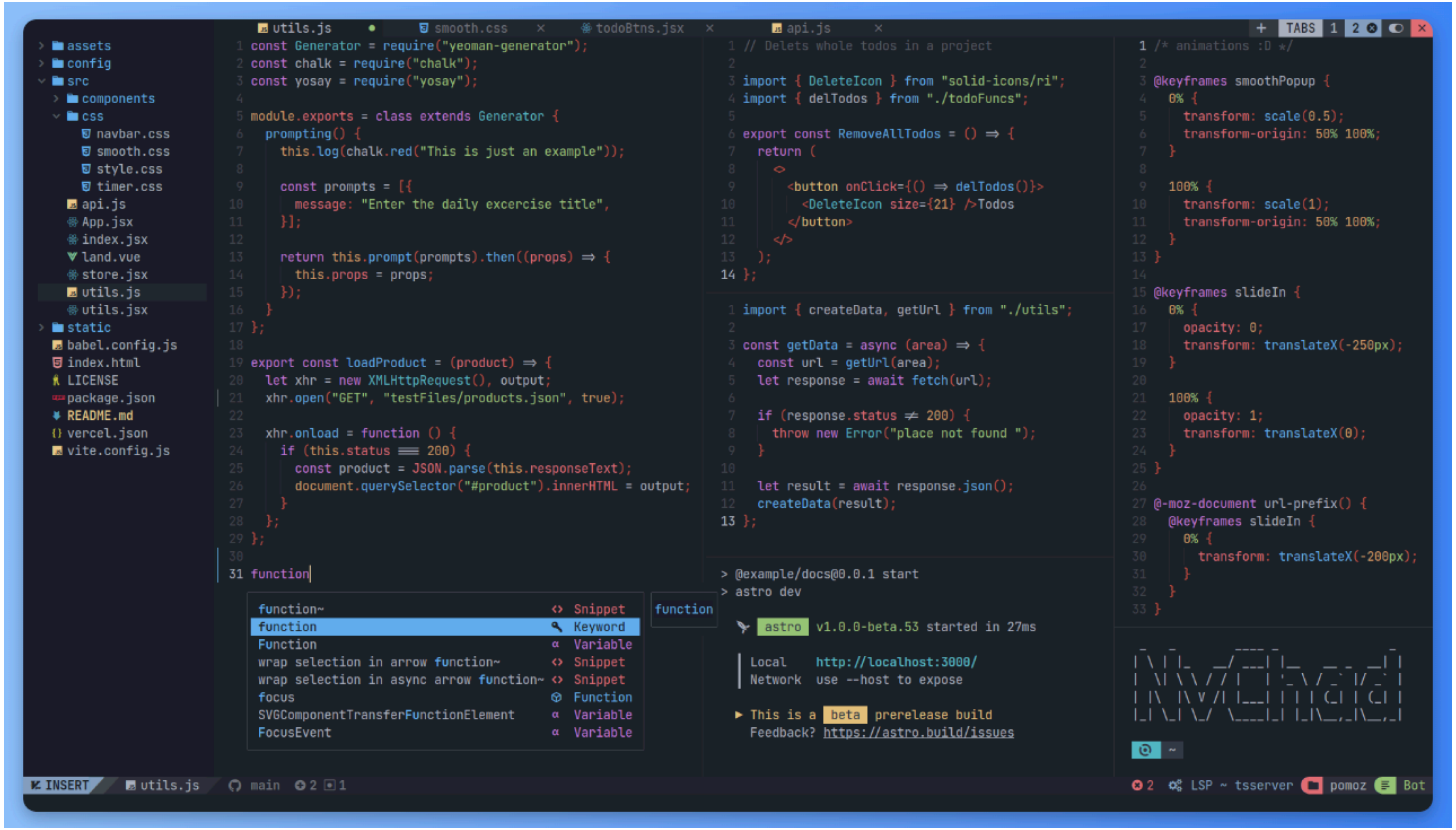
Emacs emulation for Visual Studio Code

Discussion

Personal thoughts 🤔

- Do CLI-text editors replace IDE?
- Do CLI-text editors really enhance productivity?

(n)vi(m) and/or Emacs as an IDE?



```
> assets
> config
v src
  > components
  v css
      navbar.css
      smooth.css
      style.css
      timer.css
    api.js
    App.jsx
    index.jsx
    land.vue
    store.jsx
    utils.js
    utils.jsx
> static
  babel.config.js
  index.html
  LICENSE
  package.json
  README.md
  vercel.json
  vite.config.js
```

```
utils.js ●    smooth.css ×    todoBtns.jsx ×    api.js ×    + TABS 1 2
```

```
const Generator = require("yeoman-generator");
const chalk = require("chalk");
const yosay = require("yosay");

module.exports = class extends Generator {
  prompting() {
    this.log(chalk.red("This is just an example"));

    const prompts = [{
      message: "Enter the daily excercise title",
    }];

    return this.prompt(prompts).then((props) => {
      this.props = props;
    });
  }
};

export const loadProduct = (product) => {
  let xhr = new XMLHttpRequest(), output;
  xhr.open("GET", "testFiles/products.json", true);

  xhr.onload = function () {
    if (this.status === 200) {
      const product = JSON.parse(this.responseText);
      document.querySelector("#product").innerHTML = output;
    }
  };
};

function
```

```
function~                                  Snippet
function                                   Keyword
Function                                   Variable
wrap selection in arrow function~          Snippet
wrap selection in async arrow function~    Snippet
focus                                      Function
SVGComponentTransferFunctionElement        Variable
FocusEvent                                 Variable
```

```
function
```

```
// Delets whole todos in a project

import { DeleteIcon } from "solid-icons/ri";
import { delTodos } from "./todoFuncs";

export const RemoveAllTodos = () => {
  return (
    <>
      <button onClick={() => delTodos()}>
        <DeleteIcon size={21} />Todos
      </button>
    </>
  );
};
```

```
import { createData, getUrl } from "./utils";

const getData = async (area) => {
  const url = getUrl(area);
  let response = await fetch(url);

  if (response.status != 200) {
    throw new Error("place not found ");
  }

  let result = await response.json();
  createData(result);
};
```

```
> @example/docs@0.0.1 start
> astro dev

  astro  v1.0.0-beta.53 started in 27ms

  Local    http://localhost:3000/
  Network  use --host to expose

▶ This is a  beta  prerelease build
  Feedback? https://astro.build/issues
```

```
/* animations :D */

@keyframes smoothPopup {
  0% {
    transform: scale(0.5);
    transform-origin: 50% 100%;
  }

  100% {
    transform: scale(1);
    transform-origin: 50% 100%;
  }
}

@keyframes slideIn {
  0% {
    opacity: 0;
    transform: translateX(-250px);
  }

  100% {
    opacity: 1;
    transform: translateX(0);
  }
}

@-moz-document url-prefix() {
  @keyframes slideIn {
    0% {
      transform: translateX(-200px);
    }
  }
}
```

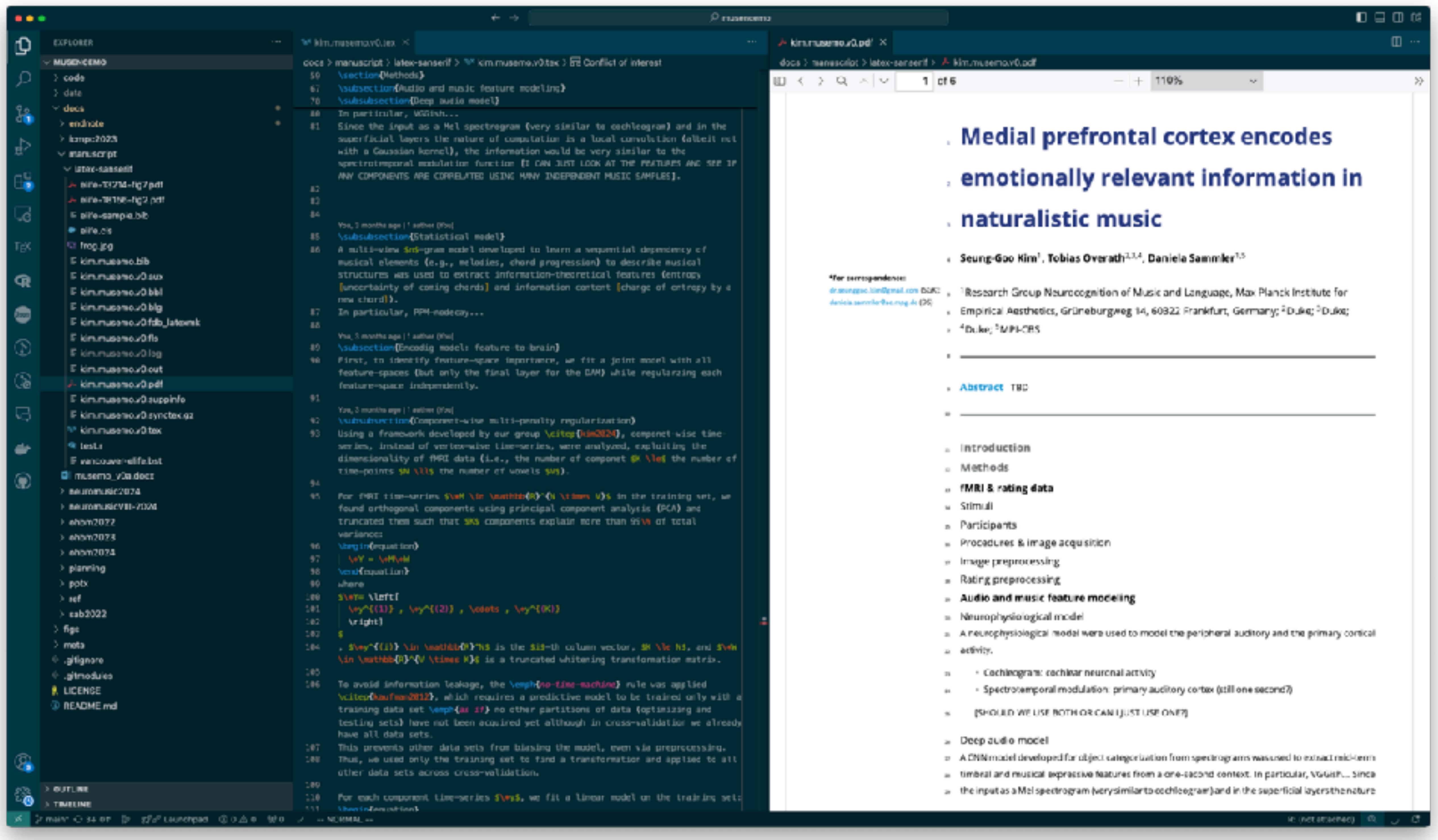
```
 _   _       ____ _               _
| \ | |_   _/ ___| |__   __ _  __| |
|  \| \ \ / / |   | '_ \ / _` |/ _` |
| |\  |\ V /| |___| | | | (_| | (_| |
|_| \_| \_/  \____|_| |_|\__,_|\__,_|

 ~
```

```
INSERT  utils.js   main  2  1                  2  LSP ~ tsserver  pomoz  Bot
```

IDE (Integrated development environment)?

Interpreter/compiler + text editor (+ shell + file browser + ... + AI)

- Visual Studio for C++ and .NET
- Visual Studio Code for JavaScript/HTML, Python, ..., and LaTex!
- IntelliJ IDEA for Java
- PyCharm, Spyder, IDLE for Python
- MATLAB for MATLAB
- RStudio for R



```
\section{Methods}
\subsection{Audio and music feature modeling}
\subsubsection{Deep audio model}
In particular, VGGish...
Since the input as a Mel spectrogram (very similar to cochleogram) and in the
superficial layers the nature of computation is a local convolution (albeit not
with a Gaussian kernel), the information would be very similar to the
spectrotemporal modulation function [I CAN JUST LOOK AT THE FEATURES AND SEE IF
ANY COMPONENTS ARE CORRELATED USING MANY INDEPENDENT MUSIC SAMPLES].



\subsubsection{Statistical model}
A multi-view $n$-gram model developed to learn a sequential dependency of
musical elements (e.g., melodies, chord progression) to describe musical
structures was used to extract information-theoretical features (entropy
[uncertainty of coming chords] and information content [change of entropy by a
new chord]).
In particular, PPM-nodecay...

\subsection{Encodig model: feature to brain}
First, to identify feature-space importance, we fit a joint model with all
feature-spaces (but only the final layer for the DAM) while regularizing each
feature-space independently.

\subsubsection{Component-wise multi-penalty regularization}
Using a framework developed by our group \citep{kim2024}, componet-wise time-
series, instead of vertex-wise time-series, were analyzed, exploiting the
dimensionality of fMRI data (i.e., the number of componet $K \leq the number of
time-points $N \ll$ the number of voxels $V$).

For fMRI time-series $\vM \in \mathbb{R}^{N \times V}$ in the training set, we
found orthogonal components using principal component analysis (PCA) and
truncated them such that $K$ components explain more than 95\% of total
variance:
\begin{equation}
  \vY = \vM\vW
\end{equation}
where
$\vY= \left[
  \vy^{(1)} , \vy^{(2)} , \vdots , \vy^{(K)}
  \right]
$
, $\vy^{(i)} \in \mathbb{R}^N$ is the $i$-th column vector, $K \le N$, and $\vW
\in \mathbb{R}^{V \times K}$ is a truncated whitening transformation matrix.

To avoid information leakage, the \emph{no-time-machine} rule was applied
\citep{kaufman2012}, which requires a predictive model to be trained only with a
training data set \emph{as if} no other partitions of data (optimizing and
testing sets) have not been acquired yet although in cross-validation we already
have all data sets.
This prevents other data sets from biasing the model, even via preprocessing.
Thus, we used only the training set to find a transformation and applied to all
other data sets across cross-validation.

For each component time-series $\vy$, we fit a linear model on the training set:
\begin{equation}
```

Medial prefrontal cortex encodes emotionally relevant information in naturalistic music

Seung-Goo Kim[1], Tobias Overath[2,3,4], Daniela Sammler[1,5]

*For correspondence:

[1]Research Group Neurocognition of Music and Language, Max Planck Institute for Empirical Aesthetics, Grüneburgweg 14, 60322 Frankfurt, Germany; [2]Duke; [3]Duke; [4]Duke; [5]MPI-CBS

Abstract TBD

Introduction
Methods
fMRI & rating data
Stimuli
Participants
Procedures & image acquisition
Image preprocessing
Rating preprocessing
Audio and music feature modeling
Neurophysiological model
A neurophysiological model were used to model the peripheral auditory and the primary cortical activity.
- Cochleogram: cochlear neuronal activity
- Spectrotemporal modulation: primary auditory cortex (still one second?)

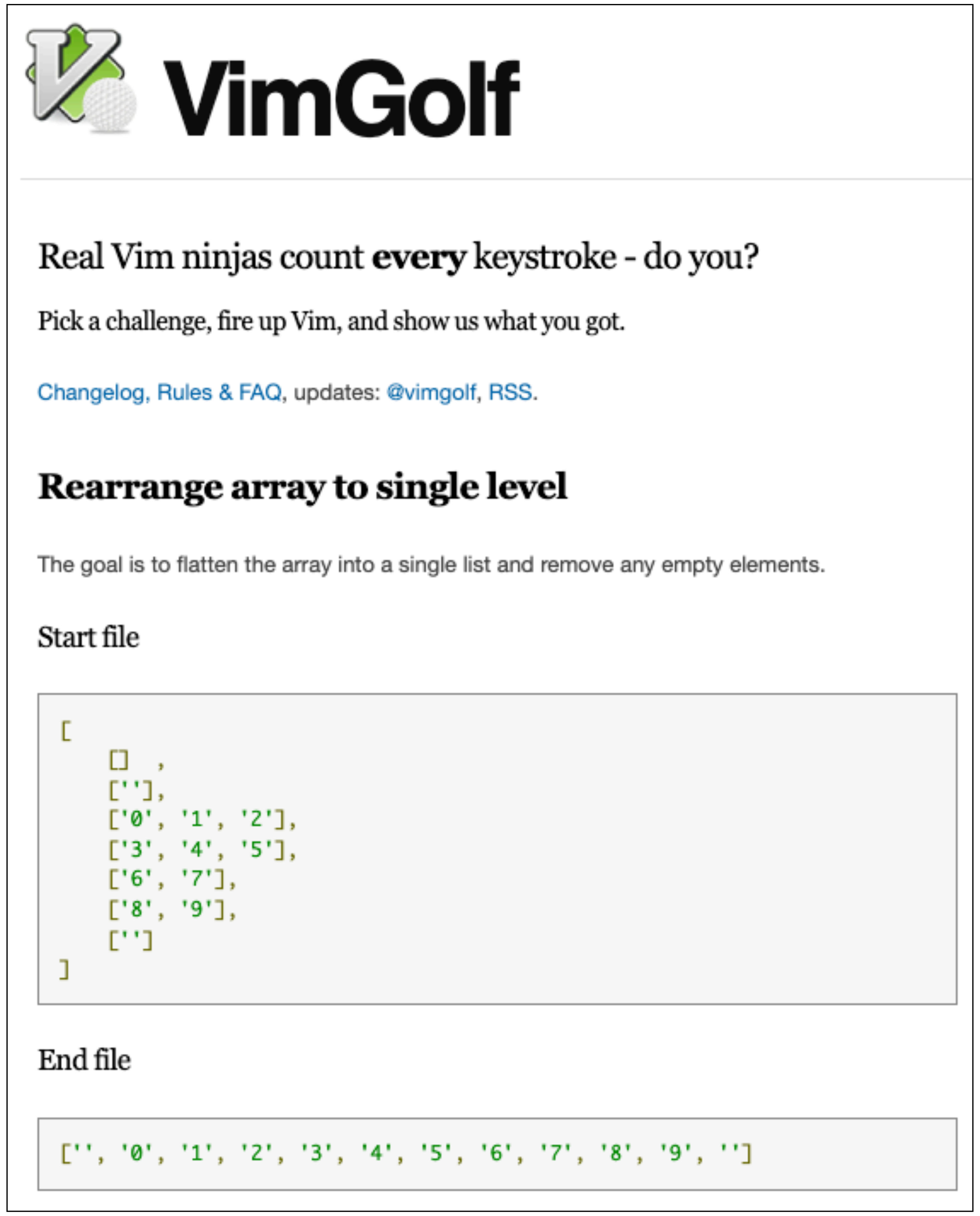
[SHOULD WE USE BOTH OR CAN I JUST USE ONE?]

Deep audio model
A CNN model developed for object categorization from spectrograms was used to extract mid-term timbral and musical expressive features from a one-second context. In particular, VGGish... Since the input as a Mel spectrogram (very similar to cochleogram) and in the superficial layers the nature

But, is it just a nerdy joke or else?

e.g., "Chuck Norris used negative one keystroke to write the entire OS"

- Obviously it's very fun.
- Or does saving a few milliseconds make really a difference?
- So far, no controlled user-test data available…😢



VimGolf

Real Vim ninjas count **every** keystroke - do you?

Pick a challenge, fire up Vim, and show us what you got.

Changelog, Rules & FAQ, updates: @vimgolf, RSS.

Rearrange array to single level

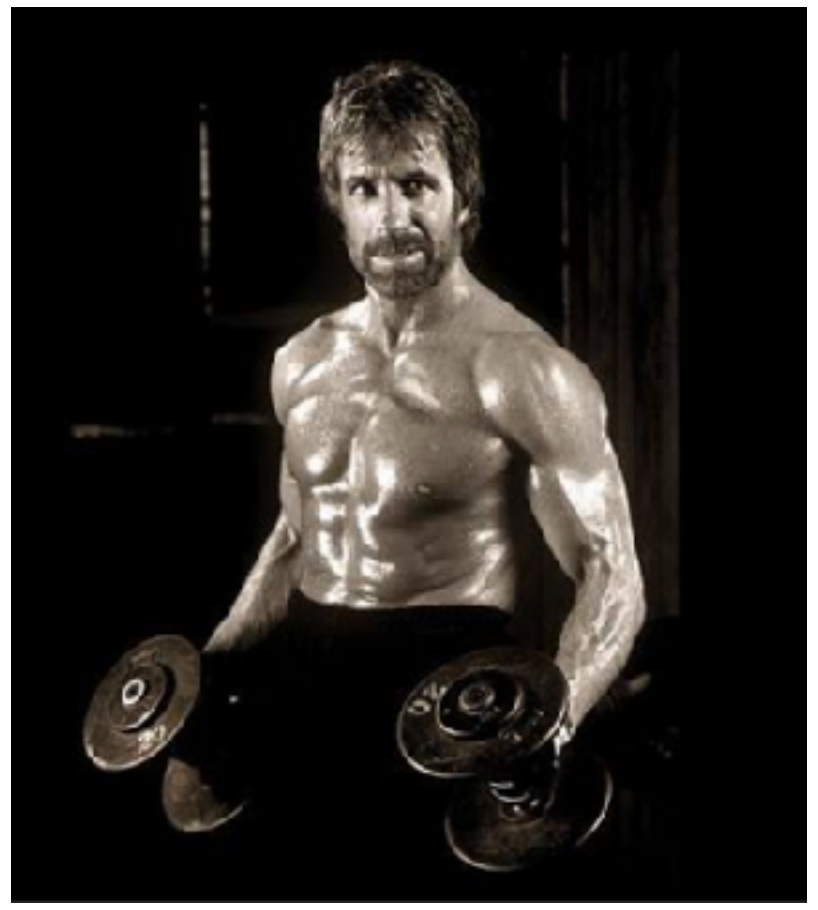
The goal is to flatten the array into a single list and remove any empty elements.

Start file

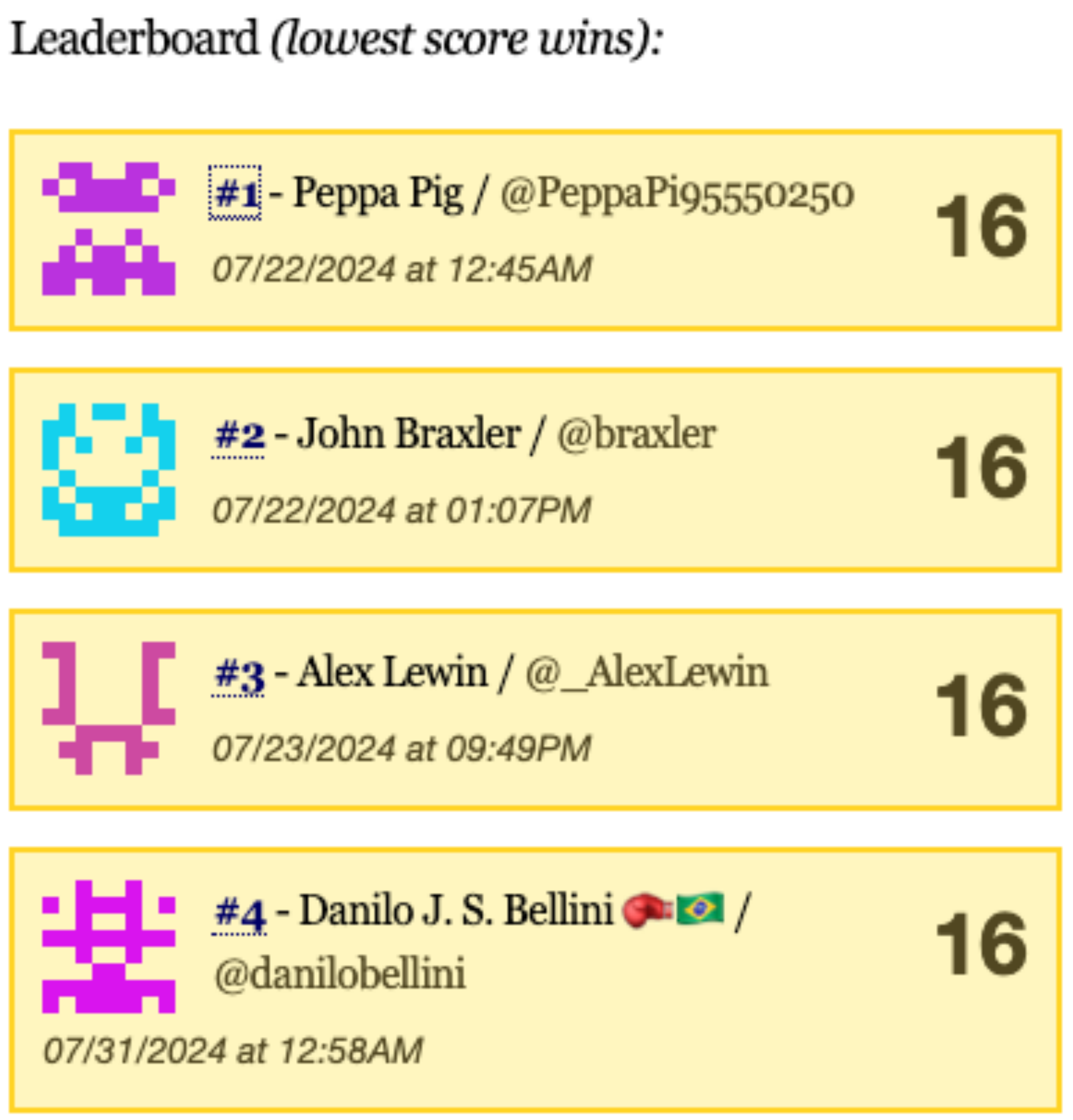
```
[
    []  ,
    [''],
    ['0', '1', '2'],
    ['3', '4', '5'],
    ['6', '7'],
    ['8', '9'],
    ['']
]
```

End file

```
['', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '']
```



Chuck Norris, from his Instagram (@chucknorris)



Leaderboard *(lowest score wins):*

#1 - Peppa Pig / @PeppaPi95550250 — 16
07/22/2024 at 12:45AM

#2 - John Braxler / @braxler — 16
07/22/2024 at 01:07PM

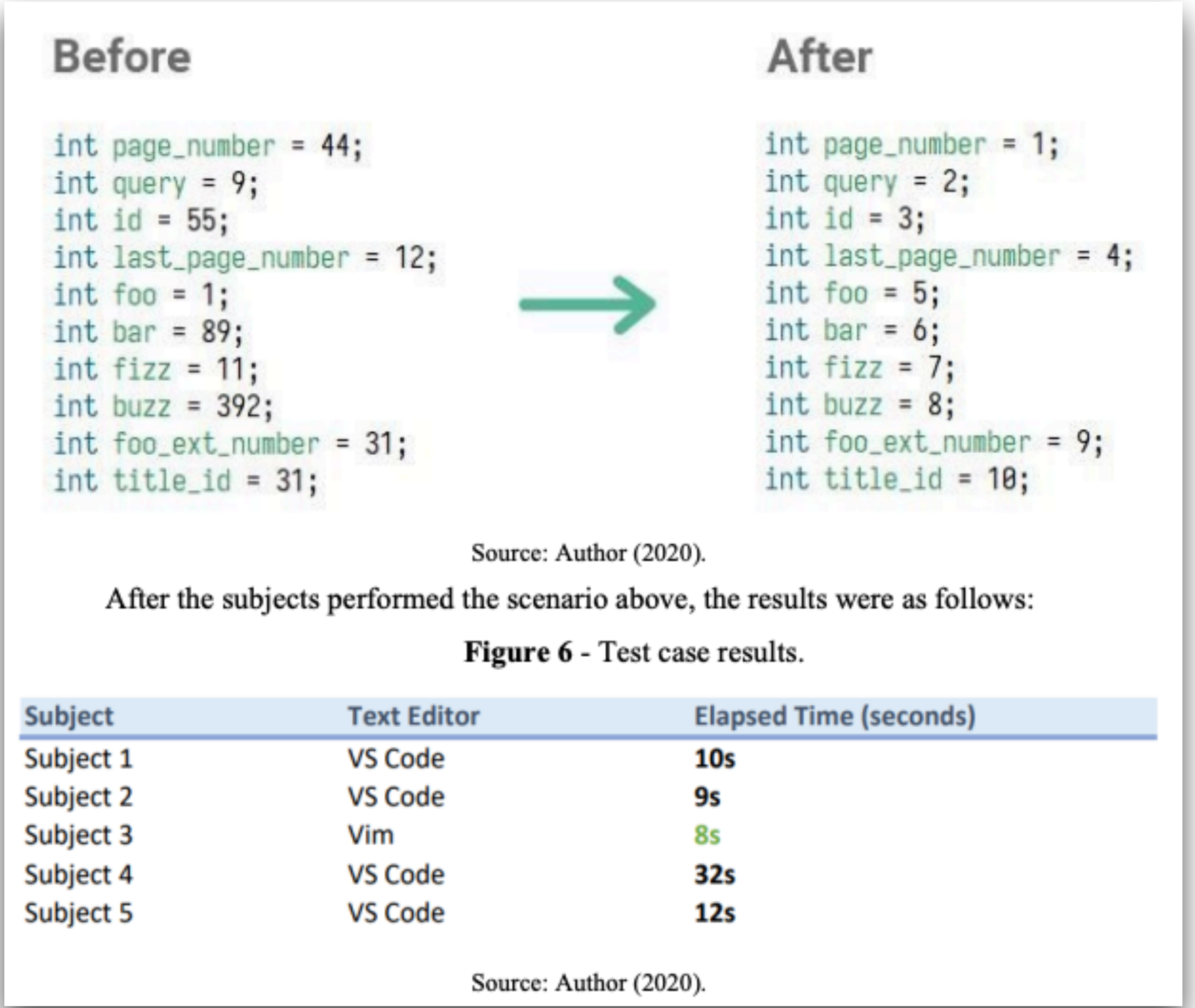
#3 - Alex Lewin / @_AlexLewin — 16
07/23/2024 at 09:49PM

#4 - Danilo J. S. Bellini 🥊🇧🇷 / @danilobellini — 16
07/31/2024 at 12:58AM

I found one from Google Scholar!

de Oliveira, B. C., & Zuchi, J. D. (2020). Efficiency in Writing Software With Vim. *Revista Interface Tecnológica*, 17(2), 386-397. https://doi.org/10.31510/infa.v17i2.1066

- Only one study from Brazil… violating all Fisher's principles — randomization, replication, orthogonality with only 5 subjects…😩🥺



Before

```
int page_number = 44;
int query = 9;
int id = 55;
int last_page_number = 12;
int foo = 1;
int bar = 89;
int fizz = 11;
int buzz = 392;
int foo_ext_number = 31;
int title_id = 31;
```

After

```
int page_number = 1;
int query = 2;
int id = 3;
int last_page_number = 4;
int foo = 5;
int bar = 6;
int fizz = 7;
int buzz = 8;
int foo_ext_number = 9;
int title_id = 10;
```

Source: Author (2020).

After the subjects performed the scenario above, the results were as follows:

Figure 6 - Test case results.

Subject	Text Editor	Elapsed Time (seconds)
Subject 1	VS Code	**10s**
Subject 2	VS Code	**9s**
Subject 3	Vim	8s
Subject 4	VS Code	**32s**
Subject 5	VS Code	**12s**

Source: Author (2020).

Perhaps?

LaTex vs. MS Word

Einleitung zum Themenheft „Ressourcenadaptive kognitive Prozesse“

Anthony Jameson, Kai Buchholz*

Sonderforschungsbereich 378, FB 14 Informatik und FR 5.1 Philosophie, Universität des Saarlandes, Postfach 151150, D-66041 Saarbrücken

In der Kognitionswissenschaft tritt der Begriff einer *beschränkten Ressource* in vielen Zusammenhängen auf. Insbesondere wird der Term *Ressource* auf sehr unterschiedliche Entitäten angewandt, unter anderem auf Zeit, (menschliches und maschinelles) Gedächtnis, Wissen und Information. Durch die breite Anwendung dieses Begriffs lassen sich einige weitere gemeinsame Begriffe – sowie damit zusammenhängende Fragestellungen – erkennen. Aus diesen Gemeinsamkeiten ergeben sich Möglichkeiten für eine fruchtbare interdisziplinäre Zusammenarbeit.

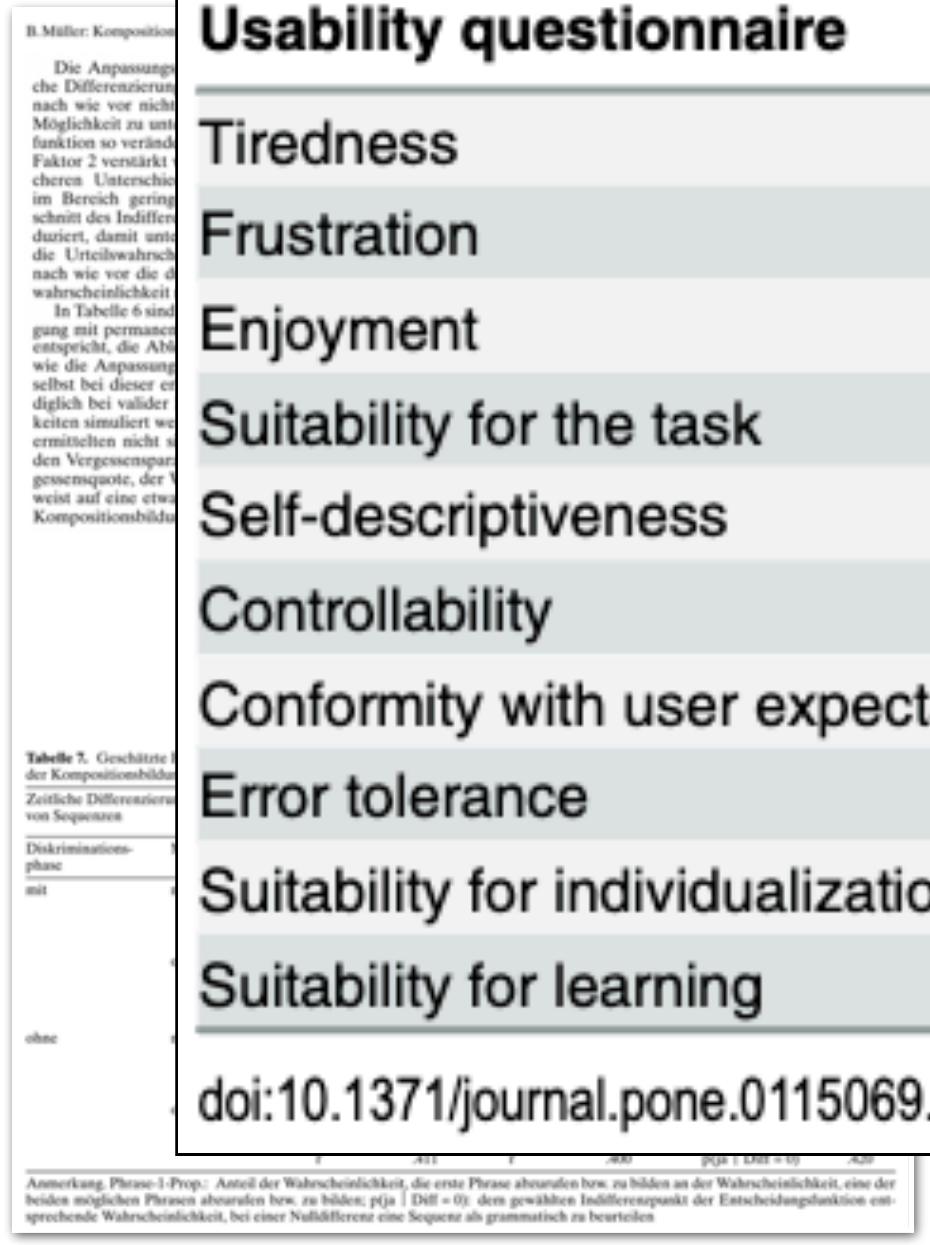
Anfang 1996 nahm in Saarbrücken der von der Deutschen Forschungsgemeinschaft unterstützte Sonderforschungsbereich „Ressourcenadaptive kognitive Prozesse“ seine Arbeit auf. Dieses Forschungsprogramm umfaßt 11 Projekte, deren Leiter in den Disziplinen Computerlinguistik, Informatik, Philosophie und Psychologie tätig sind; mehrere der Projekte werden von Vertretern verschiedener Disziplinen gemeinsam geleitet.

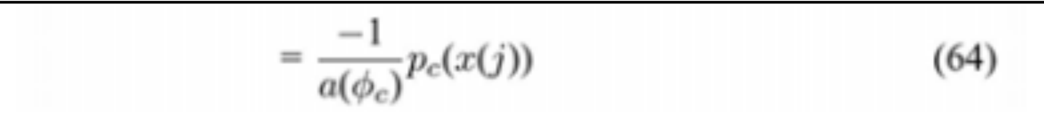
Das vorliegende Themenheft soll den Lesern eine repräsentative Stichprobe der Forschung in diesem SFB bieten. Während die Autoren und der Gastherausgeber aus dem SFB 378 stammen, wurden alle Gutachten (mindestens zwei für jede Einreichung) von Fachkollegen außerhalb Saarbrückens erstellt. Mit ihren fachkundigen und ausführlichen Kritiken haben die Gutachter das Themenheft entscheidend mitgestaltet. Aufgrund der Gutachten traf der Herausgeber der Zeitschrift *Kognitionswissenschaft*, Prof. Dr. Gerhard Weber, die Entscheidungen in bezug auf Annahme und Ablehnung von Manuskripten.

In dieser Einleitung wird zuerst der weitgefaßte Begriff einer *Ressource*, der im SFB 378 verwendet wird, kurz motiviert und skizziert. Dann werden die einzelnen Artikel im Themenheft in diesen Rahmen eingeordnet. Insbesondere wird am Schluß erörtert, welche Formen von *Ressourcenadaptivität* in den Projekten des SFB unterschieden werden.[1]

* Der erste Autor ist Gastherausgeber dieses Themenhefts. Der Beitrag des zweiten Autors zu dieser Einleitung wurde von der Deutschen Forschungsgemeinschaft im Rahmen des SFB 378 gefördert.

[1] Eine ausführliche Besprechung und Einordnung der Artikel dieses Themenhefts ist über die WWW-Seite des SFB 378 verfügbar: http://www.coli.uni-sb.de/sfb378/.





$$= \frac{-1}{a(\phi_c)} p_c(x(j)) \qquad (64)$$

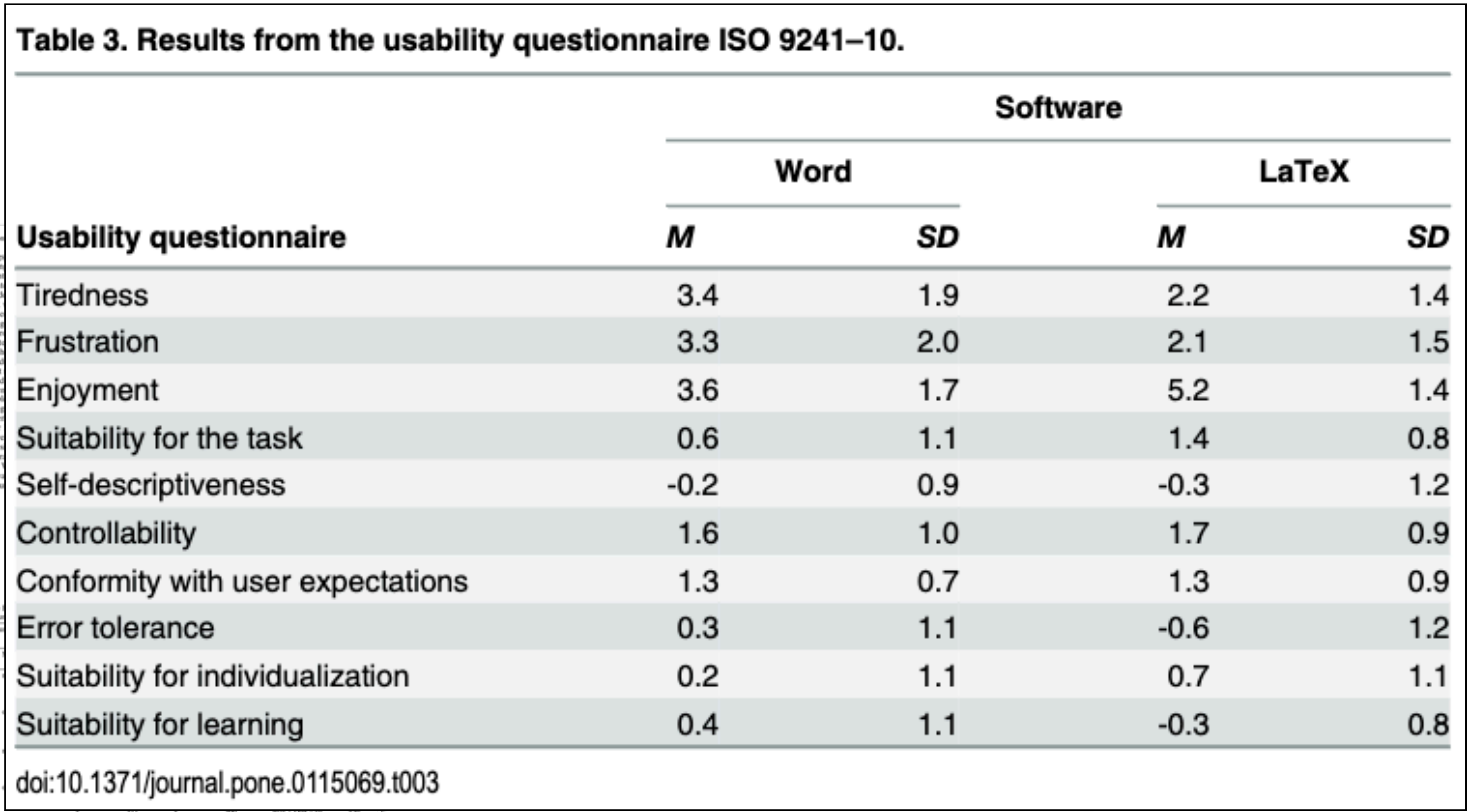


Table 3. Results from the usability questionnaire ISO 9241–10.

	Software				
	Word		LaTeX		
Usability questionnaire	*M*	*SD*	*M*	*SD*	
Tiredness	3.4	1.9	2.2	1.4	p<.05
Frustration	3.3	2.0	2.1	1.5	p<.05
Enjoyment	3.6	1.7	5.2	1.4	**p<.01**
Suitability for the task	0.6	1.1	1.4	0.8	p<.05
Self-descriptiveness	-0.2	0.9	-0.3	1.2	
Controllability	1.6	1.0	1.7	0.9	
Conformity with user expectations	1.3	0.7	1.3	0.9	
Error tolerance	0.3	1.1	-0.6	1.2	
Suitability for individualization	0.2	1.1	0.7	1.1	
Suitability for learning	0.4	1.1	-0.3	0.8	p<.05

doi:10.1371/journal.pone.0115069.t003

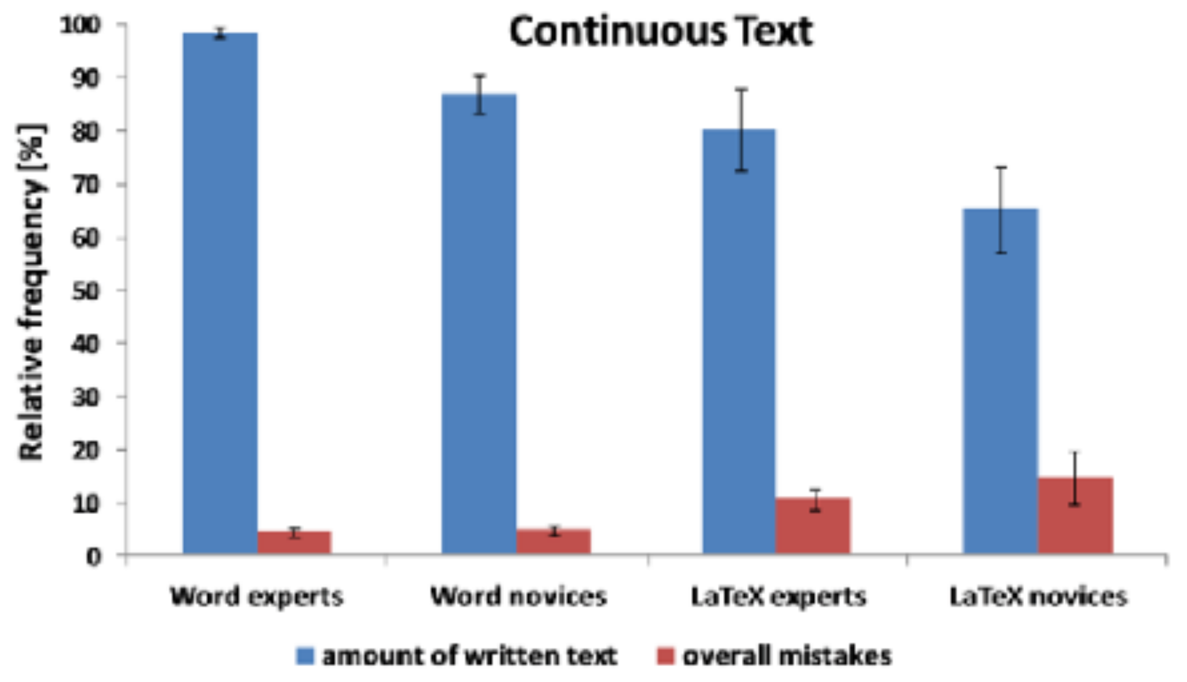


Fig 4. Mean amount of text written within 30 minutes and the overall number of mistakes for the *continuous text* for the four groups of participants (Word experts, Word novices, LaTeX experts, and LaTeX novices). Error bars represent the standard error.

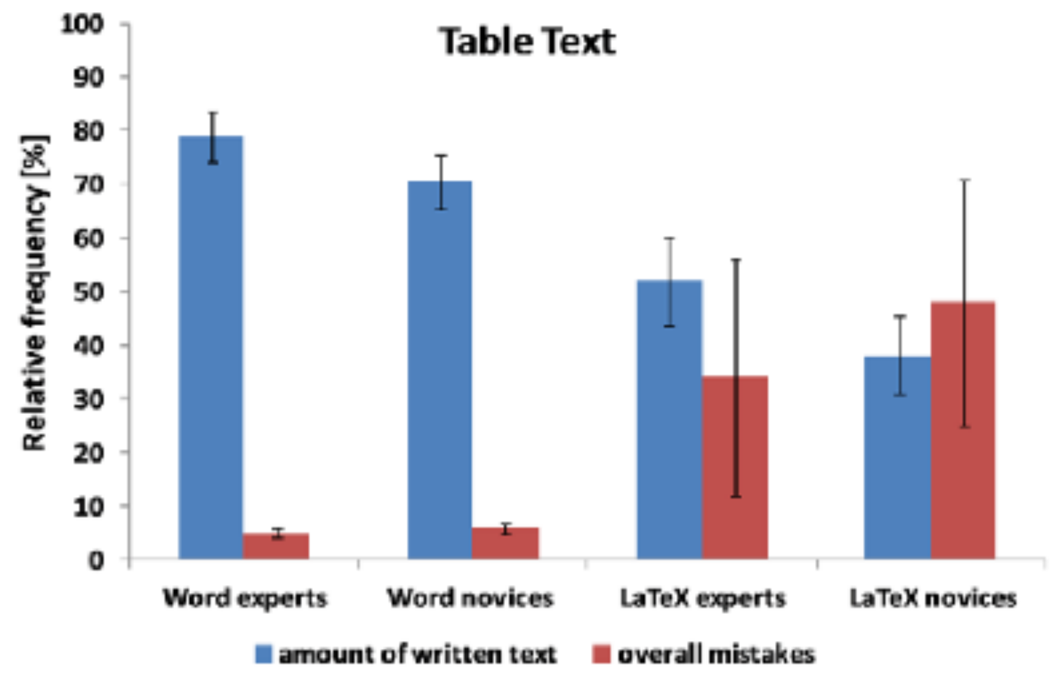


Fig 5. Mean amount of text written within 30 minutes and the overall number of mistakes for the, table text for the four groups of participants (Word experts, Word novices, LaTeX experts, and LaTeX novices). Error bars represent the standard error.

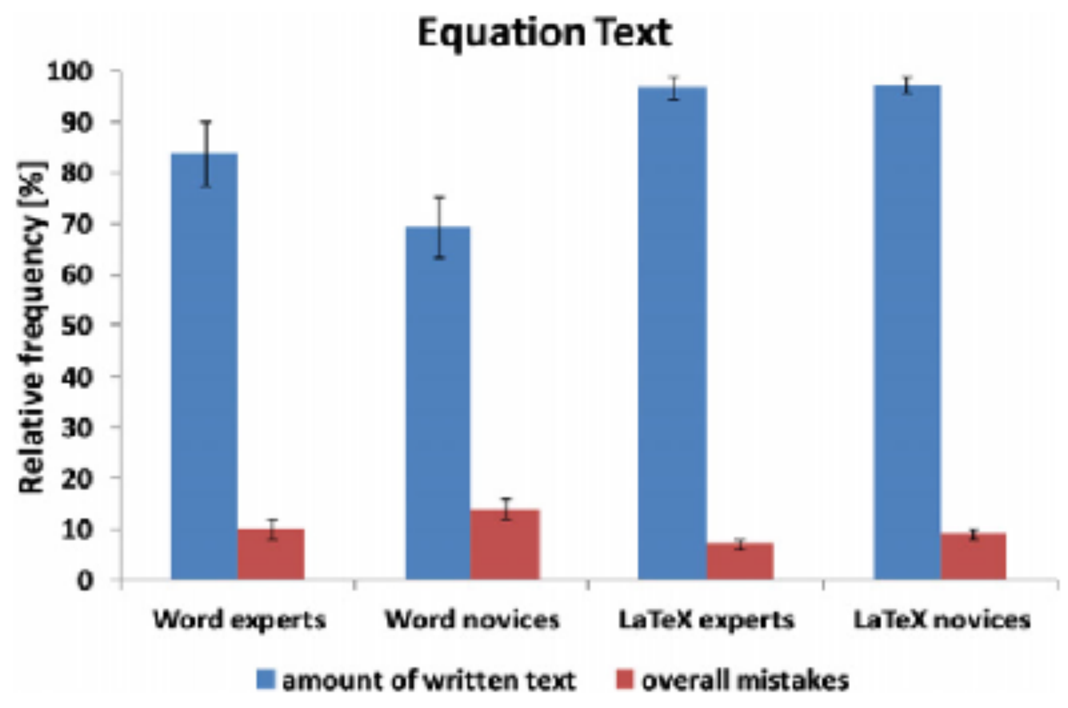


Fig 6. Mean amount of text written within 30 minutes and the overall number of mistakes for the equation text for the four groups of participants (Word experts, Word novices, LaTeX experts, and LaTeX novices). Error bars represent the standard error.

N = 10 / group
"Novices" <500 hr
"Experts" >1000 hrs

14 females, 26 males
Physics: 12, Psychology: 5, Computer Science:4, ...

https://github.com/seunggookim/clied

Slides PDF & text samples

Time for (more) questions & discussion! 🤓