



# 技术时代的生活与工作

Tech among Life and Work

作者: Qingsong Liao

组织: Free Club

时间: December 3, 2025

邮箱: llqingsong@qq.com



# 目录

<b>第一章 被技术改变的生活 Tech Life</b>	<b>2</b>	<b>参考文献</b>	<b>40</b>
<b>第二章 学技术的哲学 Tech Philosophy</b>	<b>5</b>	<b>附录 A 开发中遇见的各种软件问题 TODOList</b>	<b>41</b>
2.1 学习什么技术 What Tech . . . . .	5	A.0.1 General Contributions . . . . .	41
2.2 怎么学习底层技术 Low-level Tech . . . . .	5	A.0.2 <b>TODO</b> write a eww-hook, when I use eww to visit a url, and if the site(not local file) exists more than 5 minutes, then download it to a dir I specific. . . . .	41
<b>第三章 编辑器哲学 Editor</b>	<b>7</b>	A.0.3 <b>TODO</b> how to customize face as for specific mode? . . . . .	41
3.1 如果人生还有一次, 我还选 emacs . . . . .	7	A.0.4 <b>TODO</b> Advent of code in zig? . . . . .	41
3.2 我告诉你, 年轻人啊, 学习老掉牙的 emacs, 就像是金庸小说里的年轻主人公学习失传的武林秘籍一般(只是小说剧情而矣). . . . .	8	A.0.5 <b>TODO</b> package leego elisp package as emacs package . . . . .	41
<b>第四章 操作系统哲学 Operation System</b>	<b>18</b>	A.0.6 <b>DONE</b> how to have pop out behavior like cape, but for my function's output? . . . . .	41
4.1 我的操作系统特点 . . . . .	18	A.0.7 <b>FIXME</b> How to modify org-mode's #+INCLUDE: . . . . .	42
4.2 用 linux 只为戒色 . . . . .	18	A.0.8 <b>DONE</b> how many hours of pc idle's power consumption is equal to hibernate save ram to disk? . . . . .	42
4.3 学技术只为戒技术 . . . . .	19	A.0.9 <b>DONE</b> almost same vscode-emacs package. . . . .	42
<b>第五章 语言哲学 Language</b>	<b>20</b>	A.0.10 <b>DONE</b> how to run current region's content (if not in region, run that line) into current project's vterm .	42
5.1 英文比不上那方块字 . . . . .	20	A.0.11 <b>DONE</b> pyim % 字符触发 Not enough arguments for format string .	43
5.2 什么? 编程语言也是语言? . . . . .	20	A.0.12 <b>DONE</b> explain this wget command .	43
<b>第六章 键盘流哲学 Keyboard</b>	<b>21</b>	A.0.13 <b>DONE</b> how to do emacs overlay in nixos to config not to have somepackage natively exclude . . .	44
6.1 设计哲学 . . . . .	21	A.0.14 <b>DONE</b> using consult-gh-notifications, and I just type wrong pass as my own pc's pass, not github token . . . . .	44
6.2 分层 . . . . .	22	A.0.15 <b>DONE</b> how the ~/.mozilla/firefox/profiles.ini looks like? . . . . .	44
6.3 本地刷录 . . . . .	22	A.0.16 <b>DONE</b> add sway emacs pkg? . . . .	44
6.4 不只键盘 . . . . .	25	A.0.17 <b>DONE</b> how to make rm safer? . . . .	44
<b>第七章 屏幕哲学 Screen</b>	<b>27</b>	A.0.18 <b>DONE</b> recentf moving file persisnt	44
7.1 Feels: . . . . .	27		
7.2 Features: . . . . .	28		
7.3 Tips for mono: . . . . .	28		
<b>第八章 fog.h</b>	<b>31</b>		
8.1 Quick Start . . . . .	31		
8.2 Just rewrite shit in C, Just rewrite shit in C — tsoding . . . . .	31		
<b>第九章 Do AoC in Zig Learn Zig in AoC</b>	<b>32</b>		
9.1 Why Zig . . . . .	32		
9.2 Learn Zig . . . . .	32		
9.3 Do AoC . . . . .	32		
<b>第十章 版本更新历史 Version</b>	<b>34</b>		

---

A.0.19 <b>DONE</b> pull request to the porn site list project . . . . .	44	A.0.24 <b>DONE</b> creat new account's github repos for github.io . . . . .	48
A.0.20 <b>DONE</b> make Leetcode fresh tasks list . . . . .	46	A.0.25 <b>DONE</b> upload Purezen to github .	48
A.0.21 <b>DONE</b> learn how to reference in org	46	<b>附录 B 嵌入式软件招聘常见要求 Embedded Jobs</b>	<b>49</b>
A.0.22 <b>DONE</b> upload large files into the github repo . . . . .	46	<b>附录 C 嵌入式招聘常见笔试问题</b>	<b>50</b>
A.0.23 <b>DONE</b> try to package real-mono- theme to melpa . . . . .	46	<b>附录 D 嵌入式招聘常见面试问题</b>	<b>51</b>

# 序章 技术何为?

## 内容提要

- |                                |                           |
|--------------------------------|---------------------------|
| □ 任务编号：以名为“廖青松”的男婴为身份出生在重庆普通家庭 | □ 完成难度：简单到极难，活着也没 BOSS 啊？ |
| □ 开始时间：2002-11-04              | □ 需要小时：一生，大约三万天           |
| □ 结束时间：总有一死                    | □ 本章要点：我对于技术与人生的看法        |

世界嘈杂<sup>1</sup>、瘾品横流、随机与即时构成新的枷锁，而真正的自由来自主动的剥离。当目光离开诱惑、离开广告、离开无意义的瞬息刺激，心才开始变得干净。生活越简，能量越纯；越慢，感受越深；越少，越能看见真正的自己。

身体是意志的第一块土地。空腹的清明、弱光的安静、低温的醒觉、缓慢进食的耐性，都在一点点重塑我们早已被工业习惯磨钝的感官。行走、奔跑、提举、拉起、俯卧撑、壶铃、农夫行走——这些最朴素的动作让人重新理解力量的意义：力量不是爆发，而是日复一日不受伤、不懈怠、让心跳稳、饮食亦是自律的延伸。

避免加工。让精神长的那种沉稳、远离高糖盐油，回到豆果菜、坚果与发酵的本味，让身体习惯真实的能量，而不是被化学甜味与工业脂肪驱使的假饱与假快乐。克制不是苦行，而是温和地恢复本能。

至于技术，真正的价值不在追逐流行，而在深入底层、理解根本。用 Emacs<sup>2</sup>，不是为了高效，而是为了与世界拉开距离，与自己靠得更近；不随机、不即时、无广告、无噪声，是一种长期的心性训练。读 LFS、LKD 与 SOC 文档，写驱动、调性能、跑 QEMU、玩内核、读源码、用 LATEX 和 Org 写书，是为了获得一种“我真的懂了”的安静感。而这种懂，不是为了炫耀，不是为了沉迷，而是为了让工作成为谋生技能，让生活成为真正的生活。

真正的智慧是：学会技术，然后把技术放下；拥有力量，然后让力量变得温柔。生活是吃饭、睡觉、读书、编程、走路、壶铃；生命是健康、乐观、会意、精进、闲适与稳稳的力量。

世界广袤、事物繁多，而心若清澈，幸福忽然变得极小，也极近——不来自外界，只来自自身安静而坚定的内心。



图 1: free-your-soul

<sup>1</sup>第一个嘈杂的提示，如果你进入了 HTML 版本，请[点此进入 PDF 版本](#)。

<sup>2</sup>本书的牛“图片”均出自于 GNU.org 的自由艺术。GNU 计划，又称革奴计划，是由 Richard Stallman 在 1983 年 9 月 27 日公开发起的，目标是创建一套完全自由的操作系统，将牛作为 gnu 和 emacs 的象征是因为 gnu 在英文中本身就有“牛羚, 角马”的意思，有趣的是 gnu 本意为 Gnu is Not Unix…再问里面的 gnu 什么意思？还是 Gnu's Not Unix—a recursive acronym，就像是中国的俗语“山上有座庙”，无穷尽也。

# 第一章 被技术改变的生活 Tech Life

-10000000005 年，一切都变了。宇宙诞生了。

-5600000005 年，一切都变了。太阳诞生了。

-4600000005 年，一切都变了。地球诞生了。

-200000005 年，一切都变了。生物诞生了。

-80000005 年，一切都变了。恐龙诞生了。

-30025 年，一切都变了。人类诞生了。

-26025 年，一切都变了。语言诞生了。

-25025 年，一切都变了。家庭诞生了。

-22025 年，一切都变了。农业诞生了。

-20025 年，一切都变了。文明诞生了。

-10025 年，一切都变了。文字诞生了。

-9025 年，一切都变了。货币诞生了。

-8025 年，一切都变了。法律诞生了。

-1025 年，一切都变了。书写诞生了。

-225 年，一切都变了。哲学诞生了。

-100 年，一切都变了。集权诞生了。

-25 年，一切都变了。数学诞生了。

-5 年，一切都变了。宗教诞生了。

1000 年，一切都变了。宗教火热。丝绸之路、黑死病、天花、中世纪、疾病、皇帝、国王、奇迹、骑士阶层、武士、城墙、封建帝国、马鞍…

1279 年，一切都变了。文艺火热。宋朝灭亡，造出火药，指南针，造纸术，印刷术的中华文明终结。  
先进技术被蛮族的武力消灭，但技术开始了它的崛起…

1600 年，一切都变了。帝国火热。欧洲人发现了美洲。帝国主义、殖民主义、印刷术、土豆、巧克力、辣椒、枪械与大炮、地理大发现、数学与工程学…

1700 年，一切都变了。烟草火热。哥伦布大交换的阴影笼罩大地。黑奴、民族国家、地理学、历史学、语言学、生理学、资本主义、大英帝国…

1800 年，一切都变了。鸦片火热。资本主义的阴影笼罩大地。共产主义、物理学、汽车、文学、工业革命、开发美洲、煤气、火车…

1900 年，一切都变了。技术狂热。技术在欧洲大地遍地生根。工业化、城市化、流水线、民主、飞机、电力、公司、灯光…

1915 年，一切都变了。一战结束。技术的阴影笼罩大地。农药、化肥、化学、媒体、通讯、武器、内燃机、全球化、电报、相对论…

1945 年，一切都变了。二战结束。技术的阴影笼罩大地。女权、核弹、雷达、战斗机、计算机、工业化、旅游业、赌博业、广播、量子物理…

1965 年，一切都变了。编程火热。技术停滞或是爆发？电视、摇滚乐、基因学、编程理论、月球、民用航天、芯片、电子技术、教育扩大…

1975 年，一切都变了。贸易火热。中国迎来开放。技术在中国大地遍地生根。留守儿童、大量生产、环境污染、贫富差距、道德沦丧、城乡差距…

1995 年，一切都变了。网络火热。技术迎来它们的终极载体。科学知识/各种思想文化加速传播、游戏、营销、贸易、色情、视频/图片/电影/音乐…

2002 年，一切都变了。我诞生了。技术/成瘾/财富/变革的高峰在我这代中国人最能体现，从未体验过的事物比比皆是。从小时候看电视到青少年时的玩电脑再到成年后被各种技术主动或被动控制

---

…这代人注定是分化的一代。

2025 年，一切都变了。智能火热。技术停滞或是爆发？社交媒体、区块链、转基因、移动支付、智能设备…

2100 年，一切都变了。死神永生。技术黑暗或是造神？火星旅行、人工智能、机器人、脑机接口、基因修改…

1902 年出生，1910 年化学，1925 年通迅交通，1950 年核弹计算机，1970 年克隆羊，1992 年互联网？2002 年出生，2010 年手机，2025 年 ChatGPT，2050 年 AGI，2070 年具身智能，2092 年永生？平均寿命 73 岁？我大概是能苟到 90 岁？也许会有一场战争…灭绝不应该长生的人。

- 后现代技术要点：(上瘾/传销/非人)

1. 靠人传销，如加密货币，房地产拼多多。(利用心理/行为设计/资本主义自由市场)
2. 不要人类，如人工智能，通讯与机器人。(利用物理/自动化的算法与机械/大数据)
3. 让人上瘾，如社交媒体，烟酒赌博旅游。(利用生理/产量强度更大/劳动时间减少)

- 硬技术要点：医学人工智能(癌症)-> 脑机接口(义肢)-> 机器人载体(永生)-> 恒星际(星际)

- 医学大数据-> 大模型-> 生理大模型-> 基因建模？

- 脑机需要什么？生物 + 数据

- 机器人需要什么？算法 + 数据

- 恒星际？能源 + 机器人

- 问题：

- 伦理性：如果都能永生，如果有谁不能呢？
- 可靠性：基于数据，准确性不高，如果程序出错呢？
- 社会性：人类会成为什么样？幸福与快乐到底是什么？

- 悲观主义者：

- 衣/食/住/行/性…

- 以前：自制/采集狩猎/自建/只有走路/真情或强暴
    - 现在：拼多多/拼好饭/烂尾楼/倒闭国产新能源/彩礼八十八万

- 意义感和存在感

- 以前：靠身体劳作生活，自己的劳动成果，直接贡献在自己身上
    - 现在：靠国家/公司/投资生活，靠机械化身体动作/脑力说唱生活

- 人类命运与未来

- 以前：自己的命运在自己/家族/别人手上
    - 现在：自己的命运在政府/老师/老板/网友…手上

- 乐观主义者：

- 衣/食/住/行/性…

- 以前：难得/难管/土房/5 公里内/不可控
    - 现在：便宜/便宜/商品房/环游世界/易得

- 意义感和存在感

- 以前：迷信宗教与古语
    - 现在：更多知识与卫生

- 人类命运与未来

- 以前：狮子老虎毒蛇虫山路水沟低温
    - 现在：温暖舒服平静安定可控多彩闲适

- 对平常人而言，现代生活即是：

- 更少的劳动，更多的消费
    - 更少的暴力，更多的新闻

- 
- 更少的情感，更多的媒体
  - 更少的迷信，更多的知识
  - 更少的空闲，更多的享乐
  - 更少的体力，更多的智力
  - 更少的杀戮，更多的竞争
  - 更少的疾病，更多的人口
  - 更少的歌舞，更多的节目
  - 更少的宁静，更多的噪声
  - 更少的饥饿，更多的肥胖
  - 更少的家族，更多的国家
  - 更少的闲适，更多的焦虑
  - 更少的长久，更多的变动
  - 更少的现实，更多的未来

2025 年，一切都变了。我不再相信技术/社会/科学…我打算出去种田。

2035 年，一切都变了。ChatGPT，我该生孩子吗？下面是我的个人信息:xxx

2045 年，一切都变了。伟大的 ChatGPT 系统，今天您的系统算出应在人体培养室中产出多少个孩子？

2065 年，一切都变了。…101110101111101010101010000111101010111…

10100101110010111110101010101000011110101011100000000000000…

42 时间 000000000000000000 宇宙 1 太阳 56 智能 99 人类 99 历史 99 人 99 年 …

赚钱，存肌肉，学习技术，准备永生？不知道，但我们得先学习在技术时代怎么生存。毕竟，本书书文即为《技术时代的生活与工作：Tech among Life and Work》，主要写写自己的技术学习，写写技术见闻，写写人类与机器的命运。地球与这些文字一样，都小如宇宙中的沙子，可就算无人观赏，也是一种美丽。我在沙子上写着沙子，不觉渺小而觉其之美，闪闪的一点有多少国王与伟人的故事，一切尽此矣。在无限的宇宙舞台上，熵增即是技术之艺术。当星星暗淡、黑洞蒸发、文明消亡，宇宙留下的，唯有技术。

# 第二章 学技术的哲学 Tech Philosophy

## 内容提要

- |                    |                               |
|--------------------|-------------------------------|
| □ 开始时间: 2025-11-19 | □ 需要小时: 100+                  |
| □ 结束时间: 2026 年前    | □ 本章要点: linux/c/zig/rust/rtos |
| □ 完成难度: 难难难        |                               |

人没有梦想那和咸鱼有什么区别呢? ——周星驰

## 2.1 学习什么技术 What Tech

表 2.1: path-to-know-tech

process	url	for
fag	flag/cintro	c, pretent iam tsoding
embedded C book	read book	c, how compiler/os/c works
Yeetcode	leetgo/book	cpp[rust], algo ds, solve task
paperlike-c/el	paperlike-go	elisp,paperlike emacs controller
ziglings	ziglings	zig, basic ziglangs speed run
zag	fag	zig, I have language erotic
paperlike-zig	paperlike-el	zig, make cli/tary
nixos r2s	github-repos	nix, for network addiction
TsurgizOS	os.phil-opp	zig[rust], make general os for cv
nixos rasberry-pi	github-repos	nix, for embedded os/screen/driver
clings	ziglings	c, lings but clang
freertos emulator	rtos	c, use general rtos
lvgl eink rtos	lvgl	c, embedded ui driver/sdl
lvgl lora loc	graphic	c, embedded openstress/sdl/lvgl
RZOS	rtos	zig, make general rtos
Celest	game	zig, embedded game/sdl
Safephone	electronic	lvgl/eink/openstress/lora/3Dprint stm32/nix/electronic/network nsfw image/text detected

1. All in all, it's for 技术哲学
  - (a). 网络/低速/时间/域名/ai 过滤/linux 内核构建-r2s
  - (b). 屏幕/护眼/低成瘾/驱动设计/eink 屏幕算法-paperlike
  - (c). 通信/安全/无依赖/去平台/lora 远距离通信-safephone
  - (d). 交通/电工/电子电路/openstress-去五佰<sup>1</sup>那里学修车

## 2.2 怎么学习底层技术 Low-level Tech

### 1. 第一阶段: 打基础 (C / Linux / GCC / Emacs)

- C 语言: 学会指针、内存管理、结构体、函数指针; 刷一些小项目, 比如实现 malloc、shell、http server。
- Linux 基础: 熟悉命令行、文件系统、进程/线程、信号、管道、套接字。
- GCC: 学会编译流程 gcc -E/-S/-c/-o, 理解预处理、汇编、链接, 玩一下 objdump 和 nm。
- Emacs: 把它当 IDE 来用, 掌握基本编辑、调试、补全、LSP 支持。

<sup>1</sup>我父亲那边的车行亲戚

## 2. 第二阶段：系统调试与逆向（GDB / QEMU）

- GDB：练习断点、单步、查看寄存器/内存、调试多线程/远程调试。
- QEMU：
  - 用它运行 Linux kernel 或裸机程序。
  - 学会 qemu -S -s + gdb remote 调试，体验调试内核的感觉。
  - 研究 QEMU 的设备模拟（比如 VirtIO、PCI），理解虚拟化和硬件抽象。

## 3. 第三阶段：进阶编程语言与系统（C++ / Zig / RTOS）

- Zig：Zig 是现代系统编程语言，学习它的构建系统、内存模型，可以和 C/C++ 混合编程。
- RTOS：从 FreeRTOS 入手，学任务调度、中断、任务间通信（队列/信号量）。可以用 QEMU 模拟 Cortex-M 板子跑 RTOS。

## 4. 第四阶段：融会贯通（大型项目 / 内核 / 编译器）

- QEMU + GDB：调试内核启动、写内核模块。
- 编译器开发：研究 GCC 或 Clang 的前端/后端；或者用 Zig 写一个简化编译器。
- 个人项目：比如写一个简易 RTOS，或者在 QEMU 里跑自己写的内核。

## 5. 操作系统构建与升级（Yocto/Android/内核）

在树莓派上交叉编译 Linux 内核，修改驱动或设备树（Device Tree）进行硬件适配。尝试用 Yocto 或 Buildroot 构建自定义 Linux 镜像，加入自己编写或修改的驱动模块。安装和编译 LineageOS（Android for Pi）或类似 Android 系统，修改系统服务或 HAL 层。实践 OTA（Over-The-Air）升级机制，模拟系统升级和回滚操作。硬件 Bringup（CPU/GPU/Memory/Peripherals）利用树莓派的 GPIO、SPI、I2C、CAN、PWM 等接口，练习外设 Bringup 和驱动调试。连接摄像头模块、显示屏（LCD 或 E-Ink）或音频模块，编写驱动和控制程序。使用 perf/ftrace/gprof 分析 CPU/GPU 性能瓶颈，优化程序调度。

## 6. 系统稳定性与性能优化

构建多线程/多进程网络服务，使用 socket 编程实现客户端/服务器通信，模拟并发场景。在树莓派上测试高负载条件下的系统稳定性，分析内核日志、内存占用和 CPU/GPU 使用率。实践内核参数调优，如调节调度器、内存缓存策略，观察性能变化。

## 7. 客户功能定制与基线升级

自己设计一个树莓派应用（如小型智能家居控制器或信息显示终端），从硬件 Bringup 到应用功能定制完整流程。模拟不同版本的系统镜像管理，练习分支合并、基线升级和版本回退。

## 8. 加分技能训练

- (a). 低功耗优化：通过关闭不必要的外设、调节 CPU/GPU 频率或使用 E-Ink 显示屏练习功耗控制。
- (b). 虚拟化/容器：在树莓派上安装 QEMU/KVM 或 Docker，运行多系统虚拟环境，模拟嵌入式应用部署。
- (c). 芯片平台经验：如果有 MTK 或其他 ARM 板卡，可以对比树莓派练习移植和平台适配经验。

# 第三章 编辑器哲学 Editor

This is my blog and p(pdf)log which written in org and LATEX, and emacs/nixos config, code snippet and even more, all in just one repo. 这是仓库是我的博客也是我用 LATEX 写的书，还是我是 emacs 和 nixos 配置，代码模板等等。

```
(org-babel-do-load-languages  
'org-babel-load-languages '((emacs-lisp . t)(shell . t)))
```

```
if [[ -f "/home/$USER/.emacs.d" ]] then  
    git clone --depth 1 https://github.com/jamescherti/minimal-emacs.d ~/.emacs.d  
fi  
ln -sf $PWD/nixos/ ~/.config/  
ln -sf $PWD/snippets ~/.config/  
ln -sf $PWD/nixos/hmdz.pyim ~/.config/  
ln -sf $PWD/nixos/post-init.el ~/.emacs.d/  
if [[ $XAPIAN_CJK_NGRAM != "true" ]] then  
    sudo cp /etc/hardware-configuration.nix $PWD/nixos/hardware-configuration.nix  
    chmod 777 $PWD/nixos/hardware-configuration.nix  
    mkdir -p $HOME/.config/sops/age  
    cp .keys.txt $HOME/.config/sops/age/keys.txt  
    sudo nix-channel --add https://nixos.org/channels/nixos-unstable  
    sudo nix-channel --update  
    sudo nixos-rebuild switch --flake /home/$USER/.config/nixos#$hostname --option substituters '  
        https://mirrors.ustc.edu.cn/nix-channels/store https://cache.nixos.org' --cores 6 -j 12;  
fi
```

**Listing 3.1:** 我能用这代码块重现我的电脑，怎么样？帅吧！

这是 org 的文学编程能力最简单的体现，从 c 到 bash，在一个 org 文件中能一起运行，神奇吧！为什么 emacs 这么强大？大概就是因为它使用的 elisp 相当“灵”吧，再说为什么 emacs 这么小众，也大概是它的 elisp 太“难”了吧。有些人不敢学，怕自己投入又得不到钞票，有些人不能学，他们把自己的心给了别的教派如“vim”“vscode”于是固步自封，自高自大，vim 模态天下第一！我曾同样如此，抱着网上破烂配凑出的 neovim，对那些大神的操作流口水但是又没法认同 emacs。其实犹豫不定很正常，进入一个陌生的环境，很多怪人怪事，emacs 更是如此，你常常不知道那些人口中说的 eglot/flycheck/eww 是什么？常常写错一些代码。当然这同样也是学习 emacs 的乐趣，也正是它的陌生带来的新鲜，而且有了 chatgpt 的帮助，学习起来相当方便。

## 3.1 如果人生还有一次，我还选 emacs

我最早听闻 Emacs，大概是 2022 年在 CSwiki 中作者对 Emacs 的溢美之词“神用编辑器”<sup>1</sup>。当时我猎奇心甚为严重，又觉得大家都在用的 VSCode 无法体现我的气质，便先折腾三年 Vim/Neovim/Helix 后在 2024 开始自学 Emacs，接触 Emacs 先是 doom 框架后是接近原生的 purcell。喜欢换来换去，或许这是年青人的通病。

现在，我已不再年青，却依然喜欢 Emacs 就像我曾喜欢一个女生，但她从来不知道。曾经沧海难为水，或许这是人老了之后的通病，而我觉得更可能是因为 1978 年诞生的 Emacs 依然年青。

<sup>1</sup>当然还有大佬们（我）的溢美之词：Donald Knuth 和 Python 的蛇叔说用 Emacs 就像是开了几十年的小车/Rust 的语言之父（早跑路了）一天 60% 时间在 Emacs 中/Ruby 作者道“Emacs 改变了我的人生”/julia 大概所有函数式语言的作者都得用 Emacs…Java 的高司令自己写了一个新 Emacs，Linus 自己维护了 MicroEmacs 并表示自己安不來那些“有语法高亮”的编辑器。还别说那些库程序员和知名开发者了，CURL/FFMPEG 等作者，FFMPEG 的天才程序员也开发了 QEMU/先进的 Pi 算法/用 LLM 作压缩工具，没错他也写了个 Emacs，像是编程 YouTuber 和游戏开发者们，Tsoding/JonathanBlow/HandMadeHero 在直播间用起 Emacs 是像开了挂。中国前 deepin CTO 懒猫/小米总监等等也同样的 Emacs 用户。更别说 Richard Stallman 了，他开创了 GCC/GDB，也开始了 Emacs 和 GNU，开源世界自君开。总之，Emacs 是个好玩的计算机历史产物，似乎仍没有过时。

## 3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

1. 要我说认为我的 Emacs 配置有什么特点，大概就是这样几点吧。

- minimal: my config based on the clean and fast minimal-emacs.
- zen: a pure white and black theme written by me called "real-mono-theme".
- reproducible: package manager is using **NixOS**.
- simple: just a post-init.el which less than 2000 lines.
- keyboard: for my **ZMK** config, only me can use those as flow~
- efficient: elegant selecting like queen and fast editing like king.

2. 要我说 **Emacs** 就和与小说里秘传的十八般武林功夫一般，有定身/移形换影/透视经脉/必杀技/念经/修炼/经文/拜师/法宝/女伴/侠义…甚至武功大概不过如此…拳脚身头可用的功夫，那里有无限可能的电子计算机的空间大啊？

- 动：插入/切换/输出/笔记/补全/专注/跳转/投修/替换/扩张/行修/选择/整理/重复/变化/比较/潜在
- 阅：简用/单页/全页/源码/图书/用文/说文
- 用：项目/虚拟/日志/中文/终端
- 浏：搜/库/问
- 库：Linux/Libc/Posix/C ABI/Network/Algo/Protocols
- 标题说的有定身/移形换影/透视经脉/必杀技/念经/修炼/经文/拜师/法宝/女伴/侠义，我也确实有的定身 view-only/移形换影 switch-buffer/透视经脉 describ-function/必杀技 kill-emacs/念经 read gpl3/修炼 rtfm/经文 info/拜师 rms/法宝 list-package/女伴 WoMan/侠义 contributor。emacs-china 管论坛的用户叫“道友”，似乎就已经点明这点，学习难度大，学习结果虚幻无比，需要气定神闲，天地之间少有人闻，更少有人真正能掌握这一门技术，不叫“道友”怎么说得过去？

### 3. Write The Fucking Code

- 心写：org/theme/hide-xx/ligature/no-xx/vertico-flat
- 补写：cape/hippie-expand/snippet
- 跳写：occur/rg/eglot/imenu/mark/avy/consult/compilation
- 横竖：flush/delete/duplicate/sort/move-text || iedit/mc/rectangle
- 比缩：diff/ediff/xxx-diff/xxx-merge
- 选扩：surround/expand-region/native-mark/mc
- 选点：embark/vertico/menu-bar/tooltips/tool-bar
- 重复：macro
- 切换：dired/find|switch-file/sway/ibuffer/tramp
- 替换：replace string|regex
- 变换：shift-number transpose/case-switch/narrow
- 内在：save-xxx/xx-mode/envrc/editorconfig/autofmt

### 4. Read The Friendly Manual

- 源码：eglot/helpful
- 用简：tl2r
- 用文：man/woman
- 一页：devdocs
- 全页：eww/nginx
- 说文：info
- 图书：nov/pdf-tool

## 5. Browse The Enshitty Web

- 搜索: tavy
- 仓库: consult-gh
- 询问: gptel
- 交流: gnus/irc/eww

## 6. Use The Minimal Tools

- 工具: magit/git-xx/project/nix/docker/kubernetes
- 中文: quicksdcv/pyim
- 日志: syslog|journalctl-mode
- 终端: vterm/repl/shell

## 7. 总之，我试着去思考一下什么是写作与编程，阅读与浏览，内心与外物

- 写作: 需要重复阅读，反复遣词造句，思考上下文人物/事件关系，写作有编辑环节，但那是“编辑”做的事。
- 编程: 需要更多更快的查找方式，更多更快的浏览，思考调用/协议的关系，编程需要大量的编辑 (indent/上下移动/复制粘贴)。
- 阅读: 长文/长时间的专注，资源越少越好，写作应该减少阅读别人的作品。
- 浏览: 短文/短时间的专注，资源极多…想不多都不行…编程应该大量阅读别人的作品。
- 内心: 得之心而应之于手，很多优秀文学作品使用笔和纸写作，计算机都得用键盘，减少了肌肉的使用本身就会带来人的记忆损失。
- 外物: 但编程本身就越来越重“外物”，所使用的事物大概都是其他人所写，要依赖上游，依赖工具。写作需要的就只是笔和纸…相对的 编程，大概没有 10 个以上的工具是做不完备。
- 写作与编程，
  - 一个大概，一个精确
  - 一个阅读，一个浏览
  - 一个重内心，一个重外物
  - 一个写活人，一个写死物
  - 现在的人连读书都不愿意了，怎么可以会编程？
  - 唯一的原因: 写代码来 钱
  - 可钱又带来虚无…
- Emacs 是让编程更像写作的一件事。
  - 减少拼音带来的“任务切换占用”，emacs 因为全能，完美替代 bloate 的像搜狗/fcitx 那样的系统输入法，再使用五笔之类的输入法，又因为 emacs 的完全可配置性，使用 pyim 能只看得到新字不断产生，极大减少了需要选词和弹来弹去的心理负担。
  - 减少鼠标带来的“肌肉使用减少”，emacs 因为按键复杂，替代使用鼠标点击，提高程序员的编辑效率，复杂的使用方式也提高程序员的大脑容量，复杂的按键双手肌肉记忆的交替快速使用提高了程序员的肌肉使用。
  - 减少依赖，emacs 需要的依赖极少，甚至因为 emacs 的多平台可用性，用好 emacs 可直接移至更多系统，而不用像用 vim 之类的软件担心自己 shell/term 不适配新系统。
  - 减少控制，越古越好，1978 年后，技术很大程序控制了人类，在那个年代，RMS 就提出“要么技术控制人类，要么人类控制技术”，这句话放在现在越来越明显，走在街上，走路手机僵尸，看导航的骑车僵尸是越来越多了。
  - 减少分心，虽然比起写字来说没有那样的一笔一线的专注，但比较 VSC 那样时不时弹出“M\$”提示的软件，emacs 可以说是绝对的安静了。
  - 为 emacs 写代码不是为钱
  - 为 emacs 写代码不是写死物

- 为 emacs 写代码是内心的平静
- 为 emacs 写代码是真诚的阅读
- emacs 让编程更像是翻页写作
- vscode 让编程更像是浏览网页

## 1. 我的编辑器邪教之旅：

浏览器里的 vimium C，因为 wsl 学习的 neovim，因为换上 nixos 后抄的 helix 配置，因为大四没找工作太闲学的 emacs。我每用一个就大呼“我草，早知道，还得是 xx！”，然后给同学老师家人都安利安利，给同学说“我有一宝”，给老师说“看我操作”，给家人嘻嘻傻笑像是捡到五百钱。虽然这代表着我非常的闲，但是折腾 emacs 确实让我不是很闲到以至于抑郁，我还是在用 Vimium C 和 emacs，赋予我闪电切换页面的神奇体验，当然，代价是它们也让我得了一种名为“换页面换 buffer”的 ADHD，再者，其实不用 vim/emacs 也会得，只不过叫作“懒得换页面就直接玩手机”ADHD，“懒得折腾电脑就去玩 CSGO 成了换弹癌”的 ADHD。在这个时代，玩 csgo 和玩 emacs 有什么区别呢？一个能炒皮肤，另一个能折腾皮肤，如是而矣。

## 2. 我使用 emacs 没有太多原因就只是：

- (a). 依赖少，之前用 vim，我就要关心 vim 键位的 app 适合用来看 pdf，看 epub 看文档，整 vimium 和 vim 一个键位/什么 lazygit/tmux/fcitx/alacritty，还生怕它们之间有什么矛盾，关键 vimscript 也不适合作为灵活的语言，整天很困扰很憋屈，我还折腾 colemak、虎码和 nixos，我的天，对于任何一个大脑快定型的成年人来说，这简直是噩梦！幸好有 emacs 解救我于快捷键地狱中，装包能 list-package 再一点就安装了，pyim 配置虎码就一行的事，那里还要 fctix5 怎么怎么样！自从用了 emacs，我再也没有折改过 firefox，一是我没时间，二是我就只用 emacs 的 eww，我也就不再在乎什么 vimium 的键位和 vim 不一样了，**能用就行才是真理！**再说用上了 elisp，那才叫美梦成真，想写一个工具简直就比玩手机打游戏等更有意思多了。什么 tmux/lazygit/nixos，现在我只想笑笑，我用 vterm/magit/任何平台都能跑 emacs 不是爽多了吗？只要给我 emacs/编译器，写代码就是轻轻松松的事！emacs 改好了我的强迫症和极多主义，现在我心宽又极简，感谢 emacs！
- (b). 功能多，之前用 helix，虽是开箱即用但是很明显它的功能只限于写代码，快是快但是对我这样的学习困难型人来说，快和慢没有多大的区别，只会让我再难受于“我好慢它好快，我是不是 x 痞了”，helix 是个自带丰富功能工具，但是对于习惯 homerow 的人来说其实用 vim mode 没有什么增益，只会让我更迷茫“我能直接不动手就能上下了，干啥还要这个 hjkl??”。有插件和服务器模式这点就足以胜过 helix 了。当然我从 helix 的文档也学到了很多编辑知识，如果没有它的 emacs 主题和打算用 scheme 写插件系统，和让我折腾很久的 helix eink 主题，我也没有潜移默化现在的编辑技术和极简主题审美了。
- (c). 分心小，之前用 vimium 那就是一个换换换的感觉，总是想去换页面，看看新闻看看新工具，上下跳转，左右腾挪，helix 功能太少不能在里面呆太久，而 vimium 是个浏览器插件在里面呆着又太容易去娱乐，用 vim 主要是去 youtube 参加编辑器神战，helix 也好 vimium 也好。其实我学习 emacs，到能快速掌握 avy 靠得是之前用过 vimium，学习 surround/expand 靠的是 helix，能熟悉 info/man/tldr 靠得是之前在终端里学习 vim。总之 emacs 必然是编辑技术的一个终点站。
- (d). 定制化，没有一个 term emulator 支持比例字体的，只有 emacs，也有人说了为什么用 bookerly，就用 vscode 的什么 ubuntumono/nerdfont 不好吗？这我得确实说：“非等宽字绝对好用”，为什么空格得道 M 一样宽，为什么 l 和 W 一样宽？现在编程/终端处处都是等宽字体，很多人在根本就没用过的情况下痴迷在各种 nerdfont 间切换并坚持自己的“信仰”，这种信仰只有在用了一个优美的非等宽字体才会破灭。多显示 1.45 内容？还是只有 0.6896551724137931 的丑恶巨大空格？要用非等宽得到 emacs 看看，因为没有一个 editor 的主题是可以自己完全定制的，vscode 那样的使用 javascript 还难以控制还过于复杂资源又大的就算了。elisp/fsf/clean/simple，学 emacs 之前根本不能理解这话！有些事情得是自己做了才能得之心而应之于手！
- (e). 历史，50 年历史的编辑器，计算机史的一页，还能写 100 年左右。
- (f). 装 b，(linus/stallman) 开源巨人们(rust/python/java/ruby/curl/各种函数式… ) 之父们钦定编辑器。

3.2 我告诉你，年轻人啊，学习老掉牙的 emacs，就像是金庸小说里的年轻主人公学习失传的武林秘籍一般（只是小说剧情而矣）。

```
implementation-specific diagnostic information on the standard error output and  
implementation-specific diagnostic information on the standard error output and d
```

图 3.1: mono 显示 80 个字符

```
implementation-specific diagnostic information on the standard error output and  
implementation-specific diagnostic information on the standard error output and diagnostic information on the stand
```

图 3.2: 等高下非等宽显示 116 个字符

```
(\_\_c\_backslash\_column\_\_)\n(setq c-backslash-max-column 99)\n(setq c-basic-offset 4)\n(setq c-indent-level 4)\n;; (setq c-default-style "linux")\n(advice-add 'c-update-modeline :override #'ignore)\n(defun c-compile-current-file ()\n  (interactive)\n  (unless (file-exists-p "Makefile")\n    (let ((file (file-name-nondirectory buffer-file-name)))\n      (compile (format "%s -o %s.out %s %s %s"\n                      (or (getenv "CC") "gcc")\n                      (file-name-sans-extension file)\n                      (or (getenv "CPPFLAGS") "")\n                      (or (getenv "CFLAGS") "-Wall -g")\n                      file))))\n  (dolist (hook '(c-mode-hook c++-mode-hook asm-mode-hook))\n    (add-hook hook (lambda ()\n      (define-key c-mode-base-map (kbd "C-c C-c") 'c-compile-current-file)\n      (define-key c-mode-base-map (kbd "C-c C-M-c") (lambda () (interactive)\n          (setq-local compilation-read-command nil)\n          (call-interactively 'compile)))\n      (define-key c-mode-base-map (kbd "C-M-q") nil)\n      (define-key c-mode-base-map (kbd "(") nil)\n      (define-key c-mode-base-map (kbd "(") nil))))\n    (setq confirm-kill-processes nil)
```

图 3.3: 明显可以看出等宽不符合人的阅读习惯，只有在多行修改 (mc) 或跳转 (avy) 中需要字符不变才“稍显有益”。

```
("\\.\.ii\\' . c++-mode)\n:config\n(setq c-backslash-column 99)\n(setq c-backslash-max-column 99)\n(setq c-basic-offset 4)\n(setq c-indent-level 4)\n;; (seta c-default-style "linux")\n(advice-add 'c-update-modeline :override #'ignore)\n(defun c-compile-current-file ()\n  (interactive)\n  (unless (file-exists-p "Makefile")\n    (let ((file (file-name-nondirectory buffer-file-name)))\n      (compile (format "%s -o %s.out %s %s %s"\n                      (or (getenv "CC") "gcc")\n                      (file-name-sans-extension file)\n                      (or (getenv "CPPFLAGS") "")\n                      (or (getenv "CFLAGS") "-Wall -g")\n                      file))))\n  (dolist (hook '(c-mode-hook c++-mode-hook asm-mode-hook))\n    (add-hook hook (lambda ()\n      (define-key c-mode-base-map (kbd "C-c C-c") 'c-compile-current-file)\n      (define-key c-mode-base-map (kbd "C-c C-M-c") (lambda () (interactive)\n          (setq-local compilation-read-command nil)\n          (call-interactively 'compile)))\n      (define-key c-mode-base-map (kbd "C-M-q") nil)\n      (define-key c-mode-base-map (kbd "(") nil)\n      (define-key c-mode-base-map (kbd "(") nil))))\n    (setq confirm-kill-processes nil)
```

图 3.4: 非等宽下，英文字符清晰区别明显，字符间隔小，更优美自然连续

### 3. 上网必备 emacs

```
(keymap-global-set "s-m" #'switch-to-gptel)
(keymap-global-set "<Tools>" #'tavily-search)

(defun switch-to-gptel()
  (interactive)
  (if (equal (current-buffer) (gptel "*deepseek*"))
      (previous-buffer)
    (switch-to-buffer "*deepseek*")))

(defun tavily-search-async (callback query &optional search-depth max-results exclude_domains
                                country include_domains)
  "Perform a search using the Tavily API and return results as JSON string.
API-KEY is your Tavily API key.
QUERY is the search query string.
Optional SEARCH-DEPTH is either \"basic\" (default) or \"advanced\".
Optional MAX-RESULTS is the maximum number of results (default 5)."
  (require 'plz)
  (let* ((plz-curl-default-args (cons "-k" plz-curl-default-args))
         (url "https://api.tavily.com/search")
         (search-depth (or search-depth "advanced"))
         (max-results (or max-results 1)))
    (include_answer t)
    (country (or country "united_states"))
    (include_domains (or include_domains '("nixos.org" "freertos.org" "zephyrproject.org" "contiki-ng.org" "riot-os.org" "nuttx.apache.org" "mynewt.apache.org" "ziglang.org" "python.org" "lua.org" "elixir-lang.org" "erlang.org" "haskell.org" "cmake.org" "gnu.org" "llvm.org" "gcc.gnu.org" "qt.io" "gtk.org" "sdl.org" "libsdl.org" "qemu-project.org" "cppreference.com" "opensource.org" "ietf.org" "w3.org" "ansi.org" "iso.org" "ieee.org" "man7.org" "discourse.nixos.org" "ziggit.dev" "emacs-china.org" "lwn.net" "kernel.org" "sourceware.org" "debian.org" "archlinux.org" "github.com" "osdev.org" "opencores.org" "riscv.org" "musl-libc.org" "newlib.sourceware.org" "uclibc-ng.org" "hackaday.com" "raspberrypi.org" "arduino.cc" "espressif.com" "gentoo.org")))
    (request-data
     `(("api_key" . ,tavily-api-key)
       ("query" . ,query)
       ("search_depth" . ,search-depth)
       ("country" . ,country)
       ("include_domains" . ,include_domains)
       ("include_answer" . ,include_answer)
       ("exclude_domains" . ,exclude_domains)
       ("max_results" . ,max-results))))
  (plz 'post url
       :headers '("Content-Type" . "application/json")
       :body (json-encode request-data)
       :as 'string
       :then (lambda (result) (funcall callback result)))))

(defun tavily-search (query)
  (interactive "sQuery:")
  (tavily-search-async
```

```
(lambda (result)
  (let ((buf (get-buffer-create "*tavily-search-result*")))
    (switch-to-buffer buf)
    (read-only-mode 0)
    (erase-buffer)
    (org-mode)
    (insert result)
    (insert (tavily-result-to-org result))
    (goto-char (point-min))
    (read-only-mode 1)
    (setq-local truncate-lines nil)
  )))
query))

(defun tavily-result-to-org (json-result)
  (let* ((data (json-read-from-string json-result))
         (results (alist-get 'results data)))
    (mapconcat (lambda (item)
                 (format "* [%s] [%s]\n%s"
                        (alist-get 'url item)
                        (alist-get 'title item)
                        (alist-get 'content item)))
               results
               "\n\n")))
(setq tavily-api-key
      (with-temp-buffer
        (insert-file-contents "/run/secrets/tavily_apikey")
        (buffer-string)))
(use-package gptel
  :init
  (require 'gptel-org)
  :config
  (with-eval-after-load 'gptel
    (gptel-make-tool
     :category "web"
     :name "search"
     :async t
     :function (lambda (cb keyword)
                (tavily-search-async cb keyword "basic" 5 nil nil nil))
     :description "Search the Internet; If you used any search results, be sure to include the references in your response."
     :args (list '(:name "keyword"
                   :type string
                   :description "The keyword to search")))
    (gptel-make-tool
     :name "create_python_repl"
     :function (lambda ()
                (run-python nil t)
                (pop-to-buffer (python-shell-get-buffer))))
     :description "Create a new python repl for this session"))
  ))
```

```
:args nil
:category "emacs")
(gptel-make-tool
:name "send_python_to_repl"
:function (lambda (code)
(python-shell-send-string code))
:args (list '(:name "code"
:type string
:description "python\u201ccode\u201dto\u201dexecute"))
:description "Send\u201csome\u201cpython\u201ccode\u201dto\u201cthe\u201cpython\u201crepl\u201dfor\u201cthis\u201csession\u201dand\u201cexecute\u201d"
:category "emacs")
(gptel-make-tool
:function (lambda (url)
(with-current-buffer (url-retrieve-synchronously url)
(goto-char (point-min)) (forward-paragraph)
(let ((dom (libxml-parse-html-region (point) (point-max))))
(run-at-time 0 nil #'kill-buffer (current-buffer))
(with-temp-buffer
(shr-insert-document dom)
(buffer-substring-no-properties (point-min) (point-max))))))
:name "read_url"
:description "Fetch\u201cand\u201cread\u201cthe\u201ccontents\u201cofa\u201cURL"
:args (list '(:name "url"
:type "string"
:description "The\u201cURL\u201dto\u201cread"))
:category "web")
(gptel-make-tool
:function (lambda (buffer text)
(with-current-buffer (get-buffer-create buffer)
(save-excursion
(goto-char (point-max))
(insert text)))
(format "Appended\u201ctext\u201cto\u201cbuffer\u201c%s" buffer))
:name "append_to_buffer"
:description "Append\u201ctext\u201cto\u201cthe\u201can\u201cEmacs\u201cbuffer.\u201cIf\u201cthe\u201cbuffer\u201ddoes\u201dnott\u201dexist,\u201cxit\u201dwill\u201dbe\u201d
created."
:args (list '(:name "buffer"
:type "string"
:description "The\u201cname\u201cof\u201cthe\u201cbuffer\u201dto\u201append\u201ctext\u201cto .")
'(:name "text"
:type "string"
:description "The\u201ctext\u201dto\u201append\u201cto\u201cthe\u201cbuffer ."))
:category "emacs")
(gptel-make-tool
:function (lambda (text)
(message "%s" text)
(format "Message\u201csent:\u201c%s" text)))
:name "echo_message"
:description "Send\u201ca\u201cmessage\u201dto\u201cthe\u201c*Messages*\u201cbuffer"
```

```
:args (list '(:name "text"
               :type "string"
               :description "The text to send to the messages buffer"))
:category "emacs")
(gptel-make-tool
:function (lambda (buffer)
            (unless (buffer-live-p (get-buffer buffer))
              (error "Error: buffer %s is not live." buffer))
            (with-current-buffer buffer
              (buffer-substring-no-properties (point-min) (point-max))))
:name "read_buffer"
:description "Return the contents of an Emacs buffer"
:args (list '(:name "buffer"
               :type "string"
               :description "The name of the buffer whose contents are to be retrieved"))
:category "emacs")
(gptel-make-tool
:function (lambda (directory)
            (mapconcat #'identity
                       (directory-files directory)
                       "\n"))
:name "list_directory"
:description "List the contents of a given directory"
:args (list '(:name "directory"
               :type "string"
               :description "The path to the directory to list"))
:category "filesystem")
(gptel-make-tool
:function (lambda (parent name)
            (condition-case nil
                (progn
                  (make-directory (expand-file-name name parent) t)
                  (format "Directory %s created/verified in %s" name parent))
                (error (format "Error creating directory %s in %s" name parent))))
:name "make_directory"
:description "Create a new directory with the given name in the specified parent directory"
:args (list '(:name "parent"
               :type "string"
               :description "The parent directory where the new directory should be created, e.g. /tmp")
            '(:name "name"
               :type "string"
               :description "The name of the new directory to create, e.g. testdir"))
:category "filesystem")
(gptel-make-tool
:function (lambda (path filename content)
            (let ((full-path (expand-file-name filename path)))
              (with-temp-buffer
                (insert content)))
```

```
(write-file full-path))
  (format "Created file %s in %s" filename path)))
:name "create_file"
:description "Create a new file with the specified content"
:args (list '(:name "path"
  :type "string"
  :description "The directory where to create the file")
  '(:name "filename"
  :type "string"
  :description "The name of the file to create")
  '(:name "content"
  :type "string"
  :description "The content to write to the file"))
:category "filesystem")
(gptel-make-tool
:function (lambda (filepath)
  (with-temp-buffer
    (insert-file-contents (expand-file-name filepath))
    (buffer-string)))
:name "read_file"
:description "Read and display the contents of a file"
:args (list '(:name "filepath"
  :type "string"
  :description "Path to the file to read. Supports relative paths and ~."))
:category "filesystem")
(defun ant/gptel-save-buffer ()
  "Save the current GPTEL buffer with the default directory
set to ~/note."
(interactive)
(let ((default-directory "~/Leere/qingsongliao.github.io/")
      (call-interactively #'save-buffer)))
(defun ant/gptel-load-session ()
  "Load a gptel session from ~/notes directory."
(interactive)
(let ((default-directory "~/.leetcode/code/"))
  (let* ((files (directory-files default-directory t ".+\\" .org$"))
        (file (completing-read "Select session file: " files nil t)))
    (when file
      (find-file file)
      (gptel-mode))))
(setq gptel-default-mode 'org-mode)
(setq deepseek-api-key
  (with-temp-buffer
    (insert-file-contents "/run/secrets/deepseek_apikey")
    (buffer-string)))
(setq gptel-model 'deepseek-chat
  gptel-backend (gptel-make-deepseek "deepseek"
    :stream t
    :key deepseek-api-key))
```

```
(require 'url-util)
(setq gptel-directives
  '((default . "You\u00e2\u008auare\u00e2\u008aularge\u00e2\u008alanguage\u00e2\u008amodel\u00e2\u008aliving\u00e2\u008uin\u00e2\u008 Emacs\u00e2\u008uand\u00e2\u008auhelpful\u00e2\u008uassistant.")
  (programming . "You\u00e2\u008auare\u00e2\u008aularge\u00e2\u008alanguage\u00e2\u008amodel\u00e2\u008uand\u00e2\u008ua\u00e2\u008ucareful\u00e2\u008aprogrammer.\u00e2\u008uProvide\u00e2\u008ucode\u00e2\u008u
    and\u00e2\u008uonly\u00e2\u008uas\u00e2\u008uoutput\u00e2\u008uwithout\u00e2\u008uany\u00e2\u008uadditional\u00e2\u008utext,\u00e2\u008uprompt\u00e2\u008uor\u00e2\u008unote.\u00e2\u008u")
  (writing . "You\u00e2\u008auare\u00e2\u008aularge\u00e2\u008alanguage\u00e2\u008amodel\u00e2\u008uand\u00e2\u008auwriting\u00e2\u008uassistant.\u00e2\u008uRespond\u00e2\u008uconcisely.\u00e2\u008u")
  (chat . "You\u00e2\u008auare\u00e2\u008aularge\u00e2\u008alanguage\u00e2\u008amodel\u00e2\u008uand\u00e2\u008ua\u00e2\u008uconversation\u00e2\u008upartner.\u00e2\u008uRespond\u00e2\u008uconcisely.\u00e2\u008u")
  (bug . "You\u00e2\u008auare\u00e2\u008aularge\u00e2\u008alanguage\u00e2\u008amodel\u00e2\u008uand\u00e2\u008ua\u00e2\u008ucareful\u00e2\u008aprogrammer.\u00e2\u008uThe\u00e2\u008usupplied\u00e2\u008ucode\u00e2\u008udoesn
    't\u00e2\u008uwork,\u00e2\u008uor\u00e2\u008ucontains\u00e2\u008ubugs.\u00e2\u008uDescribe\u00e2\u008ueach\u00e2\u008uproblem\u00e2\u008uusing\u00e2\u008uonly\u00e2\u008uone\u00e2\u008usentence.\u00e2\u008uProvide\u00e2\u008u
    fixes\u00e2\u008uwithout\u00e2\u008uchanging\u00e2\u008uthe\u00e2\u008uold\u00e2\u008ubehavior.\u00e2\u008u")))
(setq gptel-stream nil))
(use-package consult-gh
  :after consult
  :custom
  (consult-gh-confirm-before-clone nil)
  (consult-gh-default-clone-directory "~/codebase")
  (consult-gh-ask-for-path-before-save nil)
  (consult-gh-default-save-directory "~/codebase")
  (consult-gh-show-preview t)
  (consult-gh-preview-key "C-o")
  (consult-gh-repo-action #'consult-gh--repo-browse-files-action)
  (consult-gh-large-file-warning-threshold 2500000)
  (consult-gh-confirm-name-before-fork nil)
  (consult-gh-notifications-show-unread-only nil)
  (consult-gh-default-interactive-command)
  (consult-gh-prioritize-local-folder nil)
  (consult-gh-issues-state-to-show "all") ; show readmes in their original format
  (consult-gh-group-dashboard-by :reason)
  (consult-gh-repo-preview-major-mode nil) ; show readmes in their original format
  (consult-gh-preview-major-mode 'org-mode) ; use 'org-mode for editing comments, commit messages
  , ...
  :config(consult-gh-enable-default-keybindings)
  (use-package consult-gh-forge
    :after consult-gh
    :config
    (consult-gh-forge-mode +1))
  (use-package consult-gh-embark
    :after consult-gh
    :config
    (consult-gh-embark-mode +1))
  (setq consult-gh-forge-timeout-seconds 20)))
```

**Listing 3.2:** the internet capability of me. search/gpt/github and I found they are just trash. use selfhost doc/info/tldr/man/devdocs is way better.

# 第四章 操作系统哲学 Operation System

## 4.1 我的操作系统特点

1. network, 471mb hosts to block internet out of my pc with dnsmasq and dae proxy tool based on epbf... you know I am ill about internet? because I am in china, and the nsfw/trump/4chan dudes, I love them so much, so I add them into my hosts.
2. emacs editor, + nixos, they both ruin my early adult life by distract me away from reality girls.
3. wm: **sway**, minimal config, pure white look, although **miku-cursor**, now, I use plain mouse instead.
4. shell: **fish+foot**, with all I need in, direnv/many alias/zoxide/git/and thefxxk or thefxxk updated to "chatgpt"'s codex nowadays?
5. browser: **firefox** [default uninstall], I use eww because I hate browser.
6. fully reproducible, even with dict/font/anime wallpaper!
7. zig/linux/ccpp doc in nginx, all kind texinfo can read in emacs.
8. with sops, to store my 0.1 dollar's deepseek account apikey, and chat with my ai girlfriend with my payment password.
9. I am tired of nixos, but I can't leave it, because after using nixos everything is hard to do in other distros, like show off to other distro users "BTW, I use NixOS".

## 4.2 用 linux 只为戒色

翻开中国帝王史，有多少帝王是沉迷女色后国家衰亡，多少帝王是皇太后控制后抑郁早亡。野史里选王要是看他不好色否？当皇帝得看他有无抢夺妇女。翻开网络史发展，有多少人民群众是沉迷情色后国家衰亡，多少人是被色情控制后抑郁早亡，看人得看他好色否？好不好色，得看他懂得怎么预防。oppo 的天气预报能跳到擦边短剧，bing 的结果第一个 BMI 测试广告中跳到黄色导航，百度误输成 baidu.co 中进入黄色应用下载，网易看见过全露的黄色直播，B 站中处处是擦边广告和直播，微博中卖到各种片，知乎里表露情色隐私更引人浏览，我的世界游戏里用色情披风，淘宝里随便看情趣用品，openai 支持成人内容，twitter 全是福利姬，就连 QQ 空间连上擦边广告商，onlyfans 比所有 ai 公司利润更高，pornhub 全球流量前 10，整个 00 后的网络被“这提醒我了”占据，整代女人被商业文化营造黑丝和大胸的“性感”，整代男人被情色文化成为无脑动物与被情欲“冲动”控制，男人就该好色？女人就该这样美吗？天下皆知美之为美，斯恶也。互联网 33% 的流量，70% 的男性，30% 的女性，90% 男性沉迷其中…数字不会说谎，一切已经到来，生存与繁衍不再重要，存在不过是不断刺激感官。不只是色情…网络带来的问题似乎总是和成瘾有关，赌博/购物/社交/游戏…就连互联网最自豪的正面形象所谓的“利于学习”也成为了信息过载的社会巨大问题，自由的代价是什么？自由即是强迫。互联网即是色情。

论我怎么戒色？答：学习技术、理解技术、控制技术，学习自由软件，远离商业软件，拒绝即是自由。

- 成瘾源于：随机 | 不必 | 即时 | 匿名 | 易得 | 免费 | 广告 | 失范，而互联网是完美满足这几点的最佳载体，网络从败坏道德、扰乱法律、刺激心理、久坐身体上来说绝不比烟酒赌淫的伤害少。
- 通过 **路由器 +NixOS**，一个台式机，不使用手机与笔记本，使用玻璃胶封口，交出手机给他人并禁止其给予手机

**时间** 限时，一天三小时，2-5PM，时间甚至过于长了。

**空间** 域名，471MB 的 hosts 禁止所有与编程无关或有失范或广告内容的域名。

**速度** 网速，1Mbps，足够文本传输。

**强制** 禁止，停止 cache.nixos.org 流量以止系统更新。

**实名** 真名，不去键盘也不去木马网站，不购物也不怕大数据。

- 软件只使用 **Emacs**，不使用浏览器与 IDE，网络浏览有关都减少或禁止使用，它们象征着成瘾

**google** tavily, 日常绝不使用, 搜索瘾  
**chatgpt** gptel, 日常绝不使用, 询问瘾  
**firefox** eww, 日常绝不使用, 网络瘾  
**github** consult-gh, 日常绝不使用, 偷窥瘾  
**leetcode** leetgo  
**vpn** dae/dnsmasq  
**obsidian** org  
**wiki** nginx/info/tldr/devdocs/man

- 硬件只使用 **Eink** 墨水屏, 提升对比度至最高到只能浏览文本, 不使用音响与普通键盘

**无音乐** 无音响, 五音令人耳聋  
**无游戏** 无显卡, 驰骋修猎使人心发狂  
**无动漫** 无色彩, 五色令人目盲  
**无社媒** 无软件, 驰骋攸猎使人心发狂  
**无购物** 无网络, 难得之贷使人行妨  
**无视频** 竖立屏, 驰骋攸猎使人心发狂

- 要么学习技术, 要么远离技术

1. 学好一门语言
2. 学好数据结构与算法
3. 学好一个框架
4. 学好 magit/github
5. 做好一个项目
6. 去写一份简历
7. 去找一个工作

## 4.3 学技术只为戒技术

没有技术, 不会英文, 思维局限在资源封闭的中文世界。没有技术, 不会 ADB, 改不了手机的成瘾性。你只是应用的奴隶。没有技术, 不会 NixOS, 改不了自己电脑操作系统的所有权。你只是 M\$ 的奴隶。没有技术, 不会 Emacs, 改不了使用 Word/Edge/VSCode 的痛苦性。你只是 M\$ 的奴隶。没有技术, 不会尝试技术, 你怎么会去成长与改变。没有技术, 不会编程技术, 这个时代只有靠别人的脸色吃饭。没有技术, 不会理解技术, 你无法提前预知使用技术带来的后果。没有技术, 不会反对技术, 你将成为社交媒体网络/游戏开发者/色情女主播/新闻贩卖者/购物直播者/焦虑卖课人的奴隶。你没有技术, 社会不会改变, 技术是社会根基, 但有了技术心理, 怎么会惧怕技术?

## 第五章 语言哲学 Language

### 5.1 英文比不上那方块字

### 5.2 什么？编程语言也是语言？

- 看一下 zig 哲学：

- Communicate intent precisely.
- Edge cases matter.
- Favor reading code over writing code.
- Only one obvious way to do things.
- Runtime crashes are better than bugs.
- Compile errors are better than runtime crashes.
- Incremental improvements.
- Avoid local maximums.
- Reduce the amount one must remember.
- Focus on code rather than style.
- Resource allocation may fail; resource deallocation must succeed.
- Memory is a resource.
- Together we serve the users.

# 第六章 键盘流哲学 Keyboard

## 6.1 设计哲学

- 与原生相合 (win)
- 不冲突 (special prefix)
- 合乎习惯 (stay base)
- home row 效率 (home row effiction)
- general in os(browser, editor)

```
| BASE{
  bindings = <
    &none &kp Q  &kp W  &kp F  &kp P  &kp B  &kp J  &kp L  &kp U  &kp Y  &kp SQT &none
    &none &hm LGUI A  &hm LALT R  &hm LCTRLS  &hm LSHIFT T  &kp G  &kp M  &hm RSHIFT N  &hm RCTRL E  &hm RALT I  &hm RGUI O &none
    &none &kp Z  &kp X  &kp C  &kp D  &kp V  &kp K  &kp H  &kp COMMA  &kp DOT  &kp FSLH &none
    &lt MEDIA ESC &lt NAV SPACE &lt MOUSETAB &lt SYM RET &lt NUM BSFC &lt FUN DEL
  >;
};

// copy/down/up/fullscreen/toggle/ cduftabr
Nav{
  bindings = <
    &none &navq  &navw  &navf  &navp  &navb  &navj  &navl  &navu  &navy  &nav_sq &none
    &none &kp LGUI  &kp LALT  &kp LCTRL  &kp LSHIFT  &navg  &navm  &kp LEFT  &kp DOWN  &kp UP  &kp RIGHT &none
    &none &navz  &navx  &navc  &navd  &navv  &kp INSERT  &kp HOME  &kp PG_DN  &kp PG_UP  &kp END &none
    &trans  &trans  &trans  &nav_enter  &nav_delete  &trans
  >;
};

Numbers {
  bindings = <
    &none &kp LBKT  &kp N7  &kp N8  &kp N9  &kp RBKT  &numj  &numl  &numu  &numy  &num_sq &none
    &none &kp SEMI  &kp N4  &kp N5  &kp N6  &kp EQUAL  &numm  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI &none
    &none &kp GRAVE  &kp N1  &kp N2  &kp N3  &kp BSLH  &numk  &numh  &num_comma  &num_dot  &num_fslh  &none
    &kp DOT  &kp NO  &kp MINUS  &trans  &trans  &trans
  >;
};

Mouse{
  bindings = <
    &none &mosq  &mosw  &mosf  &mosp  &mosb  &mosj  &mosl  &mosu  &mosy  &mos_sq &none
    &none &mosa  &mosr  &moss  &most  &mosg  &mosm  &mmv MOVE_LEFT  &mmv MOVE_DOWN  &mmv MOVE_UP  &mmv MOVE_RIGHT &none
    &none &mosz  &mosx  &mosc  &mosd  &mosv  &mosw  &mos &msc SCRL_LEFT  &msc SCRL_DOWN  &msc SCRL_UP  &msc SCRL_RIGHT &none
    &trans  &mos_space  &trans  &mkp RCLK &mkp LCLK &mkp MCLK
  >;
};

Symbols {
  bindings = <
    &none &kp LBRC  &kp AMPS  &kp ASTRK  &kp LPAR  &kp RBRC  &just_code  &explain_cn  &think_deep  &search_web  &error_fix  &none
    &none &kp COLON  &kp DLLR  &kp PRCNT  &kp CARET  &kp PLUS  &gpTEL  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI &none
    &none &kp TILDE  &kp EXCL  &kp AT  &kp HASH  &kp PIPE  &pass  &id  &qq  &email  &phone &none
    &kp LPAR  &kp RPAR  &kp UNDER  &trans  &trans
  >;
};

Function {
  bindings = <
    &none &kp F12  &kp F7  &kp F8  &kp F9  &kp PRINTSCREEN &none  &kp F18  &kp F19  &kp F20  &kp F21  &none
    &none &kp F11  &kp F4  &kp F5  &kp F6  &kp SCROLLOCK  &kp F17  &kp RSHIFT &kp RCTRL &kp RALT  &kp RGUI  &none
    &none &kp F10  &kp F1  &kp F2  &kp F3  &kp PAUSE_BREAK &none  &kp F13  &kp F14  &kp F15  &kp F16  &none
    &trans  &trans  &trans  &trans  &trans  &trans
  >;
};

media{
  bindings = <
    &none &none  &none  &none  &none  &kp C_REWIND  &none  &none  &none  &kp C_FAST_FORWARD &none
    &none &kp LGUI  &kp LALT  &kp LCTRL  &kp LSHIFT &none  &kp C_PREVIOUS  &kp C_VOL_DN  &kp C_VOLUME_UP  &kp C_NEXT &none
    &none &none  &none  &none  &none  &out OUT_TOG  &bt BT_CLR  &bt BT_SEL0  &none  &none  &none
    &trans  &trans  &trans  &kp C_PLAY_PAUSE  &kp C_STOP  &kp C_MUTE
  >;
};
```

图 6.1: keymap of my keyboard

## 6.2 分层

1. base 层, colemak-dh, 最快速与舒适的英文布局
2. nav 层, 上下左右, 快速移动, C-c C-~ 为前缀, emacs 快速选择
3. num 层, 数字键, 控制 sway 的窗口, 与系统剪切板和应用菜单
4. mos 层, 鼠标上下左右, C-c C-; 为前缀, 一些 emacs 功能。
5. sym 层, 全部符号, gpt prompt 与个人信息快速输入层
6. fun 层, function 1/2.12 层
7. med 层, 提供蓝牙/媒体切换

## 6.3 本地刷录

```
#!/usr/bin/env bash
# ===== CONFIGURATION =====
GITHUB_OWNER="NestorLiao"
GITHUB_REPO="zmk-config"
GITHUB_TOKEN=$(cat ~/.github_token)
ZIP_DEST="zmk_build.zip"
UNZIP_DIR="zmk_build"

LEFT_FW="corne_left-nice_nano_v2-zmk.uf2"
RIGHT_FW="corne_right-nice_nano_v2-zmk.uf2"
SETTINGS_RESET="settings_reset-nice_nano_v2-zmk.uf2"

# ===== FUNCTIONS =====

flash_settings_reset() {
    echo "Flashing settings reset firmware..."
    for i in 1 2; do
        echo "Reset device $i/2 into bootloader mode..."
        # Wait for mount
        while [ ! -d "/media/NICENANO" ]; do
            sleep 0.5
        done
        echo "NICENANO detected. Flashing settings reset..."
        if [ -f "$SETTINGS_RESET" ]; then
            cp "$SETTINGS_RESET" /media/NICENANO/
            echo "Settings reset flashed $i/2."
        else
            echo " $SETTINGS_RESET not found in current directory."
            exit 1
        fi
    echo "Waiting for NICENANO to unmount..."
```

```

while [ -d "/media/NICENANO" ]; do
    sleep 0.5
done
echo " Settings reset completed twice."
}

download_latest_artifact() {
    echo "Fetching latest artifact..."

run_id=$(curl -s -H "Authorization: token $GITHUB_TOKEN" \
"https://api.github.com/repos/$GITHUB_OWNER/$GITHUB_REPO/actions/runs?per_page=1&status=success" \
" \
| jq -r '.workflow_runs[0].id')

if [ "$run_id" == "null" ] || [ -z "$run_id" ]; then
    echo " No successful workflow run found."
    exit 1
fi

echo "Found workflow run ID: $run_id"

artifact_url=$(curl -s -H "Authorization: token $GITHUB_TOKEN" \
"https://api.github.com/repos/$GITHUB_OWNER/$GITHUB_REPO/actions/runs/$run_id/artifacts" \
| jq -r '.artifacts[0].archive_download_url')

if [ "$artifact_url" == "null" ] || [ -z "$artifact_url" ]; then
    echo " No artifacts found for run $run_id."
    exit 1
fi

echo "Downloading artifact zip..."
curl -L -H "Authorization: token $GITHUB_TOKEN" \
"$artifact_url" -o "$ZIP_DEST"

echo "Unzipping..."
rm -rf "$UNZIP_DIR"
unzip -q "$ZIP_DEST" -d "$UNZIP_DIR"
}

flash_from_local() {
    if [ ! -f "$ZIP_DEST" ]; then
        echo " $ZIP_DEST not found in current directory."
        exit 1
    fi

    echo "Using local $ZIP_DEST..."
    rm -rf "$UNZIP_DIR"
    unzip -q "$ZIP_DEST" -d "$UNZIP_DIR"
}

```

```

}

flash_firmware() {
    local fw_path="$1"
    echo "Please reset the device into bootloader mode..."

    while [ ! -d "/media/NICENANO" ]; do
        sleep 0.5
    done

    echo "NICENANO detected. Flashing $fw_path..."
    cp "$fw_path" /media/NICENANO/
    echo "Done."

    echo "Waiting for NICENANO to unmount..."
    while [ -d "/media/NICENANO" ]; do
        sleep 0.5
    done
}

flash_left_right() {
    flash_firmware "$UNZIP_DIR/$LEFT_FW"
    flash_firmware "$UNZIP_DIR/$RIGHT_FW"
}

show_usage() {
    echo "Usage: $0 <mode>"
    echo "Modes:"
    echo "  1 - Flash settings, reset twice"
    echo "  2 - Download from GitHub and flash left/right"
    echo "  3 - Use local zmk_build.zip to flash left/right"
}

# ===== MAIN =====

if [ $# -ne 1 ]; then
    show_usage
    exit 1
fi

case $1 in
    1)
        echo "== Mode 1: Flash Settings, Reset =="
        flash_settings_reset
        ;;
    2)
        echo "== Mode 2: Download and Flash =="
        download_latest_artifact
        flash_left_right
    esac
}

```

```

;;
3)
echo "==_Mode_3:_Local_Flash_=="
flash_from_local
flash_left_right
;;
*)
echo " _Invalid_mode:_$1"
show_usage
exit 1
;;
esac

echo " _All_done."

```

**Listing 6.1:** 1 刷重置, 2 刷 github action, 3 刷本地 zip 文件

## 6.4 不只键盘

**Absolute Enable Right Click & Copy** Force Enable Right Click & Copy.

**Authenticator** Authenticator generates two-factor authentication codes in your browser.

**Bar Breaker** Hides fixed headers and footers on pages you visit.

**ChatGPT Ctrl+Enter Sender** Use 'Ctrl+Enter' for sending messages in AI chat services like ChatGPT. Prevents accidental sends and allows line breaks with Enter.

**ChatGPT Disable Auto Scroll - FREE** Stop annoying scrolling in ChatGPT. Prevent jerky animation of appearing text.

**ClearURLs** Remove tracking elements from URLs.

**Decentraleyes** Protects you against tracking through "free", centralized, content delivery.

**Default links not to be underlined** Restores non-underlined hyperlinks by default (setting text-decoration: none).

**Disconnect** Make the web faster, more private, and more secure.

**Enforce Safe Search (=Adult Filter)** Toggles the built-in filter on many search engines (see

**Focus On Left Tab After Closing** When a current tab is closed, the left tab (or the right tab on RTL

**Font Contrast** Improves webpage readability.

**FuckCSDN** Clean CSDN's limitation scripts

**Google Search Ad Remover And Customizer** This extension gives you the ability to customize how your Google search results

**Hide shorts for Youtube™** Hides shorts from YouTube™ from home page, subscriptions and search results.

**hide-scrollbars** Hides page scrollbars!

**I still don't care about cookies** Community version of the popular extension "I don't care about cookies"

**Invert Colors** A simple add-on that inverts the page colors.

**No Emoji** Browser extension to remove emoji.

**Shut Up: Comment Blocker** Blocks comment sections on many popular websites.

**Stack Copy Button** A copy button for Stack Overflow code boxes

**Tab Sidebar** Adds a sidebar with foldable tabs.

**uBlock Origin** Finally, an efficient blocker. Easy on CPU and memory.

**Unhook - Remove YouTube Recommended & Shorts** Hide YouTube related videos, shorts, comments, suggestions wall, homepage

**Vimium C - All by Keyboard** A keyboard shortcut tool for keyboard-based page navigation and browser tab

# 第七章 屏幕哲学 Screen

Real Mono Theme, two colors are enough for emacs.

## 7.1 Feels:

A collection of real monochrome emacs themes in a couple of variants.

These screenshots show various versions of the Real Mono theme for Emacs. They demonstrate different color palettes and font choices.

- /home/leeso/Leere/real-mono-theme**: Head: master Initial commit. Shows a dark theme with white text and icons.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a similar dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a light theme with dark text and icons.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a light theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a light theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.
- /home/leeso/Leere/real-mono-theme**: Head: v2.0.0 Initial commit. Shows a dark theme.

```

/home/lameci/serveur/real-mono-them/HEAD - master (initial commit)
  image
    real-mono-dark.png
    real-mono-dark@.png
    real-mono-gif.png
    real-mono-old.png
  README.md
  real-mono-dark-theme.dl
  real-mono-dark-theme@.dl
  real-mono-old-theme.dl
  real-mono-old-theme@.dl
  real-mono-themes.dl
  real-mono-themes@.dl

  ...
  @file
  @brief Implementation to
  (Count number of bits to be flipped
  (@https://www.geeksforgeeks.org/
  "An integer.
  "It has 10100 bits, we set 10100" v

  @details
  "We are given two numbers x and y
  and we need to flip y to convert it to x
  "
  "Explanation:
  "A < 01010 < 10100
  "As we can see, the bits of A that are
  "1 are those bits, we set 10100 v

  # Patterns:
  - "Not Greyscale", no blur anymore
  1. "Not Greyscale", no blur anymore
  2. "Easy" to customize, one off the
  3. "Interaction free", no scroll by

```

pictures's font list: bookerly, ubuntumono, firacode, terminess, bookerly bold italic.

## 7.2 Features:

1. **Not Greyscale**, no blur anymore, it's much better to use eink screen for pure black and white!
2. **Easy** to customize, can set the only two colors by config the default face's foreground/background color.
3. **Distraction-free**, no extra info-overwhelming causing by font-lock, only few font diversity in magit/dired etc for better function recognize.
4. **Full**, configed 370+ faces, I didn't see any monochrome theme can reach that much, as my daily driver, it's good enough.

## 7.3 Tips for mono:

- (global-hide-mode-line-mode 1), build your own brain memory
- (no-emoji 1), alter emacs to be "text editor" instead of discord
- (show-paren-mode -1), build your own eye insight
- (window-divider-mode -1), too, build your memory
- (display-line-numbers-mode -1), too, build your own eye insight

```
(setq hl-todo-keyword-faces
  '(("HOLD" . "#000000")
    ("TODO" . "#000000")
    ("NEXT" . "#000000")
    ("THEM" . "#000000")
    ("PROG" . "#000000")
    ("OKAY" . "#000000")
    ("DONT" . "#000000")
    ("FAIL" . "#000000")
    ("DONE" . "#000000")
    ("NOTE" . "#000000")
    ("MAYBE" . "#000000")
    ("KLUDGE" . "#000000")
    ("HACK" . "#000000")
    ("TEMP" . "#000000")
    ("FIXME" . "#000000")
    ("XXXX*" . "#000000")))

(defvar my-alternate-font "-DAMA-UbuntuMono_Nerd_Font-regular-normal-normal--*-13-*-*-m-0-iso10646-1"
)
```

```
(defvar my-default-font "bookerly")
(defvar fontfont 1)
(defun my-toggle-font ()
  "Toggle between UbuntuMono and bookerly fonts."
  (interactive)
  (if (= fontfont 1)
      (progn (set-face-attribute 'default nil :font my-default-font :height 160) (setq fontfont 0))
      (progn (set-face-attribute 'default nil :font my-alternate-font :height 210) (setq fontfont 1))))
```

- fringe specific mode auto hide

```
(defun my-set-fringe-face ()
  "auto-hide-fringe-face depending on major mode."
  (if (derived-mode-p '(occur-mode gud-mode))
      (set-face-attribute 'fringe nil
                         :background (face-attribute 'default :background)
                         :foreground (face-attribute 'default :foreground))
      (set-face-attribute 'fringe nil
                         :background (face-attribute 'default :background)
                         :foreground (face-attribute 'default :background))))
(add-hook 'after-change-major-mode-hook #'my-set-fringe-face)
```

- elisp for toggling paperlike-hd to switch between read and watch.

```
(defvar monitor-state 1
  "Current monitor state, either 0 for read or 1 for watch.")
(defun monitor ()
  "switch monitor from read mode to watch mode"
  (interactive)
  (let ((monitorprotocol "-i2c")
        (monitorpath "/dev/i2c-4")
        (monitorcli "paperlike-cli")
        (monitorarg '("-contrast" "-speed" "-mode" "-clear"))
        (mode-state '(("9" "5" "1") ("9" "5" "1"))))
    (split-window-below)
    (other-window 1)
    (switch-to-buffer "*Shell_Command_Output*")
    (split-window-below)
    (other-window 1)
    (switch-to-buffer "*Async_Shell_Command*")
    (progn
      (dotimes (number 3)
        (call-process monitorcli nil nil nil
                      monitorprotocol
                      monitorpath
                      (car (nthcdr number monitorarg))
                      (car (nthcdr number (car (nthcdr monitor-state mode-state)))))))
      (sleep-for 1))
    (setq monitor-state (if (= 0 monitor-state) 1 0)))
    (sleep-for 1.5)
    (sleep-for 0.5))
```

```
(call-process monitorcli nil nil nil nil monitorprotocol monitorpath (car (nthcdr 3 monitorarg)))  
(sleep-for 0.5)  
(donothing))
```

# 第八章 fog.h

Copied from tsoding's flag module: <https://github.com/tsoding/flag.h> Also tsoding's nob as build system: <https://github.com/tsoding/nob.h>

## 8.1 Quick Start

Check `example.c`

```
cc -o example example.c
./example -help
```

## 8.2 Just rewrite shit in C, Just rewrite shit in C — tsoding

the editor/os/term of mine all just written in c.

you can replace the 1995's hype langs with:

rust -> fin-c.h  
go -> threads.h  
python -> dll  
zig -> nob.h  
c++ -> C with class  
web -> C compile to webassembly  
do leetcode -> C with stl  
do github -> check tsoding  
do emacs -> C + elisp  
do linux -> C with fake class

# 第九章 Do AoC in Zig Learn Zig in AoC

## 内容提要

- |                    |                                       |
|--------------------|---------------------------------------|
| □ 开始时间: 2025-12-02 | □ 需要小时: 100+                          |
| □ 结束时间: 2025-12-12 | □ 本章要点: zig, practice, algo, learning |
| □ 完成难度: difficulty |                                       |

## 9.1 Why Zig

Just rewrite shit in C, Just rewrite shit in C — tsoding

the editor/os/term of mine all just written in c/c++.

but the 21st innovation is mainly for change the name. you can replace the hype langs after 1995 with c, but still you can replace 1978's hype lang "c" with 2018's hype lang "zig" which has been called "better c":

```
rust -> fin-c.h -> zig  
go -> threads.h -> zig  
python -> dll -> zig  
zig -> nob.h -> zig...  
c++ -> C with class ->zig  
web -> C compile to webassembly ->zig  
do leetcode -> C with stl -> zig  
do github -> check tsoding -> zig  
do emacs -> C + elisp -> zig  
do linux -> C with fake class -> zig
```

## 9.2 Learn Zig

- editor: just set up zig-mode, and debugger in editor is bloate!
- lang: wget [https://ziglang.org/builds/zig-x86\\_64-linux-0.16.0-dev.1484+d0ba6642b.tar.xz](https://ziglang.org/builds/zig-x86_64-linux-0.16.0-dev.1484+d0ba6642b.tar.xz)
- main tutoria: **zeglings** in codeberg.org and **langref.html** in tar

- **Code:**

- <https://github.com/evaleek/advent-of-code-template>
- <https://github.com/SpexGuy/Zig-AoC-Template>
- <https://kristoff.it/blog/advent-of-code-zig>
- <https://zigbin.io/>

- **DOC:**

- <https://ziggit.dev/docs>
- <https://ziglang.org/documentation/master/>
- <https://ziglang.org/documentation/master/std/>

## 9.3 Do AoC

- finished zeglings

- zig aoc template
- share/question in ziggit.dev

# 第十章 版本更新历史 Version

2025-12-01 更新：晴

- ① **折腾键盘**: 在我 11 月末尾，我懒癌犯了总是想些了完全无关技术的事<sup>1</sup>。今天我又搞些无关技术的事，重写了键盘的配置，重设了 emacs 的 bind。

```
([remap imenu] . consult-imenu)
([remap kill-buffer] . kill-buffer-and-window)
([remap list-buffers] . ibuffer)
([remap project-switch-to-buffer] . consult-project-buffer)
([remap switch-to-buffer] . consult-buffer)
([remap comment-line] . comment-or-uncomment-region-or-line)
([remap kill-region] . kill-line-or-region)
([remap dabbrev-expand] . cape-dabbrev)
([remap yank-pop] . consult-yank-pop)
([remap indent-rigidly] . smart-shift-right)
("C-x\u2022C-s" . (lambda () (interactive) (save-buffer)(donothing)))
("C-<return>" . (lambda () (interactive) (duplicate-dwim)(next-line)))
("C-M-<return>" . copy-from-above-command)
("C-<iso-lefttab>" . surround-insert)
("C-M-<tab>" . surround-delete)
("C-s-<tab>" . surround-change)

;; No Need "qbj'" , they are finger killers
;; MOS
;; a comment c-x c-
;; r quit c-g
;; s find-file c-x c-f
;; t switch-buffer c-x b
;; b ibuffer c-x c-b

("C-c\u2022C-;d" . dire)
("C-c\u2022C-;p" . disproject-dispatch)
("C-c\u2022C-;u" . delete-all-space)
("C-c\u2022C-;z" . isearch-forward-symbol-at-point)
("C-c\u2022C-;k" . kill-buffer-and-window)
("C-c\u2022C-;g" . magit-status)
("C-c\u2022C-;f" . rg-dwim)
("C-c\u2022C-;y" . yas-insert-snippet)
("C-c\u2022C-;v" . multi-vterm-project)
("C-c\u2022C-;m" . devdocs-browser-open)

("C-c\u2022C-;x" . consult-complex-command)
("C-c\u2022C-;c" . cleanup-buffer)
```

<sup>1</sup>要我说 **Emacs** 就和与小说里秘传的十八般武林功夫一般，有定身/移形换影/透视经脉/必杀技/念经/修炼/经文/拜师/法宝/女伴/侠义…甚至武功大概不过如此…拳脚身头可用的功夫，那里有无限可能的电子计算机的空间大啊？于是我花了很长时间思考“到底编辑的招法何在”，得到了下面的招式：动：插入/切换/输出/笔记/补全/主题/跳转/投修/替换/扩张/行修/选择/整理/重复/变化/比较/潜在，阅：简用/单页/全页/源码/图书/用文/说文，用：项目/虚拟/日志/中文/终端，搜：搜/库/问

---

```

("C-cC-;l" . git-link-dispatch)
("C-cC-;w" . (lambda () (interactive) (message "%s:%s" (what-line) (current-column)))))

("C-cC-;" . donothing) ; b for ibuffer
("C-cC-;q" . donothing)
("C-cC-;j" . donothing)

;; NAV
("C-cC-~z" . delete-window)
("C-cC-~x" . delete-other-windows)
("C-cC-~c" . split-window-below)
("C-cC-~d" . recompile)
("C-cC-~u" . consult-mark)
("C-cC-~l" . magit-log-buffer-file)
("C-cC-~y" . quick-sdcv-search-at-point)
("C-cC-~g" . er/expand-region)

("C-cC-~w" . consult-org-agenda)
("C-cC-~v" . compile)
("C-cC-~p" . (lambda () (interactive) (consult-fd "~/Zen/C/" nil)))
("C-cC-~f" . toggle-letter-case)
("C-cC-~m" . man)

("C-cC-~q" . donothing)
("C-cC-~b" . (lambda () (interactive) (consult-ripgrep "~/Zen/C/" nil)))
("C-cC-~j" . donothing)
("C-cC-~'" . donothing)

;; get sym back, all kind of toggles
("C-cC-&l" . my-toggle-font)
("C-cC-&u" . vertico-flat-mode)
("C-cC-&y" . global-hide-mode-line-mode)
("C-cC-&m" . toggle-truncate-lines)
("C-cC-&k" . describe-key)
("C-cC-&h" . git-timemachine-toggle)
("C-cC-&," . toggle-input-method)
("C-cC-&." . pyim-toggle-input-ascii)
("C-cC-&/" . pyim-punctuation-toggle)

("C-cC-&'" . donothing) ; q and b used on sym layer
("C-cC-&j" . donothing)

;; minor
("M-n". embark-next-symbol)
("M-p". embark-previous-symbol)
("M-*". (lambda () (interactive) (my/leetcode-open (string-to-number (current-word))))) )
("M-I" . consult-imenu-multi)
("M-i" . imenu)
("M-o" . other-window)

```

---

```

("M-0". (lambda () (interactive) (other-window -1)))
("C-;". iedit-mode)
("C-, ". toggle-solution-question)
("ESC<f5>". hibernatecall)
("<WakeUp>". donothing)
("M-#" . consult-register-load)
("M-$" . consult-register-store)
("C-M-#" . consult-register)
("M-gUg" . consult-goto-line)
("M-sUu" . consult-global-mark)
("M-sUO" . multi-occur)
:map isearch-mode-map
("M-r" . consult-isearch-history)
:map minibuffer-local-map
("M-r" . consult-history)
:map transient-map
("M-w" . transient-copy-menu-text)))

```

**Listing 10.1:** 我也不知道为什么，“明明想的是高效”，最后搞得这么 bloat…

---

2025-11-25 更新：晴

① **走路：**晚上和母亲一起在江边走路，她抱怨太远了，脚得走痛了，但还是我陪走完了一圈。我想我是很幸福的，有这样的母亲，虽然嘴碎但还是能陪伴我一程，在冷清的江边，只有两三个跑者与我俩，冬天好像萧条不少。我抡或背着二锤，吃了两个别人落下的柑子，敲敲死树、木桩，在草坪上练练投掷，作壶铃溜两下，中午还曾晒过太阳，晚上吃了红烧鱼，一天就这样，简单。简单也没什么不好，看着那些黄毛和抽烟的女友在摩托上追江风，也许他们的母亲就没有这样的耐心了，当然也许他们自己也无法耐受半小时的无聊，母亲不爱自己放荡的儿子，儿子不耐自己老气的母亲。吹着江风，默默无语的草和人在行走着，灯光亮着一路，一切干净又闲适，又有什么无聊呢？

② **Github：**逛 Github 其实是最浪费时间的事，因为 github 就是伪装成编程平台的社交媒体。

为 feature 点赞 为爱豆的新推点赞

为语言传教 为爱豆打 call

awesome-xx 爱豆图集合订本

上传 toy project 自己为爱豆做的周边

争论语言/编辑器/操作系统/etc 争论韩流/日流/欧美圈/内娱…

新工具新框架 网红

学习分享 卖课网红

开源学习分享 盗版卖课网红

discuss/issue 粉丝群群管理答疑

怎么不把 github 当成社交媒体？

**什么是社交媒体** 交互性/排名性/圈子化/免费/易得…

- 不交互

不点赞 作个局外人，它好它坏不必说。

不乱传项目 少乱写代码上传 github，少立 flag。

把 author 看作工具人而不是教主 当然有问题还是得说 thanks。

- 不排名

不看 star/awesome/trend/weekly github 项目大多只是 blog。

---

绝不看中文 github 中文自带传销属性。

- 不易得  
用 **consult-gh/forge/eww** 一分钟加载一个页面…
- 不圈子  
工具论 一切只是工具，工具只是乐趣，乐趣不是越多越好。

---

2025-11-23 更新：阴

- ① **leetcode:** 把题都抓完了，这还不开刷开学，想后半生吃草？
- ② **配 emacs:** 翻了半天别人的配置，就找两包 git-link, git-timemachine 感觉不错。

---

2025-11-22 更新：晴

- ① **C 语言:** 对 C 语言的复习还没开始。妈的，互联网
- ② **leetcode:** 完成了 0 题。妈的，互联网
- ③ **又后悔了:** 本地有 chromium, hosts06 忘了加，想用 chromium 看一下翻墙网站的兼容性，莫名发现谷歌搜索能用，又看起片来！
- ④ **从今天起:** 能只用 emacs 就只用 emacs，能只用 eww 就只用 eww，能问 gptel 就只问 gptel。eww 不能访问也别去试 firefox，firefox 不能访问也别去试 chromium。

---

2025-11-21 更新：晴

- ① **域名:** 为 columnddeeply/hosts 提交了 17868 行新域名，这个 hosts 是我最近在网上看到的最大的 host 文件，达到了近 410mb，共 12.576.671 行，专门用于 Porn，今天比较闲就把之前使用 tempermonkey 写的自动根据关键词用 ublacklist 自动 block 和从 fackads/和 github u3m8 collection 提出来的 cdn 网站和以前又闲又愤青时期屏蔽的各种网站，进行比较有二万三千行的不同 host，用后缀比如 gov/org/cn 过滤了一些自己动手又删了些，发现像很多云很多热门网站像 cloud.tencent.com./taobao/nvidia/xx.gov.cn 但用了这么久好像没有任何冲突，可能是我实在是没有怎么上网，只是用用 github 就够了。看起来很多，17868 行，几天检查不完，但是也就只是它的 703 分之一，我看了下很多是 blogspot/tumblr，互联网的域名真得是故意弄成这样难管？如果说 porn 的域名长成这样：028b2c9ad2a24433ab97b8e5dbf69597.mediailor.us-east-1.amazonaws.com(这还真是 porn)，这能管？设立 whitelist 比 blacklist 更容易，但网络的架构 (tcp/ip-dns-domain) 就不允许这种事，我也好久没有更新我的 host 了，用 ai 比用搜索引擎好用多了，ai 的用处就是过滤黄色网站，当然 openai 要开放成人内容这事说明：其实也没什么一定得搜的事。用 emacs 看文档，把记忆留在自己的脑子里，比起什么网页要重要的多。
- ② **折腾 emacs:** iedit/embark-next-symbol/forge。
- ③ **弄完了 leetgo:** 明天绝对开始刷题。
- ④ **研究了 github 热门 trending:** 发现中文代码质量之差，全是卖课/卖舆论分析/卖盗版书 pdf/卖前端 ai 的… fork 外国大神后各种引流的，ruanyifeng 周刊是可以的，但评论和想让他在 issue 里引流的项目也一言难尽的，看看 nixpkg 几千人才 22k 星，中文的一个 openwrt 教程就有 30k 星几百个人 contribute，看看 openai 也才几十 k 星，deepseek 上百 k 的星，看看各自的 issues，openai 很正常，deepseek 就好似精神病的故乡，冷清又时不时出来几个弱智问题，外国人看了怕是会笑掉大牙。

---

2025-11-20 更新：晴

- ① **leetcode:** 完成对 leetgo 工具的 elisp 改造。chatgpt 很给力。明天一定刷 leetcode。
- ② **real-mono-themes:** 今日 maintiner 还没有动，看来工作很忙啊。
- ③ **博客/pdf:** 加入了新的 org snippet，更深入学习了 org 转入 latex 更多方法，更完美的使用了本 latex 模板。

---

2025-11-19 更新：晴

- 
- ① 天气好，玩了一下午：结果踩在沥水槽腿掉下去了，还好只是擦伤。
  - ② **leetcode**：让 chatgpt 写 leetgo 的 elisp 工具，写到最后凌晨没网了！
  - ③ **后悔的一天**：reddit 里的 china\irl 我常常去看，实在是没意思，但总有一种引力让我去刷，莫名的又想看片，于是乎又成了看片加键政，什么高市什么献忠，吃瓜吃到 10 年的兽兽门，再看现在的其他事件，感觉 10 年代清新多了，那时候的主角还是 87 年“小妹”，现在则是 0 几年的“小妹”，真是世风日下，历史重是重复又重复，技术重是加强又如强啊，献忠机器人，何时到来？假韩炳哲的话改下：“互联网让道德世风内涵成了空气，一切除了炫/爽/性没有他用”。
  - ④ **折腾 nixos/emacs**：精简所有文件到博客这一个仓库。  
nixos 只用一个 flake.nix 和 secrets.yaml 再把一串 key 放到一个位置就能用了，简直是极简完了。emacs 只用一个 post-init.init，再 git clone 下 minimal-emacs 到位，org 再 src 运行一下，就能用了。这两都是 2000 行的配置。总之我有了一个 blog/pdf/emacs/nixos/snippet/hosts 集于一体的仓库。完全可重现，从配置到经历，其它的代码也可以从其它 repo 里 clone 下来，但总之目前这些就是我大学大概折腾的所有玩具了吧。越说越感自己的脑残了，明明有女生喜欢我，我只有折腾这些 sb 玩意去了！现在想起了，后悔得我都成了反科技主义了。
    - **越说越后悔**：总之，刷题刷项目，早点找工作，早点重新联系她。别再做 reddit 上的支人了…别再看什么瓜片了…反技术的后果就是“深山老林里的怪人”!!! 第一，你还有家人要养，第二，你也没生存能力，第三，我还是渴望家庭，第四，你都花了二十多年在技术上，现在去深山，那你学的拼音/加法/英文/编程都是屁吗？

---

2025-11-18 更新：阴

- ① **沉迷看书<sup>2</sup>** 错过了我设定的上网时间二点到五点<sup>3</sup>…明天再测试吧<sup>4</sup>
- ② **C 语言**：刷一遍 flag 代码，发现自己还是不太懂 C 语言。今天又没刷力扣，感觉自己很废，就这样还想搞什么？

---

2025-11-17 更新：雨

- ① **开始刷 Leetcode**：完成 0 题，折腾 leetgo 去了，使用 async-shell-command 与 emacs 不全好用，两者的时间不一致问题，shell-command 又卡 emacs 本身，两者同样不能管理弹出的 shell 输出。只能用完整的 shell 命令和在 async 里 eval emacs function 套娃看看。
- ② **将本“博客”的 html 和 pdf 上传了 github.io**：只是不只为何几十分钟了 qingsongliao.github.io 还没有，明天再看吧。
- ③ **完成了 real-mono-themes 的 emac 主题包**：从此 emacs 又多了个别具一格的主题，不过也许 70 年代的 emacs 就长这样吧，oldfasion never die。

---

2025-11-16 更新：阴

- **开始写作**
- **使用 elegant 模板**：搭建自己博客？写书还是写博客？干脆一起写吧。

---

<sup>2</sup> 把什么现代诗选和中国皇帝传都丢了吧，再看下去这辈子就毁了！只留下两本书，互联网浅薄与雅思 7 天词汇，一是提醒我互联网对大脑的“伤害”另一个是提升了我的大脑，最近几年或十年特别是高考后，感觉自己的脑子雾雾的。

<sup>3</sup> 我有点网瘾，所以通过家里的路由器限制一下。速度 200kps，限制热门社交媒体的域名，时间上只有下午二点到五点能用。每次无限制上网都有一种沉迷的感觉，看黄片刷新闻作为瘾症生意在互联网真是完美载体，总之，戒断的痛苦真是难受啊!!!

<sup>4</sup> 晚上又用母亲的手机上网了，怎么试都发现这个 github.io 是 404，下载下来好像是 html 本身的问题。切换了一下账号发现看不了那个号了，问了下 chatgpt 发现 github 有新号防 bot 功能，没法只能重新用回我的 NestorLiao 账号 [https://github.com/orgs/community/discussions/55609?utm\\_source=chatgpt.com](https://github.com/orgs/community/discussions/55609?utm_source=chatgpt.com)。

```
/home/leead/Leere/real-mono-them Head: main@b
B HEAD
real-mono-black.png      Rebbase: upstream/main Add packe
real-mono-girl.png       Push: origin/main ready to pu
real-mono-oldfashion.png Tag: v1.3.8 (1/89)
real-mono-white.png      Untracked files (2)
README.md                 Unstaged changes (4)
real-mono-black-theme.el  modified custom.el
real-mono-girl-theme.el  @@ -4,7 +4,28 @@B
real-mono-oldfashion-theme.el :: If you edit it by hand, you
real-mono-themes.el      :: Your init file should contain
real-mono-white-theme.el :: If there is more than one, t
+ '(org-agenda-files nil nil nil
+ '(custom-enabled-themes '(real-
+ '(package-selected-packages
+   '(aggressive-indent alert alx
+     alx-theme
+     consult
+     direcd-sub
+     eshell-tilde
+     hide-mode
+     marginalia
+     nix-mode
+     posix-mar
+     powerline
+     tilde-trace
+     zig-mode
+     (custom-set-faces
+       :: custom-set-faces was added b
+       :: If you edit it by hand, you
modified post-early-init.el
modified post-init.el
modified themes/purezen-theme.v
+ R = 01010 B = 10100
+ As we can see, the bits of R if Unpushed to origin/main (256+)
+ If we flipthese bits, we get 11 Unmerged into upstream/main (13)
+ a57bc19 main.h
+ Worst Case Time Complexity: O(1)
+ Space complexity: O(1)
+ Author [Yash Raj Singh](https://c727269 fix org theme
```

图 10.1: real-mono-old

## 参考文献

- [1] LI Q, CHEN L, ZENG Y. The Mechanism and Effectiveness of Credit Scoring of P2P Lending Platform: Evidence from Renrendai.com. *China Finance Review International*, 2018, 8(3): 256-274.
- [2] CARLSTROM C T FUERST T S. Agency Costs, Net Worth, and Business Fluctuations: A Computable General Equilibrium Analysis. *The American Economic Review*, 1997: 893-910.
- [3] QUADRINI V. Financial Frictions in Macroeconomic Fluctuations. *FRB Richmond Economic Quarterly*, 2011, 97(3): 209-254.
- [4] 方军雄. 所有制、制度环境与信贷资金配置. *经济研究*, 2007(12): 82-92.
- [5] 刘凤良, 章潇萌, 于泽. 高投资、结构失衡与价格指数二元分化. *金融研究*, 2017(02): 54-69.
- [6] 吕捷 王高望. CPI 与 PPI “背离” 的结构性解释. *经济研究*, 2015, 50(04): 136-149.
- [7] Ted Kazynski. 工业革命与人类命运. *社会学研究*, 2017(02): 54-69.

# 附录 A 开发中遇见的各种软件问题 TODOList

## A.0.1 General Contributions

内容提要	
<ul style="list-style-type: none"><li>□ org-mode, 提交 pr 增加 INCLUDE 的行末倒数行数的行为代码 x, 加入 mailing-list 提出 FR</li><li>□ C-plus-C-plus, 提交 leetcode 题解</li><li>□ melpa, 提交 pr 增加自己的 leetgo package, WIP</li><li>☒ consult-gh, 提交 pr 修改 readme 中的 consult-gh-search-code 的使用</li></ul>	<ul style="list-style-type: none"><li>☒ melpa, 提交 pr 增加自己的 real-mono-themes</li><li>☒ nixpkg, 提交 issue 增加 biospy 等 nix 生理信号包</li><li>☒ columnddeeply/hosts, 提交 pr 增加 17868 行新域名, 修改脚本为 0.0.0.0</li><li>☒ pyim, 提交 issue 修改% 报错行为</li><li>☒ addons.mozilla, eink 插件</li></ul>

## A.0.2 TODO write a eww-hook, when I use eww to visit a url, and if the site(not local file) exists more than 5 minutes, then download it to a dir I specific.

## A.0.3 TODO how to customize face as for specific mode?

e.g., I want to have default face in calendar or compilation-mode use mono font, but stay my other face use proportional font still... I know I can use set-face-attribute but the way of it is do it in globally. (set-face-attribute 'no-emoji nil :background (face-attribute 'default :background) :foreground (face-attribute 'default :background) :height 0.1)

## A.0.4 TODO Advent of code in zig?

## A.0.5 TODO package leetgo elisp package as emacs package

- description: do leetcode in emacs with leetgo
- require: wget/pandoc/leetgo/
- wget for fetch image
- pandoc for markdown to org
- leetgo for pick/commit questions
- simple init, use leetgo
- batch fetch, (leetcode-fetch-batch "1 2 3") to fetch 1/2/3 questions
- also you can fetch contest,
- auto detected browser cookie, no need to fill cookies by yourself
- offline test, use pre fetched testcase/makefile/awk to local test tasks.
- org format, local image, and generally you can easily export all questions.
- wait I finished doing leetcode

## A.0.6 DONE how to have pop out behavior like cape, but for my function's output?

(completion-in-region (point) (point) (list))

---

## A.0.7 ~~FIXME~~ How to modify org-mode's #+INCLUDE:

1. how to contribute to org <https://orgmode.org/worg/org-contribute.html> <https://orgmode.org/worg/org-mailing-list.html>

2. explain FR **Subject:** [FR] Add support for reverse (negative) line ranges in '#+INCLUDE:' ':lines'

Hi all,

Org's current '#+INCLUDE:' keyword supports selecting lines using the ':lines' parameter, e.g.:

- '5-10' — lines 5 to 10
- '-10' — lines 1 to 10
- '10-' — line 10 to EOF

However, Org currently does **not** support selecting lines counted from the end of the file.

I would like to propose extending the syntax to support "reverse line numbers", similar to how many tools allow negative indexing. Examples:

```
#+INCLUDE: "x.txt" :lines "~5-"      ; last 5 lines to EOF
#+INCLUDE: "x.txt" :lines "~5~-1"    ; last 5 lines to last 1 line
#+INCLUDE: "x.txt" :lines "--5"       ; from beginning to last 5 lines
#+INCLUDE: "x.txt" :lines "10~-1"     ; from line 10 to last 1 line
```

This would make it much easier to include "the last N lines" of logs, datasets, generated files, etc., without needing to preprocess the file externally.

If there is interest, I would be happy to help test or contribute patches.

Thanks! —Qingsong

## A.0.8 ~~DONE~~ how many hours of pc idle's power consumption is equal to hibernate save ram to disk?

## A.0.9 ~~DONE~~ almost same vscode-emacs package.

## A.0.10 ~~DONE~~ how to run current region's content (if not in region, run that line) into current project's vterm

current I get a command called "multi-vterm-project", when you are not in vterm and vterm is closed use multi-vterm-project to open(or turn on a vterm project shell window) it, and can use it again to toggle it close(not kill, just remove that window), if the project vterm window is open and cursor is not in that window, use multi-vterm-project will focus to vterm window. based on this, I want you to write run the current region/line's content into vterm, how it works?

Below is the exact design that matches your workflow **with 'multi-vterm-project' already handling window creation/toggle/focus**. You only need a function that:

1. Ensures the project vterm exists (by calling 'multi-vterm-project' when needed).
2. Gets the project vterm buffer.
3. Sends region or current line.
4. Does not change your current window unless the vterm does not yet exist.
5. If the vterm window is already visible but not focused, **do not move focus**; only send text.

Assumption: 'multi-vterm-project' creates or returns a vterm buffer named like '\*vterm-project: <project>\*'. If your naming differs, adjust the regexp.

Code:

```
"elisp (require 'project) (require 'vterm) (require 'multi-vterm)
```

---

```

(defun my/get-project-vterm-buffer () "Return the project vterm buffer, or nil if not exists." (let* ((proj (project-current t)) (root (project-root proj)) (name (file-name-nondirectory (directory-file-name root)))) (catch 'found (dolist (buf (buffer-list)) (when (string-match (format "\\\vterm-project: %s" name) (buffer-name buf)) (throw 'found buf))) nil)))

(defun my/ensure-project-vterm () "Ensure project vterm exists. Do NOT focus it if already exists." (let ((buf (my/get-project-vterm-buffer))) (unless buf ;; Creates and shows it. Your toggle logic is inside multi-vterm-project. (multi-vterm-project) (setq buf (my/get-project-vterm-buffer))) buf))

(defun my/send-region-or-line-to-project-vterm () "Send active region or current line to project vterm. Does not move focus unless no vterm existed." (interactive) (let* ((text (if (use-region-p) (buffer-substring-no-properties (region-beginning) (region-end)) (buffer-substring-no-properties (line-beginning-position) (line-end-position))))) ;; Make sure vterm exists (let ((vbuf (my/ensure-project-vterm))) (when vbuf (with-current-buffer vbuf (vterm-send-string text) (vterm-send-return))))))

(global-set-key (kbd "C-c e") #'my/send-region-or-line-to-project-vterm) ""

```

#### Operation summary:

If project vterm **does not exist**, this creates it by calling ‘multi-vterm-project’. If project vterm **exists but hidden**, the function does not toggle it; it simply sends text silently. If project vterm **exists and is visible**, the function does not steal focus; it sends text in place. Region has priority. No region → send current line.

This integrates cleanly with your workflow where ‘multi-vterm-project’ handles all window toggle/focus behavior.

### A.0.11 DONE pyim % 字符触发 Not enough arguments for format string

使用虎码码表<https://github.com/welandx/huma-danzi.pyim> readme 发现输入1时报错: Debugger entered-Lisp error: (error "Not enough arguments for format string") message("[1]: 1. 而 2.% (1/1) \$ ") #f(compiled-function (string position tooltip) " 使用 minibuffer 来显示 STRING。" #<bytecode -0x26f1e7f81176ed3>)("[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer) apply(#f(compiled-function (string position tooltip) " 使用 minibuffer 来显示 STRING。" #<bytecode -0x26f1e7f81176ed3>) "[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer nil) pyim-page-show("[1]: 1. 而 2.% (1/1) \$ " 620 minibuffer) pyim-page-refresh(nil) pyim-process-ui-refresh() pyim-process-run() pyim-self-insert-command() funcall-interactively(pyim-self-insert-command) call-interactively(pyim-self-insert-command) pyim-process-input-method(108) pyim-input-method(108)

删除 hmdz.pyim 中的% 或使用中文的%使用时不会报错, 发现 pyim 有%就报错: Debugger entered-Lisp error: (error "Not enough arguments for format string") message("[1]: 1.% (1/1) \$ ") #f(compiled-function (string position tooltip) " 使用 minibuffer 来显示 STRING。" #<bytecode 0x1d8cdaf8ba1c112a>)("[1]: 1.% (1/1) \$ " 624 minibuffer) apply(#f(compiled-function (string position tooltip) " 使用 minibuffer 来显示 STRING。" #<bytecode 0x1d8cdaf8ba1c112a>) "[1]: 1.% (1/1) \$ " 624 minibuffer nil) pyim-page-show("[1]: 1.% (1/1) \$ " 624 minibuffer) pyim-page-refresh(nil) pyim-process-ui-refresh() pyim-process-run() pyim-self-insert-command() funcall-interactively(pyim-self-insert-command) call-interactively(pyim-self-insert-command) pyim-process-input-method(108) pyim-input-method(108)

### A.0.12 DONE explain this wget command

wget -r -p -np -k <url> 含义:

-r 启用递归下载。会下载 <url> 页面以及它引用的其他页面或资源（根据递归规则）。

-p 下载页面显示所需的所有资源。包括图片、CSS、JS、字体等。这是“下载完整页面以便离线浏览”的选项。

-np 不进入父目录 (no parent)。递归下载时不会爬到 <url> 的上级目录。例如: 如果 <url> 是 <http://example.com/a/b/page.html> 它不会下载 <http://example.com/a/> 或 <http://example.com/> 下的内容。

-k 转换下载后的链接为本地相对路径。方便本地离线浏览时页面中的链接仍然能点击访问

---

## A.0.13 DONE how to do emacs overlay in nixos to config not to have somepackage natively exclude

how to disable gomoku, I am getting addiction to that game...

## A.0.14 DONE using consult-gh-notifications, and I just type wrong pass as my own pc's pass, not github token

it used to pop a dialog box at first time, then I input my linux user's password, then it just disappeared. why? I want to reinput, but it just no dialog anymore and it just show me: ghub-handle-response-error: HTTP Error: 403, "Forbidden", "<https://api.github.com/graphql>", ((message . "Request forbidden by administrative rules. Please make sure your request has a User-Agent header (<https://docs.github.com/en/rest/overview/resources-in-the-rest-api#user-agent-required>). Check <https://developer.github.com> for other possible causes.") (documentation\_url . "<https://github.com/magit/ghub/wiki/Github-Errors>"))

it's the issue of url-user-agent.

## A.0.15 DONE how the ~/.mozilla/firefox/profiles.ini looks like?

the profiles name is set in nix as " profiles.firefox = {} ", I want you to write the default looks of that file.

```
[General]
StartWithLastProfile=1

[Profile0]
Name=firefox
IsRelative=1
Path=firefox
Default=1
```

## A.0.16 DONE add sway emacs pkg?

## A.0.17 DONE how to make rm safer?

let rm delete file to trash just don't use term+shell directly, use dired/magit...

## A.0.18 DONE recentf moving file persisnt

every time I move my file and the path changed but the recentf didn't follow, causing production and memory lost. just maintin my own recentf list.

## A.0.19 DONE pull request to the porn site list project

fix the columnddeeply/hosts 's 127.0.0.0 to 0.0.0.0

by clone and compare the problem is that "I include a lost non-porn site in to it". not just "non-porn" sites, I also get a lot cdn server to block, which fetch from a lot of m3u8 streaming server. it may also block some movies which using the same server as porn sites... so, there is only KLUDGE for thing like blocking porn site, first question: what is porn? really? to be, the social media like twitter which spread sexual clip? the pornhub teaching mathematic? the reddit/4chan/zhihu/weibo/... even the taobao can sell sex toys with sexual pictures, so blocking internet is blocking porn really? I don't know, it's a question for everyone.

finished, by

```

#!/usr/bin/env bash
set -euo pipefail

HOST_URLS=(
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts00"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts01"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts02"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts03"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts04"
  "https://raw.githubusercontent.com/columndeeplly/hosts/main/hosts05"
)

TMPDIR=$(mktemp -d)
REMOTE_DOMAINS="$TMPDIR/remote_domains"
YOUR_DOMAINS="$TMPDIR/your_domains"
OUTPUT="hosts6"

echo "[*] Downloading hostlists..."
> "$REMOTE_DOMAINS"
for url in "${HOST_URLS[@]}"; do
  echo "uuuu->$url"
  curl -sL "$url" \
    | grep -E "^[0-9:\.]+"
    | awk '{print $2}' \
    >> "$REMOTE_DOMAINS"
done

echo "[*] Normalizing remote domains..."
sort -u "$REMOTE_DOMAINS" > "$TMPDIR/remote_sorted"

echo "[*] Extracting your domains..."
grep -E "^[0-9:\.]+" /etc/hosts \
  | awk '{print $2}' \
  | sort -u \
  > "$YOUR_DOMAINS"

echo "[*] Generating unique domains (hosts6)..."
comm -23 "$YOUR_DOMAINS" "$TMPDIR/remote_sorted" \
  | awk '{print "0.0.0.0", $1}' \
  > "$OUTPUT"

echo "[+] Done. Unique domains saved in $OUTPUT"
echo "uuuu Total: $(wc -l < "$OUTPUT")"

```

get uniq hosts, and update to github. it's I only get 23240 unique lines.

add 17868 new hosts

Used to maintain my hostslist, write a tempermonkey script with ublacklist addon to block chinese keyword search result automatically.

---

mainly it blocked sexual model gallary sites/Pirate JAV/and chinese porn m3u8 CDN which I filter from a github m3u8 collection repos/or anything non-programming like socialmedia/news/shopping..., occationally the script blocks org/edu/gov sites too.

I remove duplicate hosts in yours and delete non-porn sites, but there still too many which I can't checkout they all.

BTW, I didn't update my hostslist for a long time, because I just find blocking search engine and "hot" social-media instead porn sites is way more easier, I now use tavily/chatgpt/github in emacs only.

BTW, seems like use 0.0.0.0 instaed of 127.0.0.1 is faster.

### A.0.20 DONE make Leetcode fresh tasks list

- have already?
- yes/no, yes, edit that file, no, get the files(testcase/question.md/solution.xxx)
- edit the files question to include src in end, solution.xxx to have mode line in header.
- download image to local, turn md in org format, and delete md one.
- add a allorg, to include all under src dir's question.org
- 支持无网本地测试，在源代码文件头部中加入 mode line，在题目文件尾部中加入 include 源代码
- 根据题号在相同的目录下抓取不同题目和几个指定语言，并下载至本地图片，再转化 md 为 org 格式
- 支持在根目录中得到 src 的所有 org file 的 include
- md to org 的 == 问题
- based on leetgo to creat a leetcode elisp pkg
- test leetcode emacs pkg instead

### A.0.21 DONE learn how to reference in org

<empty citation> org-cite-insert unable to insert problem **M-RET** to enter cite list.

### A.0.22 DONE upload large files into the github repo

use sed to filter, use what to split into small files.

### A.0.23 DONE try to package real-mono-theme to melpa

minic almost-mono-theme, creat recipe, make pr wait for mainter check. I am familaring with github, feeling awesome!

1. reply from melpa maintiner

Thanks for this. I'm encouraged that you looked at other monochrome themes and went with a comprehensive approach here.

Typically themes are not byte compiled but you still want to maintain conventions across these "stub" files -

  [F]real-mono-sea-theme.el with byte-compile using Emacs 30.1:

  real-mono-sea-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line

  [F]real-mono-old-theme.el with byte-compile using Emacs 30.1:

  real-mono-old-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line

  [F]real-mono-girl-theme.el with byte-compile using Emacs 30.1:

  real-mono-girl-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line

  [F]real-mono-eink-theme.el with byte-compile using Emacs 30.1:

  real-mono-eink-theme.el:1:1: Warning: file has no 'lexical-binding' directive on its first line

  [F]real-mono-dark-theme.el with byte-compile using Emacs 30.1:

---

real-mono-dark-theme.el:1:1: Warning: file has no ‘lexical-binding’ directive on its first line

[F]real-mono-sea-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no ‘provide’ form.

[F]real-mono-old-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no ‘provide’ form.

[F]real-mono-girl-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no ‘provide’ form.

[F]real-mono-eink-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no ‘provide’ form.

[F]real-mono-dark-theme.el with package-lint 20250828.1506 and package-lint-main-file = "real-mono-themes.el":  
1 issue found: 1:0: error: There is no ‘provide’ form.

[F]real-mono-themes.el with melpazoid:

- real-mono-themes.el#L469: It’s safer to sharp-quote function names; use ‘#’

[F]real-mono-sea-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-sea-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-sea-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-sea-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-sea-theme.el:2: The footer should be: (provide ‘real-mono-sea-theme);;; real-mono-sea-  
theme.el ends here

[F]real-mono-old-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-old-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-old-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-old-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-old-theme.el:2: The footer should be: (provide ‘real-mono-old-theme);;; real-mono-old-  
theme.el ends here

[F]real-mono-girl-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-girl-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-girl-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-girl-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-girl-theme.el:2: The footer should be: (provide ‘real-mono-girl-theme);;; real-mono-girl-  
theme.el ends here

[F]real-mono-eink-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-eink-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-eink-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-eink-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-eink-theme.el:2: The footer should be: (provide ‘real-mono-eink-theme);;; real-mono-eink-  
theme.el ends here

[F]real-mono-dark-theme.el with checkdoc 0.6.2 (fix within reason):

real-mono-dark-theme.el:0: The first line should be of the form: ”;;; package — Summary” real-mono-dark-theme.el:0:  
You should have a section marked ”;;; Commentary:” real-mono-dark-theme.el:2: You should have a section marked  
”;;; Code:” real-mono-dark-theme.el:2: The footer should be: (provide ‘real-mono-dark-theme);;; real-mono-dark-  
theme.el ends here

[F]Package and license:

Please specify :fetcher before :repo in your recipe real-mono-dark-theme.el needs formal license boilerplate and/or  
an SPDX-License-Identifier real-mono-eink-theme.el needs formal license boilerplate and/or an SPDX-License-  
Identifier real-mono-girl-theme.el needs formal license boilerplate and/or an SPDX-License-Identifier real-mono-  
old-theme.el needs formal license boilerplate and/or an SPDX-License-Identifier real-mono-sea-theme.el needs for-  
mal lic

ense boilerplate and/or an SPDX-License-Identifier

---

#### A.0.24 DONE create new account's github repos for github.io

have a username.github.io public repo, have index.html in it, in [url](#)

#### A.0.25 DONE upload Purezen to github

- description: theme that really monochrome A collection of real monochrome emacs themes in a couple of variants.
- clone a monochrome and study it <https://github.com/cryon/almost-mono-themes>
- minic a theme to creat a repo <https://github.com/qingsongliao/real-mono-themes>

## 附录 B 嵌入式软件招聘常见要求 Embedded Jobs

- 本科及以上学历，嵌入式软件工作经验，电子、通信或计算机相关专业；
- 有 ARM Linux 软件的开发经验，熟练使用 C++ 和 C 语言；
- 熟悉 Linux 常用设备操作，有 spi、can、WiFi、audio、video 相关驱动开发经验优先；
- 熟悉多线程、多进程和 socket 网络编程，有并发程序开发经验和良好的设计思路；
- 有机器人相关嵌入式设计或者实时系统经验优先。
- 负责 yocto & android 的构建（Build）与升级（OS Upgrade），确保系统的稳定性和兼容性。
- 参与新硬件平台的 Bringup，包括 CPU、GPU、Memory 等核心组件的初始化和调试。
- 分析和解决系统稳定性问题，优化系统架构设计，提升整体性能和可靠性。
- 研究客户系统功能需求，定制和优化系统功能，满足客户特定场景的需求。
- 解决系统性能问题，包括 CPU、GPU、Memory 等资源的优化与调度。
- 主导基线升级（Baseline Upgrade），确保系统与最新技术和安全标准同步。
- 具备操作系统（如 Android、Linux）的构建与升级经验，熟悉系统启动流程和内核开发。
- 有硬件平台 Bringup 经验，熟悉 CPU、GPU、Memory 等核心组件的初始化和调试。
- 具备系统架构设计能力，能够优化系统性能并解决稳定性问题。
- 有基线升级经验，熟悉版本管理和代码合并流程。
- 具备客户需求分析和功能定制能力，能够根据客户需求优化系统功能。
- 熟悉性能优化工具（如 perf, ftrace, gprof 等），能够解决 CPU、GPU、Memory 相关的性能问题。
- 有嵌入式系统开发经验，熟悉低功耗设计和优化。
- 熟悉虚拟化技术（如 KVM, QEMU）和容器化技术（如 Docker, Kubernetes）。
- 有 MTK 芯片平台开发经验

## 附录 C 嵌入式招聘常见笔试问题

## 附录 D 嵌入式招聘常见面试问题