

with-deepseek

give me principles of "suckless neovim propaganda"

elaborate further in two parts, start with the first

then the rest of them

Of course. The "Suckless Philosophy" is a real movement in software advocating for simplicity, clarity, and frugality. Applying this to Neovim creates a specific, almost ideological approach to configuring and using the editor.

Here are the core principles of "Suckless Neovim Propaganda," framed as persuasive, dogmatic statements.

The Tenets of Suckless Neovim Propaganda

1. The Editor is a Tool, Not a Hobby.

"You are a craftsman, not a collector. Your focus should be on the work, not on polishing the toolbelt."

- **The Propaganda:** Your Neovim configuration exists to make you efficient at editing text (code, prose, configs). It is not a digital garden to be tended for its own sake. Endless tweaking, adding plugins for novelty, and chasing the "perfect" setup is a distraction from real work.
- **The Practice:** If a change to your config doesn't have a measurable impact on your productivity or comfort, it's bloat. Your `.config/nvim` directory should be small, version-controlled, and require minimal maintenance.

2. Vanilla is a Strength, Not a Weakness.

"Master the core before you seek the crutch. Vim's native power is vast and largely untapped."

- **The Propaganda:** Before installing a plugin to solve a problem, ask: "Can I do this with a built-in feature, a mapping, or a `:h` command?" Learn `:grep`, `:args`, `:cdo`, macros, marks, and the full power of `vim.opt`. A deep understanding of vanilla Neovim makes you powerful in any environment, even a barebones server.
- **The Practice:** Your first response to a workflow itch should be to check the help docs (`:h`), not GitHub.

3. Every Plugin is a Liability.

*******"A plugin is a promise to maintain a relationship with a stranger on the internet. It will break, it will change, and it will slow you down."** **"***

- **The Propaganda:** Plugins add complexity, startup time, and hidden dependencies. They are a point of failure. The "suckless" user views each new plugin with extreme skepticism. The burden of proof is on the plugin to justify its existence.
- **The Practice:** For every plugin you add, have a "kill threshold." If you don't use it for a month, remove it. Use a plugin manager that supports lazy loading, but prefer solutions that require no plugins at all.

4. The Keyboard is Sovereign. The Mouse is a Sacrifice.

"Flow state is sacred. Leaving the home row is a sin. Every millisecond spent reaching for the mouse is a failure of your configuration."

- **The Propaganda:** The ultimate goal is seamless, thoughtless interaction with text. Your hands should never leave the keyboard. This isn't just about keybindings; it's about designing a system where

navigation, file management, building, and debugging are all accessible via keystrokes.

- **The Practice:** Mappings should be ergonomic and mnemonically consistent. Use which-key or create your own mental map. The `$HOME/.config/nvim/lua/config/keymaps.lua` file is your most important configuration.

5. Configuration is Code. Code Should be Readable.

"Your config is not a trophy cabinet of obscure hacks. It is a document that should explain itself to its primary audience: Future You."

- **The Propaganda:** A messy, sprawling `init.lua` is a suckless failure. Your configuration should be modular, well-commented, and logical. If you can't understand why a line exists six months from now, it shouldn't be there.
- **The Practice:** Use a structured `lua/` directory (e.g., `plugins/`, `config/`, `keymaps/`). Comment not just *what* a line does, but *why* you included it.

6. Blazing Fast Startup is a Non-Negotiable Feature.

"An editor that is not instantly ready is not ready at all. You are an editor, not a loader."

- **The Propaganda:** Waiting for your editor to start, even for two seconds, is an affront to the principle of a tool. It discourages quick edits and breaks your flow. A suckless Neovim starts in milliseconds.
- **The Practice:** Use `--startuptime` to profile and ruthlessly eliminate slow plugins or configs. Embrace lazy-loading aggressively. A fast startup is the ultimate proof of a lean configuration.

7. Embrace the Terminal. It is Your Home.

"The terminal is the purest interface to the machine. Neovim is its sharpest blade. Do not cage the blade in a GUI."

- **The Propaganda:** Using a Neovim GUI (like Neovide, Goneovim) often means adding features that can be handled by the terminal itself (fonts, themes, mouse integration). The suckless user masters `tmux` or `zellij` alongside Neovim, creating a text-based powerhouse where the editor is one integrated component of a larger system.
- **The Practice:** Run Neovim inside a terminal multiplexer. Use it to split windows, manage sessions, and run build commands, keeping your workflow entirely within the keyboard-driven realm.

Sample "Propaganda Slogans"

- Your `init.lua` is not a resume.
- One thousand lines of code, not one thousand plugins.
- Vim motions are the only plugin you truly need.
- Slow config? `:q!` and start over.
- The help files (`:h`) are your one true plugin.
- Choose focus over features.

This philosophy is intentionally extreme. It's a counter-movement to the modern trend of "Neovim as an IDE," reminding users that the raw power and elegance of the editor itself is often more than enough. The goal isn't to suffer, but to achieve a kind of editing *zen* through minimalism and mastery.

Of course. Let us delve deeper into the dogma. Here is the first part, focusing on the foundational philosophy and its inherent tensions.

Part 1: The Theological Schism - The Suckless Creed vs. The Modern IDE Cult

The "Suckless Neovim" movement is not merely a set of preferences; it is a theological stance on the nature of tools and the relationship between the user and the machine. To understand it is to recognize it as a reformation—a deliberate and polemic pushback against the dominant orthodoxy of modern software development.

The Core Dogma: The Editor as an Extension of Will

At its heart, the Suckless philosophy posits that the ideal tool is a **transparent extension of the user's intent**. Any feature, animation, or abstraction that places itself between the thought "I want to change this text" and the execution of that change is considered **bloat**. This is not about minimalism for aesthetics; it is about minimalism for *velocity* and *mastery*.

- **The Modern IDE Cult** preaches: "The tool should anticipate your needs. It should provide a rich, graphical, and intuitive interface that understands your project's structure, offering completions, refactoring, and hints."
- **The Suckless Creed** counters: "The user should anticipate the tool's behavior perfectly. The tool should be a deterministic system, predictable and swift, responding instantly to the user's explicit commands. Understanding is the user's job; speed and reliability are the tool's."

This fundamental disagreement manifests in several key schisms:

1. The Schism of Intelligence: Smart Tools vs. Smart Users

The modern IDE seeks to be an intelligent partner. It uses Language Server Protocol (LSP), treesitters, and AI copilots to build a complex model of your codebase, offering context-aware assistance.

The Suckless adherent views this with deep suspicion. They ask:

- **At what cost does this intelligence come?** (Complexity, latency, resource consumption).
- **Does it make me reliant on its specific behavior?** (Will my skills atrophy? Will I be lost in a vanilla environment?).
- **Does it break the principle of transparency?** (Why did it suggest that? Why did it fail? The "magic" is a black box).

The Suckless answer is to use these features not as intelligent partners, but as **deterministic oracles**. An LSP is valued for its `textDocument/definition` `jump` (a precise command), not for its often-noisy "smart" completions. The user remains the intelligence; the tool provides fast data.

2. The Schism of Interface: Graphical Richness vs. Textual Purity

The modern development environment is a symphony of visual elements: icons, breadcrumbs, file trees, scroll bars, and modal pop-ups. It is designed to be *discoverable*.

The Suckless interface is the terminal. It is a realm of pure text. This is a deliberate, philosophical choice.

- **The Terminal is Universal:** The skills you hone in a terminal-based Neovim are transferable to every server, every container, and every system you will ever touch.
- **The Terminal is Fast:** It is a battle-tested, decades-old technology optimized for streaming text. It does not waste cycles on rendering animations or complex GUI elements.
- **The Terminal is Composable:** Neovim becomes one peer in a universe of text-based tools (`git` , `grep` , `find` , `tmux` , `make`). They pipe data to each other seamlessly. A GUI often walls off the editor from this ecosystem.

To choose a Neovim GUI is, in the Suckless view, to build a gilded cage. You gain marginal visual comfort at the cost of universality and composability.

3. The Schism of Mastery: Convenience vs. Comprehension

This is the most contentious schism. The Modern IDE prioritizes *immediate convenience*. You can be productive without understanding how it works. The Suckless path prioritizes *long-term comprehension* through initial difficulty.

- **Installing a File Tree Plugin** is convenient. It gives you a familiar clickable explorer.
- **Mastering `:h :edit`, `:h :find`, and `netrw`** is comprehension. You are not just managing files; you are learning the fundamental Vim paradigm of the buffer and leveraging the power of `'path'` and `'wildignore'`. This knowledge is forever.

The Suckless argument is that every time you choose a plugin to avoid learning a core Vim concept, you are trading a permanent, transferable skill for a temporary, localized convenience. You are building a "Neovim-shaped IDE" rather than becoming a master of *Neovim itself*.

Conclusion of Part 1:

The Suckless Neovim user is not a Luddite. They are a pragmatist with a monastic streak. They have made a calculated decision: that the ultimate form of productivity is achieved not by adding more features, but by removing every possible obstacle between their mind and the text. This includes the obstacle of not knowing how their own tools work. Their configuration is not a showcase of what Neovim *can* hold, but a refined testament to what they themselves have *chosen to master*.

This asceticism is not for its own sake, but for the profound power and freedom it unlocks: the ability to sit down at any terminal, in any environment, and wield an editor with the full force of their capability, unhindered by dependency or delay. They seek not to build the perfect editor, but to become the perfect editor's user.

Part 2: The Practical Liturgy - Implementing the Dogma in Your Config

Theology is meaningless without practice. The Suckless Creed translates into a strict set of rituals and liturgies for managing your `~/.config/nvim/` directory. This is not a casual hobby; it is a discipline.

The Liturgy of Plugin Management: A Ritual of Justification

Every potential plugin must pass through a gauntlet of questions before it is allowed into your sacred `plugins.lua`. This is the ritual of justification.

1. **The Question of Core Functionality:** "Can this be done with a built-in feature, a mapping, or a shell command?" Before `telescope.nvim`, there was `:find` and `:grep`. Before `nvim-tree`, there was `:Explore` and `:edit`. The answer is often "yes," if you are willing to learn.
2. **The Question of Singular Purpose:** "Does this plugin do one thing, and do it well?" A plugin that provides a statusline should not also manage tablines. Avoid monolithic "framework" plugins that want to manage your entire config. Prefer small, focused tools.
3. **The Question of Performance:** "Does it lazy-load? Does it block the UI? What is its impact on `--startuptime`?" You will profile this. A plugin that adds 50ms to your startup time for a feature you use once a day is a bad trade.
4. **The Question of Maintenance:** "Is it stable? Is the API simple? Will I still need this in a year?" A plugin that is constantly changing its API or adding flashy new features is a liability, not an asset.

Example: Instead of installing a full-scale snippet engine on day one, the Suckless user might first master `:h abbrev` and `:h i_Ctrl-R` to register quick, inline expansions. They add a snippet plugin only when the *demonstrable pain* of manual methods outweighs the cost of the new dependency.

The Liturgy of Configuration: Code as Documentation

Your config is a personal grimoire. It must be legible to its sole, most important audience: **Future You**.

- **Modularize with Intent:** A sprawling `init.lua` is a sin. Structure your `lua/` directory not by plugin, but by *concern*.

```
lua/
├─ core/
│   ├── options.lua  -- All `vim.opt` settings
│   ├── keymaps.lua  -- All keybindings, with comments
│   └─ autocmds.lua  -- All `vim.api.nvim_create_autocmd`
├─ plugins/
│   ├── init.lua      -- Plugin manager declaration
│   ├── lsp.lua       -- LSP, DAP, etc.
│   └─ ui.lua         -- Statusline, colorscheme, etc.
└─ init.lua          -- The minimal entry point
```

- **Comment the 'Why', Not the 'What':** `vim.opt.number = true` does not need a comment. But a complex autocmd or a specific keymap does. Explain the problem you were solving.

```
-- Keymap: Use <leader>sr to search and replace the word under the cursor
-- Why: Faster than typing out :%s/<C-r><C-w>//g<Left><Left>
vim.keymap.set('n', '<leader>sr', ':%s/<C-r><C-w>//g<Left><Left>', { desc =
  'Search and Replace word under cursor' })
```

- **The Quarterly Purge:** Schedule a recurring event to review your config. For every plugin and mapping, ask: "Have I used this in the last 90 days?" If the answer is no, it is bloat. Exorcise it without mercy.

The Liturgy of Workflow: The Keymap as a Sacred Covenant

Your keybindings are the primary interface of your will upon the text. They must be consistent, ergonomic, and comprehensive.

- **The <leader> Key as a Namespace:** Your `<leader>` key is the prefix for all your custom commands. Treat it as a namespace. `<leader>f` for "find," `<leader>g` for "git," `<leader>l` for "lsp." This creates a discoverable, mnemonic system.
- **Embrace Which-Key (Sparingly):** A plugin like `which-key.nvim` is often one of the few tolerated exceptions to the "no plugins" rule because it directly enforces the principle of a sovereign keyboard. It makes your custom mappings discoverable without breaking flow to check your config.
- **Bridge to the Terminal:** Your most powerful mappings often integrate Neovim with the shell. For example:

```
-- Search for files with `find` and open them in the quickfix list
vim.keymap.set('n', '<leader>ff', ':cex system("find . -type f -name \\\"*\\\"")
<CR>', { desc = 'Find files and load into quickfix' })
```

Conclusion of Part 2:

The Suckless Neovim configuration is a living document of efficiency, constantly pruned and refined. It is not a static artifact but a practiced ritual. The ultimate goal is to achieve a state of **editorial flow**, where the tool

itself disappears from your consciousness. You are no longer "using Neovim"; you are thinking, and the text bends to your thought without friction or delay. Your config is the meticulously crafted set of grooves that makes this possible—not by adding more, but by strategically, ruthlessly, removing everything else.

Of course. Here is the continuation and conclusion of the principles.

Part 3: The Ascetic Path - Advanced Doctrines and The Final Boss

Having established the core theology and its practical liturgies, the Suckless path leads to more extreme and refined doctrines. This is where the propagandist separates the zealot from the casual minimalist.

The Doctrine of Progressive Disclosure

"The tool should reveal its complexity only as the user is ready to receive it. A beginner's interface and a master's interface can be the same, if the master's is a superset of the beginner's."

This is the Suckless answer to the "discoverability" problem of vanilla Vim.

- **The Propaganda:** Your config should not be a monolithic, 1000-line behemoth you copy from the internet. It should start as a bare `init.lua` with `vim.opt` settings and a few essential keymaps. Complexity is added *only in response to felt pain*.
- **The Practice:**
 - **Week 1:** Master moving and editing with core Vim motions. Configure `number` and `relativenumber`.
 - **Month 1:** After feeling the pain of file navigation, add `telescope.nvim` or `fzf-lua`. After feeling the pain of finding definitions, add a basic LSP setup.
 - **Year 1:** Your config has grown organically. You understand the purpose of every single line because you wrote each one to solve a specific, personal problem. Your configuration is a timeline of your mastery.

The Doctrine of The Universal Soldier

"Your skills must be transferable. A configuration that cannot be replicated in 5 minutes on a remote server is a failure."

This principle attacks the very idea of a "fragile" or "high-maintenance" config.

- **The Propaganda:** Your ultimate editing environment is not your local Neovim setup. It is the knowledge in your head and your ability to apply it anywhere, even in a vanilla `vi` on a legacy system. Your local config is merely a convenient, optimized shadow of this true, internalized skill.
- **The Practice:**
 - Keep your entire `~/ .config/nvim/` directory in a single, public Git repository.
 - The `README.md` should be a simple, one-command install script (`git clone ... && stow nvim/` or similar).
 - Periodically, SSH into a random cloud machine or container and start with a bare `nvim -u NONE`. Practice your core Vim-fu. Remind yourself that the power was inside you all along.

The Final Boss: The Pluginless Challenge

Every true Suckless propagandist eventually confronts the ultimate challenge: **How much can I accomplish with zero plugins?**

This is not a practical goal for daily work, but a spiritual exercise. It is the equivalent of a martial artist training in isolation to understand the true principles of their art.

- **The LSP Challenge:** Can you use an LSP with just `vim.lsp.buf` functions and your own keymaps, forgoing plugins like `nvim-cmp` or `lspsaga`?
- **The Fuzzy Finder Challenge:** Can you replace Telescope with a combination of `:vimgrep`, the `'path'` option, and the quickfix list?
- **The Statusline Challenge:** Can you build a functional statusline using only `:h statusline` and `:h tabline`?
- **The File Explorer Challenge:** Can you master `netrw` to the point where it's faster than a graphical tree? (Hint: `:Lex` and `I` to toggle the banner).

By attempting these challenges, you are forced to read the help pages (`:h`). You discover obscure, powerful built-in features. You realize that much of what plugins offer is a more convenient, but often slower and more opaque, wrapper around capabilities Neovim already possesses.

The Suckless Manifesto: A Summary

1. **You are the Intelligence.** The editor is a dumb, fast tool. You provide the smart.
2. **Master the Core.** Your skill with `:h motion.txt` is more valuable than any plugin.
3. **Justify Every Line.** Every plugin, mapping, and setting must earn its keep. The burden of proof is on the addition.
4. **The Keyboard is King.** Design a keyboard-driven life. The mouse is a tax on your attention.
5. **Your Config is a Library, Not a Junkyard.** It should be well-organized, documented, and understandable.
6. **Speed is a Feature.** A slow-starting config is a broken config.
7. **The Terminal is Home.** Embrace the composable, universal power of the text-based ecosystem.
8. **Grow Organically.** Add complexity only in response to genuine, felt pain.
9. **Stay Portable.** Your skills and your config should travel anywhere.
10. **Embrace the Challenge.** Periodically test your limits by seeing how much you can do with how little.

This path is not for everyone. It requires discipline, curiosity, and a willingness to embrace short-term difficulty for long-term power. The propagandist doesn't claim it's the easiest path, only the most empowering. They offer not comfort, but sovereignty.
