

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.

with-gemini

give me principles of "suckless neovim propaganda"

elaborate further in two parts, start with the first

then the rest of them

Ah, you seek the **Suckless Neovim Propaganda** — not merely a configuration, but a *philosophy*. A creed. A quiet rebellion against bloat, complexity, and the cult of "more features."

Here are the sacred principles — written not in plugins, but in spirit:

🗡️ The Suckless Neovim Propaganda: 7 Commandments

1. Do Not Install Plugins You Cannot Explain in One Sentence

"If you can't describe what your plugin does without using the word 'magic,' you don't need it."

Every plugin is a dependency. Every dependency is a potential break. Every break is a moment stolen from your focus.

Rule of thumb: *If it doesn't fundamentally change how you think about editing, delete it.*

2. Your Config Is Not a Portfolio — It's a Tool

No screenshots of 200-line `init.lua` with 17 color schemes and animated status lines.

Your config should be:

- **Readable** by a stranger on a Friday night.
- **Small enough** to type from memory after a coffee spill.
- **Functional** — not fashionable.

The best config is the one you didn't write — it's the one you inherited from your fingers.

3. Use Native Features. Always.

Neovim ships with a *powerful* editor.

- `:grep + :copen > fzf`
 - `:terminal > tmux` pane hopping
 - `:vimscript > Lua` abstractions you don't understand
- If a native feature does 90% of the job — do 90% of the job.
The other 10%? You'll learn it. And you'll own it.

4. No Mouse. Ever.

The mouse is the first step toward becoming a GUI zombie.

Your hands belong on the home row.

Your eyes belong on the text.

Your brain belongs in the flow.

If you reach for the trackpad, you've already lost.

5. No Themes. No Icons. No Animations.

Color schemes are vanity.

Icons are distractions.

Animations are performance tax paid in focus.

Use `:set background=dark` or `:set background=light`.

That's it.

The *content* is the art. Not your status line.

6. Your `.vimrc` Is a Manifesto, Not a Library

Write it like a letter to your future self:

*"I used to think I needed 40 plugins. Now I know: I only needed `:help`, `:map`, and patience."
Keep it under 200 lines.
Comment like you're explaining to a 12-year-old who just got their first computer.
If it's not teachable, it's not suckless.*

7. The Ultimate Suckless Hack: Use Neovim Like a Text Editor

You are not a "developer."

You are a *writer*.

You are not "configuring an IDE."

You are *refining a chisel*.

Type. Delete. Search. Replace. Jump. Repeat.

That's all.

The rest is noise.



Epilogue: The Suckless Mantra

*"I do not need more features. I need fewer distractions."
"I do not need plugins. I need understanding."
"I do not need to impress. I need to edit."*

You are not installing Neovim.

You are *uninstalling* everything that stands between you and the text.

Welcome to the quiet rebellion.

*"The best editor is the one you don't notice."
— Anonymous, probably you, at 3 AM, typing `:wq` with perfect rhythm.*

P.S. If you're reading this and thinking "I need a plugin for that," you're already halfway there. Go delete your `lua/plugins/` folder. You'll thank me tomorrow.

Certainly. Let us begin.

Part I: The Philosophy — Why Suckless Neovim Is Not a Configuration, But a State of Mind

Suckless Neovim is not about minimalism as an aesthetic.

It is not about having a 50-line `init.lua` because it looks cool on GitHub.

It is not about "I only use 3 plugins" as a badge of honor.

It is about sovereignty.

You are not a user of Neovim.

You are its *custodian*.

Every plugin you install is a foreign agent in your workflow — a piece of code you do not control, written by someone who does not know your mind, running in your editor, consuming memory, delaying startup, breaking on updates, and subtly training you to depend on automation instead of understanding.

The suckless ethos says:

"You are the editor. The editor is you."

This is why native features are sacred.

- `:grep` is not "inferior" to `:Telescope live_grep` — it is *fundamental*.
It does not require Lua, async, or a dependency tree. It uses the system's `grep`.
You know where it is. You can debug it with `strace`. You can replace it with `ripgrep` in two keystrokes.
You own it.
- `:split`, `:vsplit`, `:tabnew` — these are not "outdated." They are *atomic operations*.
You do not need a plugin to "manage windows." You need to understand how buffers, windows, and tabs work.
Once you do, you can navigate any codebase with *intention*, not with mouse-driven chaos.
- `:help` is your only true documentation.
Read `:help usr_02.txt`. Read `:help mode`. Read `:help motion.txt`.
The manual is not a relic — it is the *source code of your competence*.

Suckless Neovim rejects the modern cult of "instant gratification editing."

It does not believe in "one key to rule them all."

It believes in *muscle memory forged through repetition*, in *mental models built over years*, in *the quiet mastery of a tool that never changes — because you learned it deeply*.

You do not need a plugin to auto-format your code.

You need to understand `gq` and `formatoptions`.

You do not need a plugin to jump to definitions.

You need to know `gd`, `gD`, `gf`, and `Ctrl-J`.

You do not need a plugin to search across files.

You need `:vimgrep` and `:copen`.

These are not "hard."

They are *unlearned*.

The modern editor ecosystem has trained you to believe that complexity is progress.

Suckless Neovim says:

Progress is clarity.

Complexity is decay.

And so you strip away.

Not because you fear features — but because you *respect your attention*.

You are not trying to be "minimalist."

You are trying to be *effective*.

And effectiveness is not measured in plugins installed —
but in problems solved,

in focus maintained,
in time reclaimed.

This is the first truth of Suckless Neovim Propaganda:

*You are not configuring an editor.
You are reclaiming your mind.*

We now turn to Part II — the practice. The rituals. The daily acts of rebellion.

Absolutely. Here is **Part II: The Practice — Rituals of the Suckless Neovim Adept**

Part II: The Practice — Daily Acts of Rebellion

Suckless Neovim is not a philosophy you read.
It is a discipline you *live*.

It is not enough to delete plugins.
You must *relearn how to think*.

Here are the daily rituals — the quiet, unglamorous, deeply powerful habits of those who edit without distraction.

1. You Start Neovim With No Plugins — Every Time

"If you can't edit your config file without plugins, you don't own your editor."

Every morning, before you open your project, you do this:

```
nvim -u NONE -N
```

Then you open your `~/.config/nvim/init.vim` (or `init.lua`) — and you edit it.

No treesitter. No linters. No auto-complete.
Just you. And the text.

Why?

Because if you can't *write* a function, *map* a key, or *fix* a syntax error without plugins —
you are not a programmer.

You are a plugin operator.

This is your daily calibration.
Your spiritual cleanse.

After you've edited your config, *then* you restart with your normal setup.
But now you know:
You could do it without.

2. You Write Every Mapping by Hand — No Copy-pasta

"If you didn't type it, you don't own it."

You do not copy `mapleader = '<Space>'` from someone's dotfiles.

You type:

```
nnooremap <space> :<c-u>
```

Then you test it.

Then you break it.

Then you fix it.

You write your own mappings for:

- `jj` → `<Esc>`
- `<leader>w` → `:w`
- `<leader>f` → `:Files` (*if you must*)
- `<leader>d` → `:Diagnostics`

You don't use `which-key` to show you what your keys do.

You *remember* them.

Because if you need a menu to tell you what your keys do,
you haven't learned them —
you've just memorized a UI.

Your hands know the path.

Your mind doesn't need a map.

3. You Edit One File at a Time — No Tabs, No Windows, No Panes Unless Necessary

"Multitasking is the enemy of deep work."

You open one file.

You edit it.

You close it.

You do not open 17 tabs because "I might need to look at that later."

You do not split windows to "see the test and the code."

You use `:edit filename` — and you *remember* where things are.

If you need to see two files at once?

Use `:split` — but only when you're actively comparing.

Then close it. Immediately.

Why?

Because every window is a cognitive load.

Every tab is a mental bookmark you're not sure you'll return to.

Every pane is a fragment of your focus scattered across the screen.

The suckless editor does not *manage* files.

It *holds* one file — and holds it *well*.

You do not need a file tree.

You need `:find` and `:e **/filename<Tab>` .

You do not need a project explorer.

You need `:cd ~/proj` and `:e .`

You are not lost.
You are *oriented*.

4. You Write Your Own Snippets — No SnipMate, No UltiSnips

“A snippet is a crutch for not knowing the language.”

You do not install a plugin to expand `for` into a 10-line loop.

You type:

```
for i in range(10):  
    print(i)
```

And you do it *again*.

And again.

After 20 times, your fingers remember.

If you *must* automate something, write a single function in your `init.lua` :

```
vim.api.nvim_set_keymap('i', 'fpr', '<Esc>:call MyForLoop()<CR>a', { noremap = true,  
    silent = true })
```

And then define `MyForLoop()` — one time — in your own code.

You do not install 500 snippets for every language.

You learn the *patterns*.

Because when you move to a new language,
you don't need a new snippet pack.

You *think*.

5. You Never Use the Mouse — Not Even Once

“The mouse is the first step toward becoming a user — not a creator.”

You do not click “Open File.”

You type `:e` .

You do not scroll with the wheel.

You use `Ctrl-u` , `Ctrl-d` , `j` , `k` .

You do not click “Close Tab.”

You type `:bd` .

You do not drag to select.

You use `v` , `V` , `Ctrl-v` .

You do not use the scrollbar.

You use `gg` , `G` , `30%` .

Every time you reach for the mouse —
you pause.

You breathe.

You type the key.

This is not about “efficiency.”

It is about *integrity*.

The mouse separates you from the machine.

Your fingers on the keyboard?

That’s communion.

6. You Rebuild Your Config From Scratch Once a Year

“If your config hasn’t broken in a year, you haven’t learned anything.”

Once a year — on the winter solstice, or your birthday, or the day you finally fixed that bug — you do this:

1. Rename `~/.config/nvim` → `~/.config/nvim.bak`
2. Create a new `~/.config/nvim/init.vim` — 10 lines.
3. Start from scratch.
Only add back what you *missed* — not what you remembered.

You do not copy your old config.

You *reconstruct* it from memory.

You ask yourself:

- Did I really need that color scheme?
- Did I ever use that plugin?
- Was that keybinding ever faster than typing the command?

You’ll be shocked.

You’ll realize:

I didn’t need 80% of it.

And you’ll feel lighter.

Not because your editor is smaller.

But because *you* are wiser.

7. You Edit in Silence — No Music. No Notifications. No Status Bars That Lie.

“The best status line is the one you don’t see.”

You disable the status line. Or you make it one line:

```
set statusline=%f
```

No git branch.

No battery.

No Python version.

No “123/456 lines.”

Why?

Because your editor should not remind you of the world outside.
It should hold you *inside* the text.

You do not need to know your branch.
You need to know your *intent*.

You do not need to know your CPU usage.
You need to know your *thought*.

You edit in silence.
You work in flow.
You close your eyes when you're stuck — not to look at a plugin's docs —
but to *think*.

The Final Ritual: The 5-Minute Test

Every Friday, at 5 PM — you do this:

1. Close all files.
2. Open `nvim -u NONE -N`
3. Open a new file.
4. Write 10 lines of code — in any language.
5. Save it.
6. Close Neovim.

No plugins.
No colors.
No help menus.
No Google.

If you can do it —
you are not a Neovim user.

You are a *writer*.

And that —
is the ultimate victory.

🏁 Epilogue: The Suckless Neovim Practitioner

You do not brag about your config.
You do not post screenshots.
You do not argue about plugins.

You sit.
You type.
You solve.

And when someone asks:

“How do you edit code so fast?”

You smile.
And say:

“I don't. I just stop waiting for the editor to do it for me.”

That is the Suckless Neovim Propaganda.

Not a cult.

Not a trend.

Not a GitHub repo.

A quiet, stubborn, *unyielding* return to the essence of editing:

Text.

Thought.

Time.

And you —

you are the one who remembers.

Now go.

Close this tab.

Open Neovim.

And edit something real.

with-mistral

give me principles of "suckless neovim propaganda"

elaborate further in two parts, start with the first

then the rest of them

The "**suckless Neovim propaganda**" is a playful, community-driven philosophy that advocates for minimalism, efficiency, and a "less is more" approach when configuring and using Neovim (or Vim). It draws inspiration from the broader [suckless](#) philosophy, which emphasizes simplicity, modularity, and avoiding unnecessary complexity in software.

Here are the core principles of "suckless Neovim propaganda":

1. Minimalism First

- **Avoid bloat:** Only use plugins, features, or configurations that you *actually* need. If a feature isn't essential, don't include it.
- **Default to vanilla:** Start with Neovim's built-in features before reaching for plugins. Many tasks can be accomplished with native functionality.

2. Manual Over Magic

- **Understand what you use:** Avoid "magic" configurations or plugins that abstract away how things work. Prefer solutions you can debug and customize yourself.
- **Write your own:** If a plugin does something simple, consider writing your own minimal version. This deepens your understanding and reduces dependencies.

3. Modularity

- **Keep it modular:** Organize your configuration into small, focused files (e.g., `1sp.lua`, `keymaps.lua`). This makes it easier to maintain, update, and reason about.
- **Isolate dependencies:** If you must use plugins, keep them isolated and easy to remove. Avoid plugins that pull in many dependencies.

4. Performance Matters

- **Fast startup:** Aim for a Neovim configuration that starts quickly. Avoid plugins or scripts that slow down your editor.
- **Efficient workflows:** Optimize for speed in your daily workflows. Use native features (like registers, macros, and motions) to minimize reliance on external tools.

5. No Unnecessary Abstractions

- **Avoid frameworkitis:** Don't use plugin managers, configuration frameworks, or abstractions unless they provide *clear* value. For example, prefer `lazy.nvim` for plugin management only if it actually simplifies your workflow.
- **Direct editing:** Prefer direct file editing over GUI-like abstractions (e.g., use `:edit` instead of file explorers unless absolutely necessary).

6. Embrace the Terminal

- **Terminal integration:** Use Neovim's terminal mode for tasks like running tests, REPLs, or shell commands. Avoid GUI tools when the terminal is sufficient.
- **TUI over GUI:** Prefer text-based interfaces and tools that work well in a terminal environment.

7. Documentation and Self-Sufficiency

- **Read the docs:** Rely on Neovim's built-in documentation (`:help`) and the manual. Avoid relying on tutorials or plugins to "hold your hand."
- **Learn the fundamentals:** Master Vim/Neovim's core features (motions, registers, commands) before adding plugins.

8. Community and Sharing

- **Share minimal configs:** If you share your configuration, emphasize minimalism and explain why you made certain choices.
- **Encourage simplicity:** Advocate for simpler solutions in discussions and help others avoid unnecessary complexity.

Example "Suckless" Neovim Config

A suckless Neovim configuration might look like:

- No plugin manager (or a minimal one like `lazy.nvim`).
- Only 3-5 essential plugins (e.g., `telescope.nvim` for fuzzy finding, `nvim-treesitter` for syntax highlighting).
- Custom keybindings and functions written in Lua.
- No GUI, no unnecessary themes or animations.

Why It Matters

The "suckless" approach is about **ownership** and **understanding**. By keeping your editor simple and transparent, you gain control, reduce frustration, and create a tool that truly fits your workflow.

Would you like help applying these principles to your own Neovim setup?

Part 1: The Philosophy Behind "Suckless Neovim"

The "suckless Neovim" philosophy is rooted in a broader movement that values **simplicity, transparency, and intentionality** in software. It's not just about making Neovim "minimal" for the sake of minimalism, but about **optimizing for understanding, control, and efficiency**. Here's a deeper dive into the philosophy:

1. Rejecting Bloat and Complexity

The term "suckless" originates from the suckless.org project, which critiques modern software for being bloated, over-engineered, and dependent on unnecessary abstractions. In the context of Neovim, this translates to:

- **Avoiding "kitchen sink" plugins:** Plugins that promise to do everything (e.g., all-in-one IDE plugins) often introduce complexity, slow down your editor, and make your configuration harder to debug.
- **Questioning defaults:** Many users inherit configurations from others without questioning whether they need every feature. Suckless Neovim encourages you to **start from scratch** and only add what you truly use.

2. The Value of Manual Control

Suckless Neovim is about **understanding how your tools work**. This means:

- **Writing your own functions:** Instead of relying on a plugin for a simple task (e.g., toggling a setting), write a custom function. This not only reduces dependencies but also deepens your understanding of Neovim's API.
 - **Avoiding "magic":** Plugins or configurations that abstract away how things work (e.g., auto-completion frameworks that hide the underlying LSP) can make troubleshooting difficult. Suckless Neovim prefers **transparency**—you should know what's happening under the hood.
-

3. Performance as a Priority

Neovim is already fast, but a suckless approach ensures it stays that way:

- **Startup time:** Every plugin or script you add increases startup time. Suckless Neovim aims for **instantaneous** startup by minimizing dependencies.
 - **Runtime efficiency:** Avoid plugins that run heavy processes in the background (e.g., unnecessary linters or formatters). Use Neovim's built-in features (like `:make` or `:terminal`) where possible.
-

4. Embracing the Unix Philosophy

The suckless approach aligns with the **Unix philosophy** of writing small, focused tools that do one thing well. For Neovim, this means:

- **Modularity:** Break your configuration into small, single-purpose files (e.g., `keymaps.lua`, `lsp.lua`). This makes it easier to maintain and update.
 - **Composability:** Use Neovim's built-in features (like `:!` for shell commands or `:read` for file operations) instead of plugins that replicate shell functionality.
-

5. Self-Sufficiency and Learning

Suckless Neovim is not just about the end result—it's about the **process of learning**:

- **Reading the documentation:** Instead of Googling for solutions, use `:help` to understand Neovim's native features.
 - **Debugging your own issues:** When something breaks, a minimal configuration makes it easier to identify the problem.
 - **Customizing intentionally:** Every line in your configuration should have a purpose. If you don't know why something is there, remove it.
-

6. Community and Culture

The suckless approach fosters a culture of **sharing minimal, well-documented configurations**. It's common to see users share their dotfiles with explanations like:

- *"I use this plugin because it solves X problem, and here's why I didn't use Y."*
- *"This is my custom function for Z, which replaces a plugin."*

This culture encourages others to think critically about their own setups.

7. The Aesthetic of Minimalism

There's also an aesthetic appeal to suckless Neovim:

- **No unnecessary UI clutter:** Avoid statusline plugins, animations, or themes that don't serve a functional purpose.

- **Focus on the text:** Neovim is a text editor—your configuration should prioritize editing text, not distractions.
-

Why Does This Matter?

The suckless approach is about **ownership**. When you understand and control your tools, you:

- Reduce frustration (fewer bugs, faster workflows).
- Gain flexibility (you can adapt your setup to any task).
- Develop deeper skills (you learn how Neovim and your system work).

Would you like to dive into **Part 2: Practical Steps to a Suckless Neovim Setup** next? This will cover how to apply these principles to your actual configuration.

Part 2: Practical Steps to a Suckless Neovim Setup

Now that we've covered the philosophy, let's dive into **how to apply these principles** to your Neovim configuration. The goal is to create a setup that is **minimal, efficient, and fully understood** by you.

1. Start from Scratch

- **Delete or archive your current config:** Begin with a clean slate. This forces you to intentionally add only what you need.
 - **Use Neovim's defaults:** Start with `nvim --clean` to see what Neovim can do out of the box. You'll be surprised how much is already built in.
-

2. Organize Your Configuration Modularly

A suckless config is **easy to navigate and maintain**. Here's a suggested structure:

```
~/.config/nvim/
├─ init.lua          -- Entry point (loads everything else)
├─ lua/
│   ├─ core/         -- Core settings (options, keymaps)
│   │   ├─ options.lua
│   │   ├─ keymaps.lua
│   │   └─ autocmds.lua
│   ├─ plugins/      -- Plugin configurations (if any)
│   │   ├─ lsp.lua
│   │   ├─ telescope.lua
│   │   └─ ...
│   └─ utils/        -- Custom functions
│       ├─ helpers.lua
│       └─ ...
└─ ...
```

- **Why this works:** Each file has a single responsibility, making it easy to update or remove features.
-

3. Minimize Plugins

- **Ask yourself:** *"Can I do this with Neovim's built-in features?"*
 - Example: Use `:grep` or `:vimgrep` instead of a fuzzy finder plugin for simple searches.
 - Example: Use `:terminal` instead of a plugin for running shell commands.

- **If you must use plugins**, follow these rules:
 - **One plugin per task**: Avoid "all-in-one" plugins.
 - **Lazy-load plugins**: Use a minimal plugin manager like `lazy.nvim` to load plugins only when needed.
 - **Avoid plugins with heavy dependencies**: Prefer lightweight alternatives.

Example minimal plugin list:

Plugin	Purpose
<code>nvim-treesitter</code>	Better syntax highlighting
<code>telescope.nvim</code>	Fuzzy finding (only if needed)
<code>lspconfig</code>	LSP support (no bloated frameworks)

4. Write Your Own Functions

Instead of relying on plugins for simple tasks, write your own Lua functions. Examples:

- **Toggle line numbers**:

```
function ToggleLineNumbers()
  vim.opt.number = not vim.opt.number:get()
end
vim.keymap.set('n', '<leader>n', ToggleLineNumbers, { desc = "Toggle line numbers" })
```

- **Custom file search**:

```
function LiveGrep()
  vim.ui.input({ prompt = "Grep: " }, function(input)
    if input then
      vim.cmd("grep! " .. input)
    end
  end)
end
vim.keymap.set('n', '<leader>g', LiveGrep, { desc = "Live grep" })
```

Why this matters: You understand exactly what the code does, and you avoid plugin bloat.

5. Optimize for Performance

- **Measure startup time**: Use `nvim --startuptime` to identify slow plugins or scripts.
 - **Lazy-load everything**: Use `lazy.nvim` to load plugins only when needed (e.g., load LSP only when opening a supported file).
 - **Avoid unnecessary autocommands**: Each autocommand adds overhead. Only use them for critical tasks.
-

6. Use the Terminal

Neovim has a built-in terminal (`:terminal`). Use it for:

- Running tests
- Git commands
- REPLs (e.g., Python, Node.js)
- Shell scripts

Example: Map a key to open a terminal in a split:

```
vim.keymap.set('n', '<leader>t', ':split | terminal<CR>', { desc = "Open terminal"
})
```

7. Master Neovim's Built-in Features

Before adding a plugin, ask: *"Can I do this with Neovim's native features?"*

- **Registers:** Use `:help registers` to master copy/paste.
- **Macros:** Record and replay macros (`q` , `@`).
- **Quickfix list:** Use `:make` and `:copen` for build systems.
- **Folds:** Use `:help fold` for code folding.

8. Document Your Configuration

- **Comment your code:** Explain why you added each setting or plugin.
- **Write a README:** Document your setup for future you (or others).

Example:

```
-- lua/core/options.lua
-- Enable line numbers for better navigation
vim.opt.number = true
-- Use system clipboard for yank/put operations
vim.opt.clipboard = "unnamedplus"
```

9. Share and Iterate

- **Share your config:** Publish it on GitHub with a clear explanation of your choices.
- **Iterate:** Regularly review your config and remove what you don't use.

Example Suckless Neovim Config

Here's a minimal `init.lua` to get you started:

```
-- ~/.config/nvim/init.lua
require("core.options")    -- Basic settings
require("core.keymaps")    -- Keybindings
require("core.autocmds")   -- Autocommands
require("plugins")         -- Plugin management (if any)
```

Final Thoughts

A suckless Neovim setup is **not about deprivation**—it's about **intentionality**. By focusing on what you truly need, you create a faster, more reliable, and more enjoyable editing experience.

Would you like help refining a specific part of your configuration, or do you have a particular use case in mind?

with-ms-copilot

give me principles of "suckless neovim propaganda"

elaborate further in two parts, start with the first

then the rest of them

Got it — you're asking for the **principles of "suckless Neovim propaganda"**. That phrase riffs on the *suckless* philosophy (from projects like `dwm` and `st`), which is all about **minimalism, clarity, and rejection of bloat**. Applied to Neovim, it becomes a tongue-in-cheek manifesto for how one should evangelize and practice Neovim usage. Here's a blueprint:

Principles of Suckless Neovim Propaganda

1. Minimalism Above All

- Every plugin is suspect until proven essential.
- Default Neovim is already powerful — propaganda stresses *less is more*.
- Configuration should be readable in one sitting, not a labyrinth of abstractions.

2. Transparency & Simplicity

- No magic. Every line in `init.lua` should be understandable by its author.
- Propaganda rejects "copy-paste configs" without comprehension.
- Favor explicit mappings over hidden automation.

3. Composability

- Small, orthogonal tools > monolithic frameworks.
- Each plugin should do one thing well, and play nicely with others.
- Propaganda encourages building your own toolkit, not adopting someone else's empire.

4. Environment Awareness

- Config must adapt to different OSes and environments without duplication.
- Propaganda celebrates scripts that detect and adjust, rather than hardcoding assumptions.
- Portability is a virtue.

5. Idempotency

- Running your setup twice should never break it.
- Propaganda insists on robust error handling and safe defaults.
- "It just works" is not enough — it must *always* work.

6. Reject Bloat Evangelism

- Fancy dashboards, bloated statuslines, and heavy IDE-like plugins are propaganda targets.
- The suckless Neovim propagandist says: *if you want an IDE, use an IDE*.
- Neovim is for editing text, fast and lean.

7. Know Your Tools

- Master native Neovim features before reaching for plugins.
- Propaganda emphasizes `:help`, motions, registers, macros, and Lua APIs.
- Plugins should extend, not replace, core knowledge.

8. Propaganda Style

- Speak in absolutes: “Plugins are lies,” “Your config is too big.”
- Use humor and exaggeration to make the point memorable.
- The goal isn’t purity, but to provoke thought and cut through complacency.

Example Propaganda Slogans

- *“Every plugin is a crutch — walk on your own.”*
- *“Your `init.lua` should fit in your head, not in a repo.”*
- *“Neovim is already an IDE — if you know how to use it.”*
- *“Suckless means ruthless: delete half your config today.”*

This is the kind of manifesto that thrives in hacker culture: provocative, minimalist, and uncompromising.

👉 Do you want me to **expand this into a full “propaganda pamphlet” style text** (like a satirical manifesto), or keep it as a set of guiding principles?

Perfect — let’s expand the **“suckless Neovim propaganda”** into a more elaborate two-part manifesto. I’ll start with **Part I: The Doctrine**, which lays out the philosophical foundations and rhetorical style. Then, in the next step, I’ll move to **Part II: The Praxis**, which will cover practical tactics, examples, and how to live the propaganda in your config and workflow.

Part I: The Doctrine of Suckless Neovim Propaganda

1. The Creed of Minimalism

- Neovim is already a cathedral of text editing. Plugins are merely pews — useful, but not sacred.
- Propaganda insists: strip away the excess until only the essential remains.
- A config bloated with 50 plugins is not power, it’s dependency. True power is knowing the motions, the registers, the Lua API.

2. The Gospel of Transparency

- Every line in your `init.lua` should be legible to you at 3 AM after two cups of coffee.
- Propaganda rejects “black box” frameworks that hide complexity behind layers of abstraction.
- If you can’t explain your config to a fellow hacker in five minutes, it’s not suckless.

3. The Orthodoxy of Composability

- One plugin, one purpose.
- Propaganda scorns “mega-plugins” that try to be IDEs inside Neovim.
- Instead, it preaches modularity: small tools that can be combined like Lego bricks, never welded into a monolith.

4. The Litany of Environment Awareness

- A config that breaks on macOS or Fedora is heresy.
- Propaganda demands adaptability: detect the environment, adjust paths, avoid duplication.
- Portability is not optional — it is the mark of a true propagandist.

5. The Commandment of Idempotency

- Run your setup once, run it twice, run it a hundred times — it must never fail.
- Propaganda elevates error handling and safe defaults to sacred law.
- A fragile config is propaganda against yourself.

6. The Denunciation of Bloat

- Dashboards, animated statuslines, and IDE-wannabe plugins are false idols.
- Propaganda declares: Neovim is not VS Code, nor should it aspire to be.
- The editor is for editing — speed, clarity, and precision are the only virtues.

7. The Catechism of Mastery

- Before plugins, there is `:help`. Before frameworks, there are motions.
- Propaganda teaches: know the native tools, then extend them.
- To rely on plugins without mastering the core is to worship shadows.

8. The Rhetoric of Propaganda

- Speak in absolutes: “Your config is too big.”
- Use humor, exaggeration, and provocation to cut through complacency.
- The goal is not purity, but awakening — to make others question their defaults.

✂️ That’s the **Doctrine** — the philosophical backbone of suckless Neovim propaganda.

Next, in **Part II: The Praxis**, I’ll show how to *apply* this doctrine: concrete propaganda tactics, config examples, slogans, and how to evangelize suckless Neovim in practice.

Would you like me to go straight into **Part II: The Praxis** now, or pause so you can react to the doctrine first?

Alright — let’s complete the manifesto with **Part II: The Praxis of Suckless Neovim Propaganda**. This is where the doctrine gets applied in practice: concrete tactics, config examples, and the art of evangelizing suckless Neovim.

Part II: The Praxis of Suckless Neovim Propaganda

1. Config Minimalism in Action

- Keep your `init.lua` under 300 lines — propaganda thrives on brevity.
- Example skeleton:

```
-- init.lua: suckless edition
vim.opt.number = true
vim.opt.relativenumber = true
vim.opt.ignorecase = true
vim.opt.smartcase = true

-- one plugin, one purpose
local plug = require("lazy").setup({
  { "tpope/vim-surround" }, -- text objects
```

```
{ "junegunn/fzf.vim" }, -- search
})
```

- Propaganda tactic: show others how much they can do with *just a handful* of plugins.
-

2. Propaganda Tactics

- **Mock Bloat:** “Your dashboard plugin is a screensaver, not a tool.”
 - **Praise Defaults:** “Neovim ships with LSP — why install three wrappers?”
 - **Public Demos:** Open a file, run macros, split windows, and show speed. The propaganda is in the performance.
 - **Slogans:** Short, punchy lines that stick. Example: “*Delete half your config, double your skill.*”
-

3. Evangelizing Composability

- Show how small plugins combine elegantly:
 - vim-surround + vim-repeat → composable editing.
 - fzf.vim + ripgrep → composable search.
 - Propaganda insists: “Two plugins in harmony beat one plugin empire.”
-

4. Environment Awareness in Practice

- Example Bash snippet for cross-platform setup:

```
if [[ "$OSTYPE" == "darwin"* ]]; then
  brew install neovim
elif [[ "$OSTYPE" == "linux-gnu"* ]]; then
  sudo dnf install neovim
fi
```

- Propaganda tactic: brag that your config works everywhere, unlike bloated setups tied to one OS.
-

5. Idempotency Rituals

- Always wrap installs in checks:

```
if not pcall(require, "lazy") then
  print("Lazy.nvim not installed")
end
```

- Propaganda line: “*Run it twice, it never breaks.*”
-

6. Anti-Bloat Campaigns

- Publicly reject heavy plugins:
 - No dashboards.
 - No “IDE-like” frameworks.
 - No plugins that duplicate built-in features.
 - Propaganda tactic: ridicule configs that look like VS Code inside Neovim.
-

7. Mastery Demonstrations

- Show off native Neovim power:
 - Complex macros (`q` , `@`)
 - Registers (`"a` , `"*`)
 - Motions (`cīw` , `dap`)
 - Propaganda insists: "If you don't know motions, you don't know Neovim."
-

8. Propaganda in Style

- Speak with humor, exaggeration, and provocation:
 - *"Your plugin manager is your crutch."*
 - *"If your config needs a README, it's already too big."*
 - *"Neovim is not your IDE cosplay."*
 - Propaganda thrives on memorable one-liners.
-



Closing Manifesto

- **Doctrine:** Minimalism, transparency, composability, awareness, idempotency, anti-bloat, mastery, rhetoric.
 - **Praxis:** Config examples, slogans, demos, ridicule of excess, celebration of defaults.
 - **Goal:** Not purity, but awakening — to make others *question their defaults* and embrace lean, suckless Neovim.
-

Fudzer, this is now a complete **two-part propaganda blueprint**: the philosophy and the practice.

Would you like me to **condense this into a punchy "pamphlet" format** (like a hacker zine page with slogans and short blurbs), or keep it as a manifesto for deeper study?

with-qwen

give me principles of "suckless neovim propaganda"

elaborate further in two parts, start with the first

then the rest of them

✨ Suckless Neovim Propaganda ✨

🚀 Why Suffer? Embrace the Suckless Philosophy for Your Editor.

Tired of feature bloat, endless configuration files, and editors that feel like a sluggish operating system? Join the movement that values **simplicity, clarity, and speed** above all else.

💡 What is Suckless?

It's a principle, not just a set of tools. It means:

- **Minimalism:** No features you don't need. The source code fits in your head.
- **Clarity:** Code that is easy to audit, understand, and modify.
- **Efficiency:** Tools that are lightning-fast because they are lightweight.

⚡ Neovim: The Suckless Editor (If You Configure It Right)

While Neovim is technically feature-rich, its power comes from its **extensibility** and **sane defaults**.

- **Forget VSCode/Emacs:** Stop installing 500 plugins just to get basic functionality.
- **The *Actual* Goal:** Use Neovim's built-in power (`lsp` , `treesitter` , etc.) combined with **only essential Lua plugins** to build a tailored, minimal, and *fast* environment.
- **Your Editor, Your Source Code:** Control every byte of your configuration. If you don't know what a plugin does, **delete it**.

Feature	Bloated Editor (The Enemy)	Suckless Neovim
Startup Time	Slow (Loading electron/megabytes of plugins)	Instantaneous
Configuration	Complex, multi-layered, opaque	Single, clean <code>init.lua</code>
Memory Use	Gigabytes	Megabytes
Philosophy	Add, add, add	Subtract, simplify, conquer

🔧 How to Go Suckless in Neovim

1. **Delete Your Plugin Manager:** (Optional, but very Suckless) Use built-in packages or a simple, fast manager like `lazy.nvim` —but use it *sparingly*.
2. **Learn the Built-ins:** Master the internal LSP, `telescope.nvim` (only one major fuzzy finder!), and Neovim's keybindings.
3. **Lua Everything:** Write all configuration in Lua. It's fast, simple, and the native way.
4. **One Tool, One Job:** Don't install five file explorers. Pick one great one (`nvim-tree.lua` or built-in `netrw`) and stick to it.
5. **Embrace the Terminal:** Use external terminal tools for tasks (like `grep`/`find`) instead of heavy GUI-style plugins.

"A tool should be as simple as possible, but no simpler." - Slightly adapted Suckless Proverb

🚩 Conclusion: Be the Master of Your Machine.

Stop being a victim of someone else's idea of a perfect editor. Stop the bloat. **Simplify. Master Neovim.**

#suckless #neovim #minimalism #speed

That's a great request! I'll elaborate further on the "**Suckless Neovim Propaganda**" in two distinct parts.

Here is **Part One: Deep Dive into the Suckless Philosophy and Neovim's Alignment**.

🧘 Part One: The Philosophy of Suckless and Neovim's Minimalist Alignment

The "Suckless" philosophy is often associated with the developers of the *suckless.org* software collection (like **dwm** and **st**), but it represents a broader principle that directly opposes the modern trend of software bloat.

📖 The Core Tenets of Suckless

The philosophy is governed by several strict, counter-cultural principles:

1. **Simplicity Over Complexity:** Suckless software strives to be **minimal**. It aims to solve *one problem* well and avoids creeping feature-ism (or *featuritis*). This means minimal codebases that are easy for the user to read, understand, and modify. If a feature can be accomplished better by another existing tool (like using an external file manager instead of embedding one), it should be omitted.
2. **Clarity and Transparency:** The source code should be small enough to "fit in your head." This low line count (LoC) reduces bugs, improves security (as the code is easy to audit), and makes the software **truly free**—free to be modified by the user.
3. **Use External Tools:** Rather than being a monolithic application that tries to do everything, Suckless tools often rely on the **Unix philosophy**—connecting small, single-purpose utilities together via pipes. This is a core reason why tools like **dwm** rely on external shell scripts for status bar information.

🧠 How Neovim Embodies This (or Can Be Forced To)

Neovim, due to its history and architecture, is an excellent candidate for a Suckless-compliant editor, provided the user adopts a disciplined approach.

- **The Power of Built-ins:** The most "Suckless" Neovim configuration is the one that uses the **least number of external plugins**. Neovim now ships with powerful, native components:
 - **Built-in LSP (Language Server Protocol):** Eliminates the need for monolithic language-specific plugins. The user only needs to configure the server and client setup once.
 - **Tree-sitter Integration:** Provides superior syntax highlighting and structural navigation, often replacing numerous older, slower regex-based highlighting schemes.
 - **Native Lua Configuration:** Lua is fast, lightweight, and directly integrated, making configurations much quicker to load than older Vimscript files.
- **The Configuration as Source Code:** A Suckless Neovim user treats their `init.lua` (or `init.vim`) not as a temporary script, but as a piece of **essential source code**. Every line must be justified. A user should be able to look at their config and know exactly what every line and every *required* plugin does.

- **Minimizing the Plugin Interface:** Instead of installing plugins for every small task, the Suckless approach dictates leveraging Neovim's core features (like registers, macros, and command-line execution) and using the terminal for external tasks:
 - Need to find files? Use `find` or `grep` in a terminal pane, not a slow, embedded file-search tool.
 - Need to view a file structure? Use the built-in `netrw` or a simple, fast tree-like utility, avoiding heavy GUI-styled explorers.

By following this minimal configuration path, you transform Neovim from a potentially heavy editor into a **highly optimized, bespoke tool**—a true manifestation of the Suckless ethos in an editing environment.

Would you like me to continue with Part Two, which will focus on specific configuration techniques and plugin choices to achieve this Suckless state?

Absolutely! Here is **Part Two: Specific Configuration Techniques and Plugin Choices for a Suckless Neovim Setup**.

Part Two: Techniques for a Truly Suckless Neovim Configuration

Achieving a "Suckless" editor experience requires discipline in how you choose and manage your Neovim extensions. It's about being **intentionally lean** and prioritizing the terminal's built-in power.

The "Don't Install" List

The first step to Suckless is subtraction. Avoid plugins that duplicate functionality better served by the terminal or Neovim's built-ins:

- **GUI-Style File Explorers:** Avoid heavy, mouse-centric tree plugins. Learn and use `:Ex` (the built-in `netrw`), or if you must, use a simple one like `nvim-tree.lua` only for projects.
- **Built-in Task Runners:** Use `make` or shell scripts run directly from a Neovim terminal (`:term`) or through simple mappings (e.g., `:!make`).
- **Non-Essential Eye Candy:** Avoid massive status-line generators, excessive animations, or tab-line managers. Keep it simple with the built-in status line or a minimal utility like `luafile.nvim`.

Plugin Discipline (If You Must)

If a task cannot be reasonably done with core Neovim, only choose plugins that follow the Suckless spirit: **small, fast, and Lua-based**.

Category	Suckless Choice	Why It Works
Fuzzy Finder	<code>telescope.nvim</code>	One powerful tool for all searching (files, greps, commands). Fast and native.
Plugin Manager	<code>lazy.nvim</code>	Extremely fast and allows for conditional loading, keeping startup instantaneous.
Completion/Snippet	<code>nvim-cmp</code>	Modular and essential for LSP use, but only install <i>necessary</i> sources.

Git Integration	fugitive.vim	A classic, minimal interface for Git that relies on Vim commands, not heavy pop-ups.
Syntax Highlighting	nvim-treesitter	The native, fast solution that replaces bulky regex-based syntax files.

Configuration Techniques

A Suckless Neovim configuration is **imperative**—it explicitly defines exactly what the user wants.

1. **Use Conditional Loading:** If you use a plugin manager like `lazy.nvim`, use its capabilities to only load plugins when they are actually needed (e.g., only load a specific language server when opening a file of that type). This ensures **zero overhead** on general startup.
2. **Mapping Over Commands:** Set up efficient keymaps that chain built-in commands together. For instance, mapping a single key to compile and then execute a script:

```
:nnoremap <leader>r :w <bar> !./run_script.sh<CR>
```

3. **Lean and Clean init.lua** : Structure your configuration into small, well-named files (e.g., `mappings.lua`, `lsp.lua`, `plugins.lua`). Crucially, your top-level `init.lua` should be a **manifest** that simply loads these modules, making the entire setup transparent and easy to audit.

The End Result: Pure Velocity

By adhering to these rules, your Neovim configuration becomes a true example of high performance through subtraction. You gain:

- **Near-Instant Startup:** No more waiting for hundreds of plugins to initialize.
- **Total Control:** Every feature in your editor is there because *you* decided it should be.
- **Minimal Resource Use:** Your editor runs efficiently, allowing your system resources to be dedicated to compilation or computation.

The Suckless way is about maximizing **user productivity** by minimizing software complexity. It's the ultimate customization through **deletion**.

Would you like me to find a specific video showing a very minimal, Suckless-style Neovim configuration?
