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., `lsp.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
`nvim-treesitter`	Better syntax highlighting
`telescope.nvim`	Fuzzy finding (only if needed)
`lspconfig`	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**:

```lua
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**:

```lua
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?
