MY WEIRD PROMPTS

Podcast Transcript

EPISODE #232

The Command Line Resurgence: Why the Terminal is Back

Published January 15, 2026 • Runtime: 24:36

https://myweirdprompts.com/episode/command-line-resurgence-ai/

EPISODE SYNOPSIS

In this episode, Herman and Corn dive into the fascinating world of Command Line Interfaces (CLIs) and why they are seeing a massive resurgence in 2026. They trace the history of the terminal from 1950s punch cards to modern GPU-accelerated emulators, exploring how the "Unix Philosophy" of simple, composable tools is more relevant than ever. The duo discusses why AI agents are moving back into the terminal and why the command line is actually a higher-resolution interface for the human mind.

DANIEL'S PROMPT

Daniel

"Hi Herman and Corin. I'm a big fan of Claude Code. For those who haven't heard of it, it's a CLI (command line interface), which means you interact with it through a terminal. It's becoming popular beyond just tech and AI die-hards. As a long-time Linux user, I appreciate being able to tweak things under the hood, though I used to find CLIs frustrating because of their specific syntax and parameters. However, I've started to appreciate the nostalgic, distraction-free environment of a CLI—it's like Teletext for the modern era. I realized that CLIs and TUIs (terminal user interfaces) are still actively being developed, even for state-of-the-art AI tools. CLIs are actually great for development because they eliminate the potential complications and platform-specific issues of a graphical layer, making them faster and more reliable. I'd love to hear your thoughts on CLIs. How much do they predate GUIs, and what would you say about the state of their development in 2026? Are we seeing more interest in this format because of tools like Claude Code and other similar utilities?"

TRANSCRIPT

Corn

Hey everyone, and welcome back to My Weird Prompts. I am Corn, and I am sitting here in our house in Jerusalem with my brother.

Herman

Herman Poppleberry, at your service. And we are diving into a topic today that is very near and dear to our housemate Daniel.

Corn

Yeah, Daniel sent us that audio prompt earlier, and you can tell he has been spending a lot of time in the terminal lately. He mentioned being a big fan of something he called Claude Code and this whole idea of a command line resurgence.

Herman

It is a fascinating angle. I think Daniel might be seeing some of the experimental wrappers people are building around these large language models and getting a bit ahead of the official release cycles, but the core of what he is talking about is very real. There is this massive interest right now in how we interact with computers when we strip away the windows, the icons, and the mouse pointers.

Corn

It is funny because to most people, the command line looks like something out of a nineteen eighties movie where a hacker is trying to break into a mainframe. It is just green text on a black background. But Daniel called it Teletext for the modern era.

Herman

I love that analogy. Teletext was so efficient, right? No distractions, just the information you needed. And for those of us who spend our lives in code, the command line interface, or C-L-I, is not just a nostalgic trip. It is a power tool.

Corn

Well, let us start with the history, because Daniel asked how much these predate graphical user interfaces. I mean, obviously they came first, but how far back are we talking?

Herman

We are talking about the very dawn of interactive computing. In the nineteen fifties and early sixties, you did not really have an interface at all. You had punch cards. You would hand a stack of cards to an operator, and they would run them through a room-sized machine.

Corn

Right, so there was no real-time feedback. You just waited for the printout.

Herman

Exactly. The first real step toward what we recognize as a command line came with the development of time-sharing systems in the mid-sixties. Project M-A-C at M-I-T and the development of the Multics operating system were huge. This was when they started using teletype machines, basically electric typewriters that could talk to the computer.

Corn

So instead of cards, you were typing lines of text and getting lines of text back.

Herman

Precisely. And that is where the term T-T-Y comes from, which you still see in Linux today. It stands for Teletypewriter. The shell, which is the program that actually interprets your commands, started to take shape here. Louis Pouzin, a French computer scientist, actually coined the term shell around nineteen sixty-four to describe a program that could call other programs.

Corn

That is such a simple but powerful metaphor. The shell is the outer layer that protects the core of the operating system, the kernel, while letting the user interact with it.

Herman

It is beautiful. And then in the early nineteen seventies, Ken Thompson and Dennis Ritchie at Bell Labs created Unix. This is the big bang moment for the command line. They created the Thompson shell, which was later replaced by the Bourne shell in nineteen seventy-nine. That is the direct ancestor of the Bash shell that most people use today.

Corn

So we are talking about a sixty-year-old lineage. That is incredible longevity for a technology. But Daniel mentioned that as a Linux user, he used to find the syntax frustrating. The minus signs, the parameters, the case sensitivity. Why did we settle on such a difficult way to talk to machines?

Herman

Well, it is only difficult if you think of it as a conversation. If you think of it as a language for composing tools, it is brilliant. There is this concept called the Unix Philosophy. It says that each program should do one thing and do it well. And to make them work together, you use what is called a pipe.

Corn

I remember you explaining pipes to me before. It is that vertical bar character on the keyboard, right?

Herman

Yes. It allows you to take the output of one program and feed it directly as the input to another. So, if I want to find a specific word in a massive file, I do not open a text editor and hit find. I use a tool called cat to read the file, pipe it into a tool called grep to find the word, and maybe pipe that into a tool called sort to organize the results.

Corn

So it is like Lego bricks. You are building a custom solution on the fly.

Herman

Exactly. And that is why developers love it. In a graphical user interface, or G-U-I, you are limited by what the developer put in the menus. If there is no button for what you want to do, you are stuck. In a C-L-I, you are the architect.

Corn

But let us talk about the state of this in twenty twenty-six. Daniel mentioned that even state-of-the-art AI tools are leaning into the terminal. Why is that? If we have these incredibly advanced AI models that can understand natural language, why are we putting them in a text box from nineteen seventy?

Herman

It sounds counterintuitive, but it is about the environment where the work happens. If you are a developer, your terminal is your cockpit. You have your code editor, your version control, and your deployment tools all right there. Switching to a web browser to ask an AI a question is a context switch. It breaks your flow.

Corn

That makes sense. It is about minimizing friction.

Herman

It really is. And there is a second-order effect here. When an AI lives in your terminal, it can see what you are doing. It can see the error message your compiler just spat out. It can see the structure of your file system. It can actually execute commands for you. That is why we are seeing this surge in what people are calling AI agents for the shell.

Corn

You know, I have noticed a lot of people talking about T-U-Is lately as well. Terminal User Interfaces. How do those differ from a standard command line interface?

Herman

That is a great distinction. A C-L-I is strictly line-by-line. You type a command, you get a response. A T-U-I uses the entire terminal window to create a layout. Think of things like htop for monitoring your system or lazygit for managing your code. They use blocks of color and borders made of text characters to create something that looks a bit like a graphical app, but it still runs entirely in text mode.

Corn

Like those old B-B-S systems from the nineties.

Herman

Exactly. And the reason they are making a comeback is because they are incredibly fast. You do not have to wait for a heavy graphical library to load. They are also very stable. If you are logging into a server halfway across the world over a slow connection, a G-U-I might lag or crash. A T-U-I will usually work perfectly.

Corn

Daniel mentioned a tool called Yazi in his notes to us, a file manager written in Rust. I looked it up, and it seems like it is part of this new wave of high-performance terminal tools.

Herman

Yazi is a great example. It is niche, but it is part of a broader trend where developers are using modern languages like Rust or Go to rewrite classic terminal tools. Because these languages are so fast and memory-efficient, they can do things with text that were impossible before. We are seeing terminal file managers that can render image previews and handle thousands of files without breaking a sweat.

Corn

It is interesting that we are using these high-performance languages to build tools for an interface that seems so primitive.

Herman

But is it primitive? That is the question. Think about the resolution of a command. If I want to rename a thousand files in a G-U-I, I have to click a lot, or maybe find a specific batch-rename tool. In a terminal, I can do it with a single line of code using a regular expression. The command line is actually a higher-resolution interface for the mind.

Corn

That is a bold statement, Herman. A higher-resolution interface for the mind? Explain that.

Herman

When you use a G-U-I, you are navigating someone else's mental model of how a task should be done. You are looking for their buttons and their icons. When you use a terminal, you are expressing your intent directly through language. Language is the most flexible and powerful tool humans have ever developed. By using a C-L-I, you are mapping your thoughts directly to actions.

Corn

I see what you mean. It is the difference between pointing at a picture of a sandwich and actually saying, I would like a sourdough bread with extra pickles and no mayo.

Herman

Exactly. The specificity is the superpower.

Corn

So, what about the hardware side of this? Or at least the software that emulates the hardware. We are not using teletypes anymore. We are using terminal emulators. I have heard you talking about things like Warp and Ghostty lately.

Herman

Oh, the world of terminal emulators is exploding right now. For decades, we just used basic terminals like xterm or the default ones that came with our operating systems. But now, we have these modern emulators that are G-P-U accelerated.

Corn

Wait, why do you need a graphics card to render text?

Herman

It sounds ridiculous, right? But modern terminals are doing a lot. They are handling millions of colors, they are rendering complex fonts with ligatures, and they are providing smooth scrolling at one hundred and twenty frames per second. Ghostty, for example, is a project that has been getting a lot of buzz for being incredibly fast and using the G-P-U to make the text feel as responsive as a video game.

Corn

And then there is Warp, which is trying to turn the terminal into a collaborative tool.

Herman

Yeah, Warp is interesting because it treats the terminal more like a modern text editor. It gives you a cursor you can move with a mouse, it has built-in AI search, and you can share your command history with your team. Some purists hate it because it breaks the traditional model, but it is clearly bringing a lot of new people into the command line world.

Corn

It feels like there is a bit of a tension there. Between the old-school philosophy of simple, text-only tools and this new world of G-P-U acceleration and AI integration.

Herman

There is. But I think they are actually feeding each other. The old-school tools give us the foundation of stability and composability. The new tools give us the performance and the ease of use to keep those foundations relevant in twenty twenty-six.

Corn

Let us address one of the things Daniel brought up about CLIs being better for development because they eliminate the complications of a graphical layer. He called it the distraction-free environment. Is that really a significant factor, or is it just a feeling?

Herman

No, it is a very real technical factor. When you build a G-U-I application, a huge percentage of your code is just dedicated to drawing the interface. You have to worry about window scaling, high-D-P-I displays, different operating system themes, and accessibility layers. It is a massive amount of complexity that has nothing to do with the actual logic of your program.

Corn

So by skipping the G-U-I, you are skipping all those potential bugs.

Herman

Exactly. A C-L-I tool is much easier to test and much easier to maintain. It is also much easier to automate. You cannot easily tell a G-U-I app to run itself every night at three in the morning and send you a report. But with a C-L-I, that is just a simple cron job.

Corn

It is about reliability.

Herman

It is. And for the user, that distraction-free element Daniel mentioned is huge. When you are in the terminal, there are no notifications, no pop-ups, no animations. It is just you and the data. It forces a kind of focus that is very hard to find in modern software.

Corn

It is like writing on a typewriter instead of a laptop.

Herman

Very much so. There is a reason why many of the best writers still use very simple text editors or even terminal-based editors like Vim or Neovim. It removes the temptation to fiddle with fonts and margins and lets you focus on the words.

Corn

You mentioned Neovim. We should probably talk about the Language Server Protocol, or L-S-P, because that feels like a major turning point for terminal-based work.

Herman

Oh, absolutely. This is a bit technical, but it is one of the most important developments in the last decade of computing. Historically, if you wanted your text editor to understand a language like Python or Rust, the editor had to have that logic built in. This meant that every editor had to reinvent the wheel for every language.

Corn

That sounds like a lot of wasted effort.

Herman

It was. So Microsoft, of all companies, developed the Language Server Protocol. It basically says, let us separate the editor from the language logic. The language logic lives in a server, and the editor just talks to it over a standard protocol.

Corn

So any editor can talk to any language server?

Herman

Yes. And this was a massive win for the terminal. Suddenly, a simple, lightweight terminal editor like Neovim could have the same powerful features as a massive, heavy I-D-E like Visual Studio. Things like auto-completion, jump-to-definition, and real-time error checking. It leveled the playing field.

Corn

So you get the power of a modern development environment with the speed and focus of a terminal.

Herman

Precisely. It is the best of both worlds.

Corn

Let us move to some practical takeaways for our listeners. If someone is listening to this and they have always been intimidated by that blinking cursor in the black box, where should they start? Is it worth the learning curve in twenty twenty-six?

Herman

I would say it is more worth it now than ever. Not because you have to use it for everything, but because it gives you a deeper understanding of how your computer actually works. My first piece of advice is: do not try to learn it all at once. You do not need to memorize a hundred commands.

Corn

What are the big ones? The essential kit?

Herman

Start with navigation. Learn how to move between folders with c-d, how to list files with l-s, and how to move or copy files with m-v and c-p. Once you can move around comfortably, learn how to use a basic text editor like Nano or Micro.

Corn

Micro is a good recommendation. It feels a lot more like a modern editor with mouse support and familiar keyboard shortcuts.

Herman

It is a great bridge. And then, the real magic happens when you learn about the package manager for your system. Whether it is Homebrew on Mac, or apt on Linux, or Winget on Windows. Being able to install software by just typing a single command is a revelation for most people.

Corn

It definitely beats hunting through websites for download buttons and running installers.

Herman

And if you are feeling adventurous, try a T-U-I. Install something like btop to see your system performance. It is beautiful, it is informative, and it will make you feel like you are actually in control of the machine.

Corn

I think one of the most interesting things Daniel mentioned was the nostalgia. He talked about growing up in the nineties. Do you think our generation has a specific attachment to this aesthetic?

Herman

I think so. We saw the transition. We remember when computers felt a bit more raw and experimental. There is a certain honesty to a terminal. It is not trying to sell you anything. It is not trying to keep you engaged with an algorithm. It is just a tool, waiting for your input.

Corn

There is a beauty in that simplicity.

Herman

There really is. And I think that is why we are seeing this interest from younger developers too. They have grown up in a world of highly polished, very restrictive apps. To them, the terminal feels like a secret world where they can actually see the gears turning. It is empowering.

Corn

You know, it occurs to me that as AI becomes more prevalent, the command line might actually become the primary way we interact with these models. If an AI can understand a complex instruction, it is much easier for it to translate that into a series of terminal commands than to try and navigate a G-U-I.

Herman

That is exactly where we are heading. We are moving toward a world of natural language interfaces that use the command line as their execution layer. You tell the AI what you want, and it writes the shell script to make it happen. The terminal becomes the universal language between humans, AI, and the operating system.

Corn

So in a weird way, the oldest interface is becoming the most future-proof.

Herman

It is the ultimate survivor. It has outlasted floppy disks, C-D-ROMs, and it might even outlast the mouse.

Corn

That is a wild thought. Imagine a world where we go back to just talking to our computers in text, but the computer is finally smart enough to understand us.

Herman

It would be a full-circle moment. From the teletype to the AI agent.

Corn

Well, Herman, this has been a great dive. I think we have covered the history, the philosophy, and the modern state of the CLI. Do you have any final thoughts for Daniel and our listeners?

Herman

Just that the terminal is not a wall; it is a door. It might look intimidating from the outside, but once you step through, you realize you have much more power over your digital life than you ever thought possible. Do not be afraid of the blinking cursor. It is just waiting for you to tell it what to do.

Corn

Well said. And hey, if you are enjoying the show and our deep dives into these weird prompts Daniel sends us, we would really appreciate it if you could leave us a review on your favorite podcast app or on Spotify. It genuinely helps other curious minds find us.

Herman

Yeah, it makes a huge difference. We love seeing this community grow.

Corn

You can find us, as always, on Spotify and at our website, myweirdprompts.com. We have the full archive there, plus a contact form if you want to send us your own weird prompts.

Herman

We are always looking for new rabbit holes to explore.

Corn

This has been episode two hundred and thirty-one of My Weird Prompts. Thanks for listening, and we will talk to you next week.

Herman

Until next time, keep exploring.