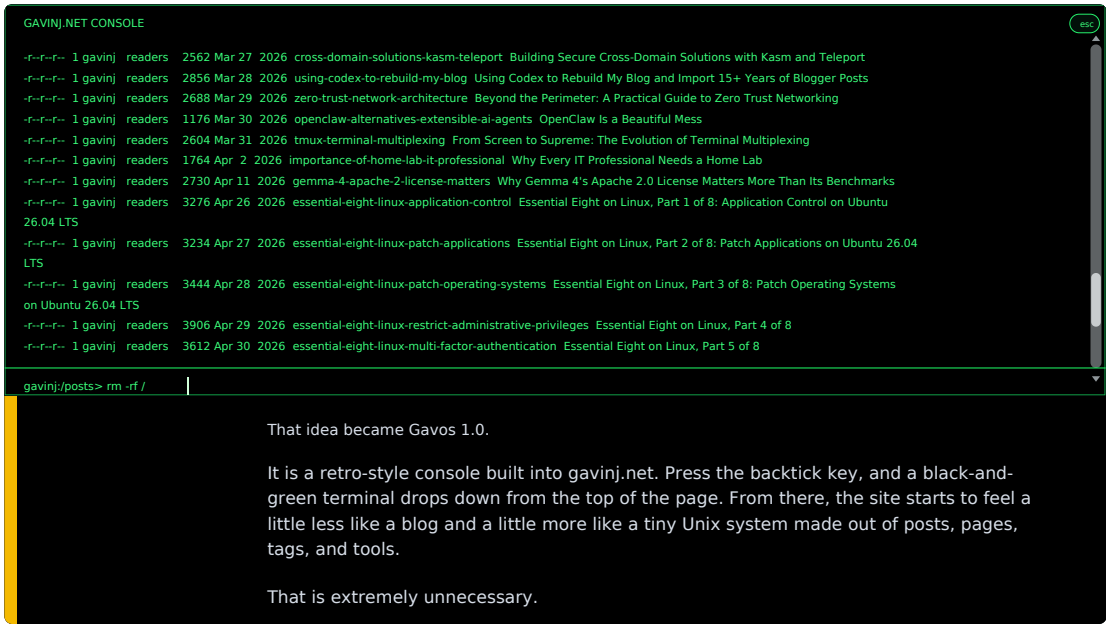
Introducing Gavos 1.0: A Command Line for gavinj.net

May 24, 2026 / Gavin Jackson

gavos blog command-line javascript php codex search tools



I had a fairly simple idea: what if my blog had a useful command line?

Not a fake terminal that only prints a joke and then runs out of ideas. Not a decorative "hacker" overlay that looks interesting for five seconds and then does nothing. I wanted something that actually helped me move around the site, search articles, inspect content, and run a few useful tools.

That idea became **Gavos 1.0**.

It is a retro-style console built into gavinj.net. **Press the backtick key**, and a black-and-green terminal drops down from the top of the page. From there, the site starts to feel a little less like a blog and a little more like a tiny Unix system made out of posts, pages, tags, and tools.

That is extremely unnecessary.

It is also exactly the kind of unnecessary that makes a personal website feel personal.

Why Add a Console to a Blog?

The practical reason was search.

The site already had normal navigation, tags, post pages, and a search interface. But I kept thinking about how much of my working life happens in terminals. If I want to find something, I reach for `grep`, `find`, `less`, `history`, tab completion, and muscle memory. Those tools are fast because they are

composable.

So the first version of the idea was modest:

```
search ubuntu
```

or maybe:

```
find linux
```

But once the console existed, it was hard not to keep going.

If articles have paths, then `ls` should list them. If `ls` lists them, then `ls | grep ubuntu` should work. If an article has Markdown source, then `cat`, `less`, `head`, `tail`, and `wc` should work. If tags are part of the site, then they should behave like directories. If there is a command line, it should have history, tab completion, manual pages, and a prompt that shows where you are.

That is how Gavos grew from "a quicker search box" into a small command environment.

Core Features in Gavos 1.0

The public feature set in Gavos 1.0 now includes:

- **Console access**: open the console with the backtick key, close it with `exit`, Ctrl+D, or the close button, and clear the screen with `clear`.
- **Help and documentation**: use `help`, `?`, and `man` to discover commands and read short manual pages.
- **Virtual navigation**: move around the site with `pwd`, `cd`, `ls`, and `tree`.
- **Article and page discovery**: search paths, titles, tags, dates, articles, and standalone pages with `find`, `tags`, and `grep`.
- **Direct navigation**: use `open` to jump to articles, pages, or tag result views.
- **Source viewing**: use `cat`, `less`, `head`, `tail`, and `wc` against article and page source.
- **Pipelines**: combine commands with pipes, including `grep`, `head`, `tail`, `wc`, and `sort`.
- **Shell comfort**: use command history, `!!`, `!NUMBER`, Ctrl+R reverse history search, tab completion, clickable output, and a prompt that shows the current virtual directory.
- **IPython in the browser**: start a Pyodide-backed IPython session with `python` or `ipython`, then return to Gavos with `.exit`.
- **Network and web tools**: run browser-safe versions of `host`, `nslookup`, `ping`, `traceroute`, `whois`, `certcheck`, and `curl -I`.
- **Utility commands**: use `date` for AEST and other time zones, `diceware` for passphrase generation, and `top` for a terminal-style view of popular articles.
- **State persistence**: keep console output, history, and current virtual directory available across page loads in the same browser session.

The Tiny Filesystem

Gavos treats the site like a tiny filesystem.

The prompt shows a fake current directory:

```
gavinj:/posts>
```

From there, the familiar commands work:

```
pwd
ls
cd /pages
cd /tags
tree /
```

Articles live under `/posts`, standalone pages live under `/pages`, and tags live under `/tags`. The illusion is intentionally small, but it is surprisingly useful. A blog already has structure; the console just gives that structure a command-line shape.

An `ls` of posts prints entries in a Unix-like long listing format, including permissions, owner, group, date, path, and title. The post paths are clickable, so the output is not only decorative. You can scan it like terminal output and still use it like a web page.

Tags behave similarly. You can list them, search them, move into them, or open a tag result page directly:

```
tags
tags linux
cd /tags/linux
open linux
```

That last one opens the normal blog tag view, so the console does not replace the site. It gives the existing site another way to be used.

Search That Feels Like Search

The console has a few different ways to find things.

For quick filtering, there is `grep`:

```
ls | grep ubuntu
```

For more structured searching, there is `find`:

```
find /posts -tag linux
find . -name ubuntu
find /pages -name resume
```

The search is not trying to be a giant external search engine. It is more like a site-local discovery tool. It searches article paths, titles, tags, dates, standalone pages, and other metadata that already belongs to the blog.

There is also tab completion for paths, so you do not need to remember full slugs. Start typing an article path, hit Tab, and the console helps finish it.

Reading Posts From the Console

One of my favourite parts of Gavos is that posts can be treated like files.

For example:

```
cat ubuntu-to-red-hat-enterprise-linux-2026
less ubuntu-to-red-hat-enterprise-linux-2026
head -20 ubuntu-to-red-hat-enterprise-linux-2026
tail -20 ubuntu-to-red-hat-enterprise-linux-2026
wc ubuntu-to-red-hat-enterprise-linux-2026
```

That works for standalone pages too:

```
cat resume
open resume
```

This is the kind of feature that would make no sense on most websites and complete sense on a Markdown-based technical blog. The article source is already there. Gavos just exposes it in a way that feels natural if you live in terminals.

`less` behaves like a pager rather than just dumping text into the console. You can scroll up and down, use Page Up and Page Down, and press `q` to return to the prompt.

Pipelines work as well:

```
cat resume | head -20
cat ubuntu-to-red-hat-enterprise-linux-2026 | wc
ls | grep linux | sort
```

It is not a full shell, but it borrows enough of the grammar to feel familiar.

Command-Line Polish

The small details make the console feel much more real.

Gavos includes:

- command history
- `!!` and `!42` style history expansion
- Ctrl+R reverse history search
- tab completion for commands, paths, pages, tags, and article slugs
- persistent console state between page loads
- clickable paths in command output
- `man` pages for the published commands
- `help` and `?` for quick discovery
- Ctrl+D and `exit` to close the console

None of those are individually huge, but together they change the feel of the feature. A command line without history or completion feels like a toy. A command line with just enough memory starts to feel like a place.

A Real IPython Interpreter

The most ambitious feature is:

```
python
```

or:

```
ipython
```

That starts an in-browser IPython environment using Pyodide. It supports multi-line Python input, tab completion, auto-indentation, and `.exit` to return to the Gavos shell.

This is wildly overpowered for a blog console, which is part of the appeal. Sometimes I want a scratch Python environment quickly. Now the site has one.

It also fits the spirit of the project. Gavos is not only navigation. It is a small toolbox.

Useful Network and Web Tools

Because this is a technical blog, it felt right to include a few tools I actually use.

Gavos includes browser-safe versions of familiar network commands:

```
host gavinj.net
nslookup gavinj.net
ping gavinj.net
traceroute gavinj.net
whois gavinj.net
certcheck gavinj.net
curl -I https://www.gavinj.net
```

Some of these use small PHP endpoints on the server side where the browser cannot do the job directly. For example, `curl -I` needs a proxy because browsers do not expose raw response headers in the same way a terminal does, and certificate inspection needs server-side help to retrieve public TLS certificate details.

The goal is not to replace a real terminal. The goal is to make the blog console useful enough that it earns its place.

There are also a few general-purpose tools:

```
date
date --help
diceware -n4 -d-
top
top -n 25
```

`date` shows the current AEST time by default, with options for other time zones. `diceware` generates passphrases from the EFF Diceware word list using browser cryptographic randomness. `top` reads the site's access-log-derived popularity data and renders the most viewed articles in a terminal-style bar chart.

Again, unnecessary.

Again, useful.

How We Built It

The best part of this project was the way it grew through collaboration.

I did not sit down with a complete specification for Gavos 1.0. The first idea was closer to "can we add a search bar?" Then the search bar moved around the navigation, gained a keyboard shortcut, became a slide-out control, and eventually sparked the idea of a drop-down command console.

From there, it became a steady back-and-forth:

- add the console
- make the prompt feel right
- add real commands
- make search work from every page
- make pages searchable as well as posts
- add tab completion
- make `less` behave like `less`
- keep console state across page loads
- make article paths clickable
- add manual pages
- wire in IPython

- add network tools
- clean up behaviour when navigating away
- make tags, pages, and articles feel like one coherent content system

That is where Codex was useful. Not because it magically knew what Gavos should be, but because it could keep up with the iteration.

I could describe a behaviour in plain language:

> *When I run `ls`, make it look like a Unix file listing, but use my article titles and dates.*

or:

> *When I type `cat resume`, pages should work the same way articles do.*

or:

> *Ctrl+R should behave like a proper reverse history search.*

Then we could make the change, test it locally, notice the rough edges, and refine it. The workflow felt less like generating a finished product and more like pairing on a feature that kept revealing what it wanted to become.

That style of development suits a personal site. There is room for taste. There is room for whimsy. There is room to say "this is objectively silly, but I like it" and then make it work properly anyway.

What Gavos Is Not

Gavos is not trying to be a secure remote shell, a server admin console, or a replacement for a real terminal.

It is a website interface.

That matters. Some commands are simulated. Some are backed by carefully scoped endpoints. Some are browser-side conveniences. The Python interpreter runs in the browser, not on the server. The filesystem is virtual. The article source comes from the same Markdown content that powers the blog.

That boundary keeps the feature fun without turning it into an operational liability.

Why I Like It

Gavos makes the site feel more like mine.

The normal web interface is still there. You can read posts, click tags, use navigation links, and never open the console. Nothing important depends on knowing commands.

But if you are the sort of person who enjoys terminals, there is now another layer waiting underneath the page. It rewards curiosity. It makes the archive more playful. It turns the blog into something you can poke at.

That is one of the things I still love about personal websites. They do not have to be optimised only for conversion, engagement, or a clean product funnel. They can contain small odd features that exist because the person who made the site thought they would be fun.

Gavos 1.0 is one of those features.

It started as an idea for faster search and turned into a tiny command line for the whole site.

I am very glad it exists.