Ron's Notebook

Better to Be Underestimated

August 22, 2026 · Ron
Canonical URL: https://promptforge.social/rons-notebook/better-to-be-underestimated/

I've been thinking a lot lately about what Dwarf Goats is becoming, and maybe more importantly, why I'm building it the way I am.

One thing I should make clear while I'm talking about all of this:

Dwarf Goats is the project codename.

It is not necessarily the final name of the operating system.

I'm still working through that part carefully. Naming something that you expect to live with for years is harder than it looks, especially when you want a name that actually belongs to the project instead of colliding with twenty other pieces of software.

I've narrowed it down to a small handful of candidates, but I'm not going to force the decision just to get it over with.

The right name will come.

Until then, Dwarf Goats is the name of the project that is building it.

Some of this goes back a long way.

I've been debugging code since I was a kid. One of the earliest tools I remember using was `TRON` in BASIC on the TRS-80 Color Computer. Trace on. You turned it on, ran the program, watched the line numbers go by and figured out where reality stopped matching what you thought the program was doing.

Then you fixed it.

That basic instinct never really left me.

Watch the system.

Understand what it is actually doing.

Don’t argue with reality.

Fix what reality shows you.

Decades later I’m doing basically the same thing with an operating system.

A lot of the time when I’m working I probably don’t look like I’m doing anything at all. I might be staring at a wall with a fan running in the background.

The fan matters. I like white noise when I’m coding.

And I’m usually not staring at the wall.

I’m running the system in my head.

I can mentally walk through an operating system, follow components, imagine where things connect, run failure cases, move pieces around and build before my hands ever touch the keyboard.

That has always been natural to me.

Then eventually I have to test the thing against reality.

That is becoming one of the most important ideas behind Dwarf Goats.

I want to actually use this operating system.

Not just boot it for a milestone.

Not just run a test harness against it.

Use it.

SSH into it. Sit at the `dwarf` shell. Run commands. Move around. Try things. Notice what feels good and what feels like hell.

If a command is missing, that tells us something.

If an error message sucks, that tells us something.

If networking feels awkward, that tells us something.

If something underneath the shell breaks during normal use, that tells us something.

If the system works but looks sterile or feels like somebody else's operating system, that tells us something too.

Those are my real audits.

My runtime audits are going to build Dwarf Goats both under the hood and cosmetically.

Use it.

Listen to it.

Learn from it.

Change it.

Use it again.

I am not trying to copy Linux.

That needs to be said clearly because I have enormous respect for Linux, GNU, GNU Mach, GNU Hurd and the generations of people whose work exists underneath what I am doing.

Dwarf Goats could not exist in a vacuum.

But respect does not mean imitation.

I don't want to reproduce somebody else's operating system culture just because that is what an operating system is expected to look like.

I want to carve our own path.

And I'm starting to understand who I want this system to be for.

Builders.

Pioneers.

Tinkerers.

Garage hackers.

Self-taught people.

Outsiders.

Loners.

People who want to understand the machine instead of being told they aren’t qualified to touch it.

People who have been underestimated.

I know something about being underestimated.

I don’t necessarily look like what some people expect a computer person or systems builder to look like.

Sometimes people make assumptions before they have any idea what you actually know or what you can build.

I’ve experienced that plenty of times.

But I have no interest in letting other people’s assumptions define me.

You observe.

You learn.

You adapt.

You build.

One of my strongest beliefs is that it is better to be underestimated than overestimated.

Being underestimated gives you room.

Room to listen.

Room to learn.

Room to make mistakes without an audience.

Room to improve.

Room to build capability while everybody else is busy deciding what they think you can or cannot do.

There is freedom in that.

I have another belief that goes with it.

Once people see what you can really do, there is no turning back.

I don’t mean that as a statement about power or status.

I mean that once people understand what you are actually capable of, things change.

Expectations change.

Attention changes.

Opportunities change.

Sometimes the responsibilities change too.

You cannot make people unknow what they have seen.

So I believe capability should be revealed deliberately.

Not by bragging.

Not by telling everyone what you are going to build six months from now.

Build it first.

Prove it.

Then show the evidence.

That is increasingly how I want Dwarf Goats to operate too.

Operate in stealth.

Not because there is something to hide, but because unfinished work does not always need an audience.

Listen.

Learn.

Build.

Defend.

I don’t want Dwarf Goats deliberately dumbed down either.

This isn’t about making a toy operating system for people who supposedly can’t handle the real thing.

Quite the opposite.

Give people real power.

But don’t make understanding that power dependent on membership in some priesthood.

Commands should be discoverable.

Errors should tell you something useful.

The system should let you see what it is doing.

Recovery should be understandable.

A curious person should be able to get underneath the surface, learn the machine and eventually take ownership of it.

The system should have personality too.

It doesn’t have to look like a sterile enterprise terminal because somebody decided a long time ago that serious computing has to look a certain way.

Dwarf should look and feel like Dwarf.

That philosophy reaches beyond the operating system.

I still remember buying technical computer books where the book didn’t just tell you about the software.

It came with the software.

I remember buying a Linux system-administration survival-type book and getting a CD-ROM with an early Slackware Linux system on it.

You could read about this strange operating system and then actually put the disc in a machine and use the thing.

That stayed with me.

I want to bring some of that spirit back when the Dwarf Goats book eventually happens.

I don’t want to write a book that just tells people about an operating system.

I want readers to be able to use the operating system the book is talking about.

Read about it.

Verify it.

Boot it.

Use it.

The important idea is simple:

The book should be a doorway into the machine.

And the machine should be something you can make your own.

There is another strange intersection in all of this.

I grew up debugging with things like `TRON`.

Now I’m building an operating system with frontier AI working beside me in a physical laboratory.

Those two worlds sound like they should have nothing to do with each other.

To me they connect perfectly.

The tools changed.

The basic method did not.

Observe.

Think.

Build.

Run it.

Watch what actually happens.

Fix it.

Repeat.

AI is part of how Dwarf Goats is being developed, but Dwarf Goats is not supposed to depend on AI to exist.

AI-developed.

Not AI-dependent.

I didn’t just give frontier AI a problem.

I gave it a laboratory.

And ultimately the machine still has to stand on its own.

There is one sentence that keeps coming back to me as all of this takes shape:

You don’t need permission to build.

You don’t need to look right.

You don’t need to sound right.

You don’t need the right pedigree.

You need curiosity.

You need persistence.

And you need something you want badly enough to make real.

That is who I want Dwarf Goats to belong to.

The builders.

The pioneers.

The people working quietly while somebody else is busy underestimating them.

Let them.

Build anyway.

-RLS

Provenance