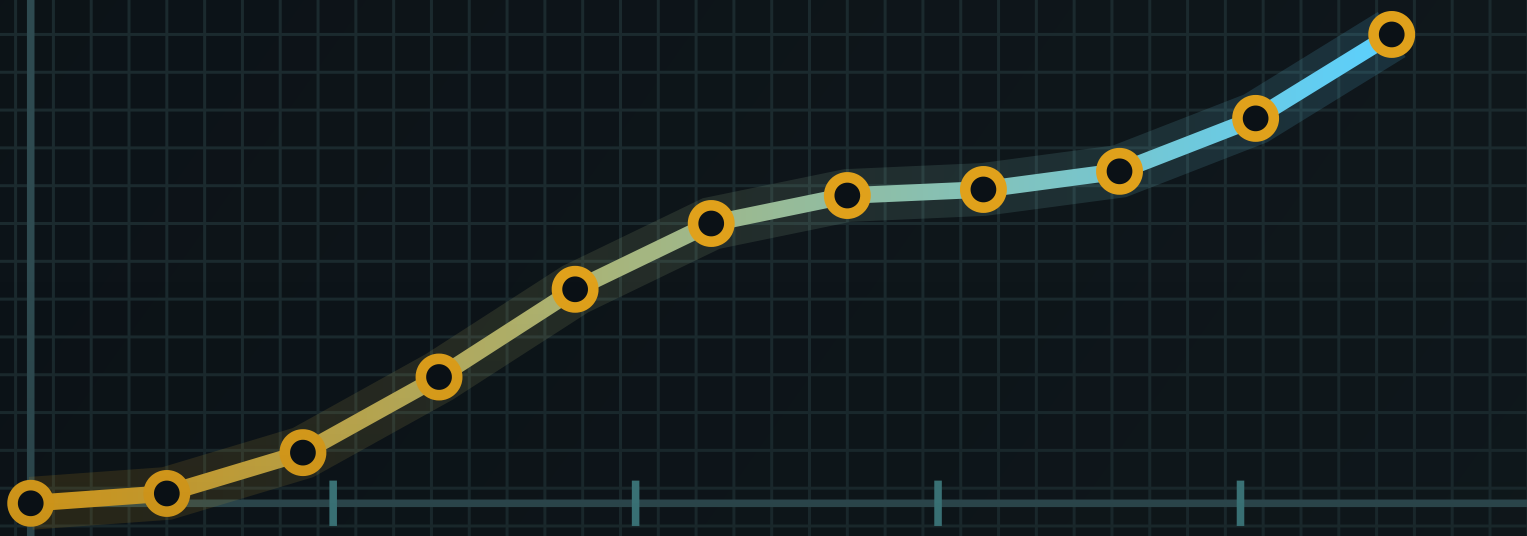
Vocabotics

A VOCABOTICS LAB REPORT

LAB NOTEBOOK · dated



THE LAB NOTEBOOK

An operating system, written by an AI

In November 2024 we set an AI to write a real 64-bit operating system in Rust from bare metal up. It reached 50,000+ lines across 11 subsystems and compiled with zero errors — then stalled on a 32/64-bit boot handoff we should have decided on day one. Kept on the page because the failure is the lesson.

The dated R&D archive

Research — measured, partial

COVER: A DATED NOTEBOOK GRATICULE WITH A PLOTTED, RISING MEASURED TRACE — THE LEAD OVER TIME

Vocabotics

A VOCABOTICS LAB REPORT

An operating system, written by an AI

In November 2024 we set an AI to write a real 64-bit operating system in Rust from bare metal up. It reached 50,000+ lines across 11 subsystems and compiled with zero errors — then stalled on a 32/64-bit boot handoff we should have decided on day one. Kept on the page because the failure is the lesson.

PREPARED

15 November 2024

VERIFIED BY

Jon Ross — Founder, Vocabotics — 15 years building safety-critical systems

HOW TO READ IT

Drafted with AI, verified by a named human. Sources at the back.

01 The entry, dated

The entry, dated

Nov 2024
entry date

Operating systems
category

~1-2 years ahead
lead over the world

🔓 Public
access

What the world had at the time

Late 2024: AI coding for most people meant autocomplete and ChatGPT snippets — a function here, a component there. The idea that you could point an AI at bare metal and have it produce a booting 64-bit kernel was not something people were seriously attempting in the open.

> *We asked whether an AI could write a whole operating system, not a snippet. It wrote fifty thousand lines of Rust and compiled clean — then got the boot mode wrong.*
>
> The costly signal

The measured result

Every number carries its own honesty tier — the tiering is the credibility.

50K+
lines of Rust, AI-written
🔬 RESEARCH

11
kernel subsystems
🔬 RESEARCH

0
compile errors at freeze
🔬 RESEARCH

1
architecture decision that sank it
🔬 RESEARCH

A year before this, in August 2023, we had gone back to bare metal by hand — a 16-bit x86 bootloader and a toy kernel, written to understand what actually sits beneath every web app. That curiosity left a question hanging: if we understand the bottom of the stack, could an **AI** build the whole thing? Not a function. The operating system.

02 What it was

The bet was deliberately extreme. Most people in late 2024 used AI to autocomplete a line or draft a component. We wanted to know if the same tools could hold a **bare-metal, 64-bit operating system** in their head — memory management, interrupts, a scheduler, drivers, a boot path — and write it in Rust, a language that refuses to compile most of the mistakes a kernel invites.

It was as much a test of *us* as of the AI: could a single person, directing an AI, produce something that normally takes a team and years?

03 What we built

A 64-bit ELF kernel designed to boot via GRUB using Multiboot, structured into **eleven subsystems** and reaching **over 50,000 lines of Rust**. The architecture separated a universal agent base, a kernel-agent layer, the bootloader, and the kernel proper — an early sign of the "everything is an agent" instinct that would define the year after.

🔬 RESEARCH — MEASURED, PARTIAL

The headline measured fact: at the point we froze it, the whole thing **compiled with zero errors**. Fifty thousand lines of systems Rust, generated and debugged with AI, that the Rust compiler accepted. For a language whose borrow-checker is famously unforgiving at this layer, a clean compile across a codebase this size is a real signal about what AI-assisted systems programming could do.

04 What we learned — including the honest negative

It did not boot. It got stuck on a **32-bit to 64-bit handoff mismatch** at the Multiboot stage — the processor comes up in one mode, and the kernel expected to be entered in another.

Why keep a kernel that never booted on the page?

Compiling clean is not the same as running. Zero compile errors measured the *code*; it did not measure the *machine*. Deleting this result would make the capability look cleaner than it was. The gap between "the compiler is happy" and "the hardware is happy" is exactly the thing a systems engineer has to respect, and an AI that writes fluent Rust does not automatically respect it for you.

The deeper lesson was about **process, not the AI**. The boot mode is one of the very first architectural decisions in an OS, and we had let it drift — made, effectively, too late to change without unwinding a lot of the 50,000 lines. The root cause was a decision made in the wrong order.

That produced a rule we still keep: **get the lowest-level decisions right first — everything above them inherits the mistake.** In safety-critical rail work, being wrong is a safety risk; here, being wrong at the bottom is a rewrite. Same discipline, cheaper stakes.

05 Where it went / status

Archived, honestly, as a **superseded prototype**. It was never resurrected as an OS — but the instincts it proved out carried directly forward: that one person directing an AI could hold systems-scale work; that the agent decomposition was the right shape; and that the bottom of the stack is where the real decisions live. The lab's later GPU-OS and inference work stands on this failure, not around it.

What is still open — kept visible

The honest edges of this lane, shown next to the wins. None of these is a footnote; each is a claim we have not yet earned — and marking it is what earns the right to be believed on the rest.

- [ ] It never booted end-to-end: a 32-bit/64-bit handoff mismatch stopped it at the GRUB Multiboot stage.
- [ ] Compiling clean is not running. Zero errors measured the code, not the machine.
- [ ] The lowest-level decision (boot mode) was made too late to change cheaply — the root cause was process, not the AI.

06 Sources, glossary & the ledger

Where every claim comes from — and what the plain words mean.

Sources

01 **Vocabotics project audit — AIOS / VocabOS (Rust kernel), Nov 2024** — Vocabotics internal project history (as of November 2024)

Vocabotics

THE HONESTY LEDGER

🟢 Shipped — measured, gate-clean.

🔬 Research — measured, partial; the gap is shown.

🔭 Vision — designed, honest; not yet built.

The bigger claim goes up a tier, never down. We show the eval beside the demo — the tiering is the credibility.

See the discipline this failure forced

Getting the lowest-level decision right first became a rule. Eighteen months later it became a cull of 57 projects down to one.

/lab/the-great-cull

Nothing faked.

A real, dated lab report from our own R&D. Measured, not asserted; the honest negatives are kept on the page next to the wins — because a lab that hides its failures forfeits the right to be believed on its results.

Vocabotics · calm, clear, candid · no client names · global rail & metro heritage