Context Protocol

The LLM is the CPU. Your files are the memory. You are the operating system.

THE PROBLEM

LLMs feel inconsistent because they are stateless. Chat history overflows. Attention decays. "Memory" features are shallow. The result: ideas resurface, decisions vanish, and trust erodes.

THE INSIGHT

Stop expecting AI to remember. Start acting like an Operating System. LLMs are excellent stateless processors. Decision memory, auditability, and long-horizon state are human responsibilities.

THE ARCHITECTURE

Component	Role	Implementation
LLM	CPU (processes, forgets)	Claude / ChatGPT / Gemini
Your Files	RAM (persistent state)	Plain markdown + Git
You	Operating System	Inject context, ratify outputs

THE SYSTEM

5 Commands	3 Constraint Tags
• CHECKPOINT — Export state as diff • SCOPE LOCK — Stay on topic • HARD STOP — Emergency brake • MODE: STRATEGY — Deterministic • MODE: EXPLORATION — Creative	• <locked_decisions> Decisions made. Don't revisit. • <rejected_ideas> Ideas killed. Don't resurrect. • <constraints> Rules to obey. No exceptions.

THE WORKFLOW

Start Session	End Session
1. Paste CORE_PROMPT.md 2. Paste your thread state 3. Work normally	1. Say "CHECKPOINT" 2. Copy structured output 3. Update state file → Git commit

WHY IT WORKS

Traditional AI Workflow	Context Protocol
Memory on their servers	Memory in YOUR files

Locked to one model	Model-agnostic (Claude, GPT, Gemini)
Vector DBs, embeddings, RAG	Plain markdown + Git
Agent decides next step	YOU decide, AI proposes
Opaque, proprietary formats	Auditable, readable forever

The LLM proposes. You ratify. The system records.

Open Source (MIT) — github.com/zohaibus/context-protocol