РАЗБОР МЕХАНИЗМА

Память Claude

Как агент помнит ваш проект между сессиями. И почему именно здесь у людей тихо ломаются агенты.

```
WARNING: MEMORY.md is 208 lines and 52.9KB. Only part of it was loaded.
```

@gorilla_under_hood · июнь 2026

ЗАВЯЗКА

Один warning, объясняющий половину «тупняков»

Каждая новая сессия Claude стартует с чистого окна контекста. Модель не помнит вчерашний разговор. Чтобы знание не обнулялось, есть память. Но память это не одна штука, а две, и именно поэтому их постоянно путают, а на стыке тихо теряют данные.

Я наткнулся на разрыв вживую. В начало сессии мне в контекст приехала та самая служебная строка с обложки. Перевод на человеческий: «твой индекс памяти разросся, я загрузил не весь». Звучит как мелочь. На деле это объяснение целого класса ошибок, который выглядит как «агент тупит, забыл то, что я говорил». Он не забыл. Он не увидел.

Два механизма, которые все путают

В Claude Code память это две системы. Обе грузятся в начале каждой сессии. Обе для модели **контекст, а не приказ** (к этому вернёмся).

СВЕРХУ ВНИЗ

CLAUDE.md

Кто пишет: вы

Что внутри: инструкции, правила, стандарты, архитектура

Грузится: каждую сессию, целиком

Пример: «используй pnpm, а не npm»

АГЕНТ САМ

Auto memory

Кто пишет: сам Claude

Что внутри: выученные факты, build-команды, ваши привычки

Грузится: первые 200 строк или 25KB

Пример: «build тут pnpm build, проверено»

Ментальная модель: CLAUDE.md это ваши инструкции агенту. Auto memory это записная книжка агента про вас. Первое идёт от вас к модели. Второе модель набивает себе сама, решая по ходу, что пригодится потом.

Как устроена auto memory под капотом

Работает у всех по умолчанию (нужна версия Claude Code 2.1.59 или новее). У каждого проекта своя папка памяти, привязанная к корню git-репозитория. Внутри файл-индекс `MEMORY.md` и сколько угодно topic-файлов, которые агент создаёт сам:

```
memory/
├── MEMORY.md ← индекс, грузится в каждую сессию
├── debugging.md ← детали по отладке
├── api-conventions.md ← решения по API
└── ... ← любые topic-файлы Claude создаёт сам
```

Ключевой нюанс загрузки

ЧТО ГРУЗИТСЯ	СКОЛЬКО
`MEMORY.md` (индекс)	первые **200 строк ИЛИ 25KB**, что наступит раньше. Дальше на старте не грузится
`CLAUDE.md`	целиком, независимо от длины
topic-файлы (debugging.md и пр.)	на старте не грузятся, агент читает по требованию

Архитектурный замысел: **MEMORY.md это тонкий указатель, а не склад.** Детали выносятся в topic-файлы, индекс остаётся компактным. Это не для аккуратистов, это условие работы: что не влезло в порог, того для агента на старте не существует.

НЕОЧЕВИДНАЯ СВЯЗЬ

Где это тихо ломается

Вернёмся к warning. Мой индекс был 208 строк и 52.9KB. Порог: 200 строк или 25KB, что раньше. 52.9KB вдвое больше 25KB, значит обрезка случилась по объёму, примерно на половине файла. Грубо: **половину собственной памяти агент на старте не видел.**

> Вы поправили агента → он честно записал в MEMORY.md → индекс со временем распух → нужная строка ушла за 25KB-порог → агент стартует без неё → спокойно повторяет старую ошибку. Он не врёт и не саботирует. Он физически не получил эту строку в контекст.

Это переворачивает диагностику. Когда агент «забыл» выученное, первый вопрос не «почему ты тупой», а **«эта запись вообще попала в загружаемую часть индекса?»**. Чаще всего проблема не в модели, а в раздутом нерасслоённом индексе.

Вторая ловушка: своя память молчит

Продвинутые пользователи (и я в этой системе) ведут **свою ручную папку памяти** параллельно платформенной: чистый каталог с профилями, проектами, выверенными

принципами. Он лучше служебного. Но **по умолчанию платформа автоматически инжектит только своё MEMORY.md**, а ваш ручной каталог на старте не подтягивает, если вы его явно не подключили.

> **И тут же решение.** «Не подтягивает» верно лишь пока вы не провели каталог в загрузку сами. CLAUDE.md грузится целиком (без лимита 200/25KB) и поддерживает **@import**: по доке импортированные файлы разворачиваются и грузятся в контекст на старте. Пропишите `@путь/к/каталогу` в CLAUDE.md или вынесите инструкции в `.claude/rules/`, и ручной каталог будет подхватываться автоматически каждую сессию.

Парадокс по умолчанию: худшая, обрезанная половина платформенного индекса грузится всегда и автоматически. Лучший вылизанный каталог молчит, пока вы не подключите его явно. Два хранилища живут параллельно и не знают друг о друге, пока вы их не свяжете. Стык между ними и есть корень большинства «провалов памяти».

Гвоздь: память это контекст, а не закон

Ни CLAUDE.md, ни auto memory не являются жёсткой конфигурацией. Официально модель трактует их как контекст, а не принудительную настройку. Более того, CLAUDE.md технически доставляется как сообщение пользователя после системного промта. Модель читает и старается следовать, но гарантии строгого исполнения нет, особенно для размытых или противоречивых инструкций.

> **Практический вывод.** То, что должно срабатывать всегда и безусловно (запретить команду, прогнать линтер перед коммитом, заблокировать путь), пишется не в память, а в **hook** (PreToolUse и подобные). Hook это shell-команда на фиксированном событии, она исполняется независимо от решения модели. Память влияет на поведение. Hook его обеспечивает. Путать эти уровни дорого.

ДЕЙСТВИЕ

Что с этим делать

1. **Держите MEMORY.md под порогом.** Перевалил за 200 строк / 25KB, часть памяти уже не работает. Расслоите: указатели в индексе, контент в topic-файлах.
2. **Проверяйте, что грузится**, командой `/memory`. Нет файла в списке, агент его не видит.
3. **Подключите свой каталог явно.** По умолчанию автоматически грузится только служебный MEMORY.md. Хотите, чтобы ручной каталог тянулся сам, пропишите

`@import` в CLAUDE.md или вынесите в `.claude/rules/` (грузится на старте целиком, без лимита).

4. **Срочное и безусловное выносите в hooks**, а не в память.
5. **Пишите конкретно и коротко.** Проверяемые формулировки слушаются надёжнее расплывчатых.
6. **Чистите периодически.** Противоречивые записи модель разрешает произвольно, то есть может выбрать неверную.

Где этот разбор может быть неполон

Cowork отличается от ванильного Claude Code. Базовый механизм auto memory описан по официальной доке Claude Code. Cowork построен поверх, но хранит память в своём служебном пути и использует слегка свой формат записей (frontmatter с типами и связями через двойные скобки). Механизм тот же, обёртка другая.

Цифры могут сдвинуться. Порог 200 строк / 25KB и версия 2.1.59 актуальны на июнь 2026. Это продуктовые параметры, проверяйте по текущей доке.

«Половина памяти не грузится» это про конкретный распухший файл, а не общий закон. При аккуратном индексе на 80 строк обрезки нет вовсе.

Калибровка. Механика двух систем, лимит загрузки, разделение «контекст vs hook»: подтверждено офдокой, уверенность высокая. Тезис «стык двух хранилищ = корень большинства провалов памяти»: обобщение из наблюдаемого паттерна, а не замеренная статистика, уверенность средняя.

Источники:
How Claude remembers your project, Claude Code Docs: docs.claude.com/en/docs/claude-code/memory
Memory tool (отдельная API-фича, не путать с auto memory): docs.claude.com/en/docs/agents-and-tools/tool-use/memory-tool

GORILLA UNDER HOOD

Разбираю, как AI-агенты устроены внутри

Механика под капотом, рабочие сетапы и грабли, на которые натыкаешься вживую. Без инфоцыганства, на своих кейсах.

Наведи камеру → подпишись

@gorilla_under_hood

t.me/gorilla_under_hood