98% токенов впустую

16 советов как не сжигать контекст и продлить полезное окно работы

Что вы получите из этого гайда

- Понимание, **куда реально уходят токены** в Claude Code и почему лимит заканчивается так быстро.
- 16 практических приёмов, чтобы продлить полезное окно работы без перехода на дорогой тариф.
- 5 топ-приёмов для срочного внедрения - если попробуете только их, разница будет заметна сразу.
- Правила для `CLAUDE.md` и `MEMORY.md`, которые держат контекст лёгким.

Время чтения: ≈ 14 минут. **Уровень:** для тех, кто уже неделю работает с Claude Code и регулярно упирается в лимиты.

> Если ещё не пользовались Claude Code - сначала возьмите «Клод Код за вечер». Этот гайд предполагает, что у вас уже есть рабочий проект и сессии в чате.

Почему лимит уходит так быстро

«Открываю сессию, делаю пару задач - и лимит на исходе. Чувство такое, что ничего особенного не сделал». Знакомо? Это самый частый вопрос, который мне приходит.

Чтобы советы дальше стали очевидными, разберёмся **как работают токены** - тогда станет понятно, что именно их съедает.

Факт 1. Один токен ≈ 0,75 слова

1000 токенов - это примерно 750 слов. Каждое слово в вашем сообщении и в ответе Claude стоит токены. Ничего нового.

Факт 2. Claude перечитывает весь разговор каждый раз

Это и есть главная причина, почему стоимость растёт неравномерно. Когда вы отправляете новое сообщение, Claude читает не только его - он перечитывает всю историю окна с самого начала. Ваше первое сообщение, его ответ, второе, его ответ - и так до текущего.

Поэтому **первое сообщение может стоить 500 токенов, а тридцатое - уже 15 000**. Просто потому что Claude перечитал всю историю до того, как перешёл к новому.

> Один разработчик замерил: в чате из 100 сообщений **до 98% токенов** уходит на перечитывание старой истории. Не на вашу задачу - просто на повторное чтение того, что уже было.

Факт 3. Свежая сессия уже стоит токены - до первого сообщения

Прежде чем вы напишете хотя бы одно слово, Claude уже потратил токены. При запуске новой сессии он загружает: системные промпты, файл `CLAUDE.md`, файл `MEMORY.md`, подключённые MCP-серверы, доступные скиллы и тулы.

На моём аккаунте Pro с моделью Sonnet (контекстное окно - 200 000 токенов) свежая сессия без единого сообщения уже **съедает около 22 000 токенов - это 11% окна**. Проверить можно командой `/context` - она покажет распределение по категориям.

Факт 4. Переполненный контекст хуже работает

Если в сессии много сообщений, Claude **лучше всего видит начало и конец**. Середина обрабатывается заметно хуже. Получается двойная потеря: вы платите больше токенами, и при этом результат получаете хуже.

Когда понятно, что́ест токены, советы дальше становятся очевидными.

16 приёмов чтобы не сжигать контекст

Приём 1. Новый разговор на каждую задачу

Самый простой и самый важный. Если переходите к другой задаче - открывайте новое окно или используйте `/clear`.

`/clear` очищает историю прямо в текущем окне - не нужно открывать новую вкладку. Сначала пугаешься: «А вдруг Claude забудет что-то важное?». Не забудет, если попросить - «запомни это перед тем как очистить».

У меня для этого собран скилл `learn`: он перед каждым `/clear` сохраняет принятые решения, статусы задач, идеи в `MEMORY.md`. Если что-то нужно записать в `CLAUDE.md` - спрашивает у меня (чтобы не засорять). Один вызов - и можно смело очищать.

Приём 2. Объединяйте запросы в одно сообщение

Это прямо следствие из факта 2. Три отдельных сообщения стоят примерно в **3 раза дороже**, чем одно с тремя пунктами. Каждое отдельное сообщение - это перечитывание всей истории.

Если у вас несколько связанных задач уже в голове - отправьте их вместе:

> «Сделай три вещи: 1) собери данные из писем за март, 2) подготовь шаблон отчёта, 3) проверь оба результата на совпадение цифр».

Один проход истории - три задачи решены.

Приём 3. Plan Mode перед большой задачей

Самый дорогой сценарий - когда Claude ушёл в неправильном направлении, сделал кучу изменений, и оказывается, что это не то. Токены потрачены впустую, всё переделывать.

Plan Mode решает это. Claude составляет план, задаёт уточняющие вопросы, убеждается что понял правильно - и только потом выполняет.

Включается переключателем внизу чата или `Shift+Tab`. Используйте перед созданием скилла, перед сложной автоматизацией, перед любой задачей где результат не очевиден с первого раза.

Приём 4. Не уходите, пока Claude не пошёл в правильном направлении

Не отправляйте задачу и не уходите делать что-то другое. Иногда Claude начинает работать и зависает: ходит не туда, читает одни и те же файлы снова и снова.

Если видите это - **Escape** мгновенно останавливает работу. Токены на застрявшую петлю не тратятся. Корректируете задачу, запускаете заново.

Хотя бы первые 30 секунд после старта проследите - убедитесь, что Claude подключился к нужным файлам и пошёл в нужном направлении. Это сильно экономит и время, и токены.

Приём 5. Будьте точны с файлами и запросами

Перед тем как вставить большой файл - подумайте, нужен ли он Claude целиком или только кусочек. Ошибка в одной функции? Отдайте только её. Нужно перевести один абзац? Дайте только этот абзац. Лишний контекст - не помощь, а лишние токены.

Та же логика с формулировками. Вместо «улучши этот проект» или «проанализируй всю папку с документами» (Claude начнёт читать всё подряд, пытаясь понять с чего начать) - конкретно: «есть такие-то файлы, проанализируй их в первую очередь и от них отталкивайся».

Чем точнее запрос - тем меньше лишних чтений и тем меньше расход.

Приём 6. Используйте `/context` чтобы увидеть, куда уходят токены

Если не видишь проблему - не решишь её. Команда `/context` показывает текущее распределение токенов по категориям: системные промпты, MCP, `CLAUDE.md`, `MEMORY.md`, скиллы, сообщения, свободное место.

Какие-то категории не контролируются - системные тулы, базовые инструкции. Но `CLAUDE.md`, `MEMORY.md` и скиллы - **полностью под вашим управлением**. И обычно именно они разрастаются, когда не контролируешь.

Приём 7. Используйте `/cost` для проверки стоимости (если работаете через API)

`/cost` показывает сколько стоит текущая сессия. **Важная оговорка:** актуально только для тех, кто работает через Anthropic API с pay-per-token. Если у вас подписка Pro или Max - команда покажет «subscription», и стоимости вы не увидите. Платите фиксированную сумму в месяц.

Приём 8. Не давайте лимиту застать вас врасплох

Открывайте дашборд лимитов **до того как сядете работать**. На claude.ai → Settings → Usage. Видите две главные вещи: текущая сессия (% использования + когда обнулится) и недельный лимит.

Я держу эту вкладку открытой и время от времени переключаюсь, чтобы видеть запас. Простой habit - но избавляет от сюрприза «лимит закончился посреди важной задачи».

Приём 9. Отключайте MCP-серверы которые не используете

MCP - это, наверное, **самая тяжёлая статья расхода**. Один MCP-сервер может съедать **до 18 000 токенов за каждое сообщение**. Не за сессию - за каждое отдельное сообщение.

Посмотреть подключённые MCP - команда `/mcp`. Если какой-то сервер подключён, но вы им не пользуетесь сейчас - отключите. Кликаете на «Connected» → «Disable». Переподключить можно в любой момент.

Принцип: **подключено только то, что используется в текущей задаче**. Не «на всякий случай».

Приём 10. Сжимайте контекст сами при 50-60%, не ждите автоматики

При 95-100% использования контекста Claude автоматически сжимает историю. Но к этому моменту он уже работает заметно хуже - переполненный контекст игнорирует середину.

Не ждите автоматики. Запускайте `/compact` сами при 50-60% использования. И при вызове **прямо укажите**, что Claude должен сохранить:

> «/compact - сохрани решения по архитектуре проекта, текущий статус скилла `process-leads` и настройки MCP».

Так вы контролируете, что точно не потеряется при сжатии.

Альтернатива: добавить в `CLAUDE.md` раздел с инструкциями по сжатию - что всегда нужно сохранять. Тогда даже автоматическое сжатие не выкинет важное.

Приём 11. После 3-4 сжатий - переходите в новую сессию

Если в одном окне вы уже сделали 3-4 `/compact`, лучше перейти в новую сессию. Сжатый контекст всё равно накапливает шум.

Простой рецепт: попросите Claude дать summary сессии перед переходом.

> «Я перехожу в новое окно. Дай, пожалуйста, краткий итог нашего разговора: что сделано, что осталось, какие решения приняты».

Копируете итог, делаете `/clear`, в новом окне вставляете summary первым сообщением. Контекст перенесён, окно чистое.

Приём 12. Выбирайте правильную модель под задачу

Не все задачи требуют самой мощной модели. По умолчанию подключена **Sonnet** - подходит для большинства. Проверить текущую модель - команда `/model`.

- **Opus** - для сложных задач: создание скилла, детальный план, обработка большого промпта.
- **Sonnet** - для большинства повседневных задач (default).
- **Haiku** - для простых: переименовать файлы, сделать сортировку, ответить на короткий вопрос, перевести.

Понимание «какая задача под какую модель» приходит с практикой - чем больше работаете, тем точнее чувствуете.

Приём 13. Запускайте субагентов на Haiku вместо Sonnet

Субагенты - удобно, но **дорого**: каждый запускается со своим полным контекстом. Получается в 7-10 раз дороже обычной сессии.

Приём: **прописать в `CLAUDE.md` инструкцию запускать субагентов на Haiku** для простых задач. Большая часть токенов уйдёт на дешёвую модель, результат тот же.

> «Все субагенты запускаются на модели Haiku, кроме случаев когда задача требует анализа сложной логики или генерации кода - тогда Sonnet».

Приём 14. Держите `CLAUDE.md` коротким

`CLAUDE.md` читается **при каждой новой сессии без исключений**. Если файл разрастается - каждое сообщение становится дороже.

Официальная рекомендация Anthropic: **меньше 200 строк**. В идеале - около 60.

Главный сдвиг в восприятии: **`CLAUDE.md` - это не хранилище, а индекс**. Как оглавление в книге - не пересказывает содержание, а подсказывает где что лежит.

Мой подход: - Первые несколько строк: для чего проект, кто я, очень кратко. - Дальше - **references**: «Описание бизнеса и услуги → путь к файлу», «Стиль общения с клиентами → путь к файлу», «Бренд и цвета → путь к файлу».

Когда нужны детали - Claude знает где искать. Файл лёгкий, ничего лишнего, инструкция чёткая.

И отдельно: **подробные инструкции под конкретные задачи не пишите в `CLAUDE.md`**. Для этого есть скиллы - они загружаются только при вызове, а не в каждой сессии.

Приём 15. Чистите `MEMORY.md` периодически

`MEMORY.md` тоже читается при каждом запуске. И тоже разрастается: устаревшие статусы задач, дубликаты с другими файлами, детали которые уже не актуальны.

Найти файл руками сложно. Просто попросите Claude:

> «Открой мой `MEMORY.md` файл».

И затем:

> «Проанализируй этот файл и дай список рекомендаций - что можно удалить, что устарело, как лучше структурировать».

В моём случае было 130 строк, после чистки осталось 65. Половина - выжимка вашего контекста, не разросшийся мусор.

Приём 16. Снижайте extended thinking для простых задач

По умолчанию Claude **думает перед каждым ответом** - это extended thinking. На само мышление тоже уходят токены.

Для сложных задач это оправдано: глубже анализирует, меньше ошибается. Но если вы просите переименовать файлы, перевести абзац, ответить на короткий вопрос - токены на «глубокое мышление» уходят впустую, результат был бы тот же.

Команда `/effort` показывает текущий уровень. У меня по умолчанию High на Sonnet. Можно переключить на Medium для повседневных задач, на Low - для совсем простых. На Opus есть ещё четвёртый уровень.

Простое правило: **Medium хватает для большинства повседневных задач**. На High возвращайтесь когда задача реально сложная.

Если возьмёте только 5 приёмов

16 - это много. Если хочется внедрить только то, что даст эффект быстрее всего:

1. **Приём 1. Новый разговор на каждую задачу** - свежий контекст всегда лучше работает.
2. **Приём 2. Объединяйте запросы** - одно сообщение с тремя пунктами вместо трёх отдельных.
3. **Приём 5. Точность с файлами и запросами** - дайте кусочек, не весь файл.
4. **Приём 6. Команда `/context`** - смотрите куда уходят токены, прежде чем оптимизировать.
5. **Приём 10. `/compact` сами при 50-60%** - не ждите автоматики, она работает плохо.

Это не вопрос тарифа. Это вопрос привычек. На Pro можно работать продуктивно весь день, если не прогонять Claude Code десятки раз через одну и ту же историю.

Если регулярно упираетесь в лимиты - это, кстати, **хороший знак**: значит реально работаете с инструментом, а не пробуете теорию. Эти приёмы как раз для тех, кто внедряет на практике.

После этого гайда

Когда работа с контекстом стала экономной - возвращайтесь в каталог за следующими материалами:

- **«17 приёмов в Клод Код»** - конкретные приёмы из ежедневной работы. Этот гайд - углубление в Блок 2 «17 приёмов».
- **«Скиллы в Клод Код с нуля»** - как собрать первый скилл за вечер. Развитие приёма про автоматизацию повторяющихся задач.
- **«Клод Код + Notebook LM»** - связка двух инструментов для исследований и контента.

Подписывайтесь, чтобы видеть новое

YouTube · **@ai_with_marina** Подробные разборы AI-инструментов и пошаговые гайды для русскоязычной аудитории. Новое видео раз в неделю.

Telegram · **@ai_with_marina** Короткие заметки, анонсы новых материалов AIAtlas, обсуждения и быстрые ответы на вопросы.

Если что-то в гайде не работает или нашли неточность - напишите в Telegram-канал, отвечу.